

6man Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

S. Jiang
Huawei Technologies Co., Ltd
G. Chen
China Mobile
S. Krishnan
Ericsson
October 22, 2012

A Generic IPv6 Addresses Registration Solution Using DHCPv6
draft-ietf-dhc-addr-registration-01

Abstract

In networks that are centrally managed, self-generated addresses cause traceability issues due to their decentralized nature. To minimize the issues due to lack of traceability, these self-generated addresses can be registered with the network for allowing centralized address administration. This document defines a generic address registration solution using DHCPv6, using a new ND option and a new DHCPv6 option in order to communicate the use of self-generated addresses.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Overview of Generic Address Registration Solution	3
4. Propagating the Address Registration Solicitation	4
4.1. ND Address Registration Solicitation Option	5
4.2. DHCPv6 Address Registration Solicitation Option	6
5. DHCPv6 Address Registration Procedure	7
5.1. DHCPv6 Address Registration Request	7
5.2. DHCPv6 Address Registration Acknowledge	8
6. Security Considerations	8
7. IANA Considerations	8
8. Acknowledgements	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Authors' Addresses	10

1. Introduction

In several common network scenarios, IPv6 addresses are self-generated by the end-hosts using some information propagated to them by the network (i.e. the network prefix). Examples of self-generated addresses include those created using IPv6 Stateless Address Configuration [RFC4862], temporary addresses [RFC4941] and Cryptographically Generated Addresses (CGA) [RFC3972] etc. These addresses are potentially incompatible with networks with a centrally managed address architecture such as DHCPv6 [RFC3315] as they lack traceability and stability.

Many operators of enterprise networks and similarly tightly administered networks have expressed the desire to be at least aware of the hosts' self-generated addresses when migrating to IPv6.

One potential way to provide network administrators with most of their needs while retaining compatibility with normal stateless configuration would be to register the self-generated addresses with the systems in place to centrally administer the addresses. The host may be required to perform this registration in some scenarios since only registered IPv6 addresses may be granted access to the network resources.

This document introduces a new IPv6 Neighbor Discovery option and a new DHCPv6 option to propagate the address registration information from the hosts to the network. The DHCPv6 protocol is used to perform the address registration procedure while the address registration server role may be performed by a DHCPv6 server or a stand-alone server.

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. Overview of Generic Address Registration Solution

In the generic address registration solution, the network management system solicits hosts to register their self-generated addresses, by sending solicitation messages from either local router (step 1a in Figure 1) or DHCPv6 server (step 1b in Figure 1).

After receiving such solicitations, a host implementing this specification and using a self-generated address SHOULD send an

address registration request message to the address registration server (step 2 in Figure 1). The address registration server may be acted by a DHCPv6 server. By received the address registration request, the address registration server records the requested address in the address database, which MAY be used by other network functions, such as DNS or ACL, etc. The address registration server should also assign lifetimes for the requested address. An acknowledgement is sent back to the host with the assigned lifetimes (step 3 in Figure 1).

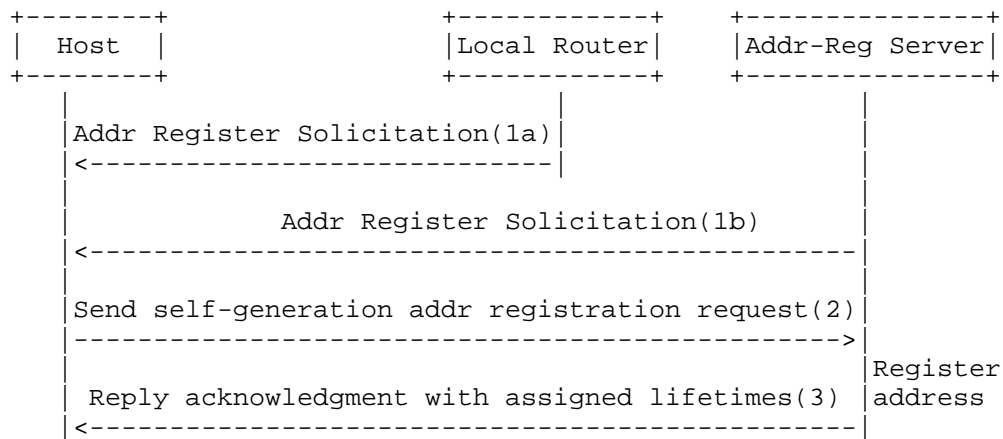


Figure 1: Address Registration Procedure

By received the acknowledgement, the host can continue use the registered address. It SHOULD use the assigned preferred and valid lifetime for the correspondending address.

4. Propagating the Address Registration Solicitation

In order to request the hosts with self-generated addresses to register their addresses and the appointed address registration server, new solicitation options are defined.

There is more than one mechanism by which configuration parameters could be pushed to the end hosts. The address registration solicitation option can be carried in Router Advertisement (RA) message, which is broadcasted by local routers. In the DHCPv6 managed network, it can also be carried in DHCPv6 messages.

This document defines a new ND option and one new DHCPv6 option that convey a Fully Qualified Domain Name (FQDN, as per Section 3.1 of [RFC1035] to be used as the destination of the address registration

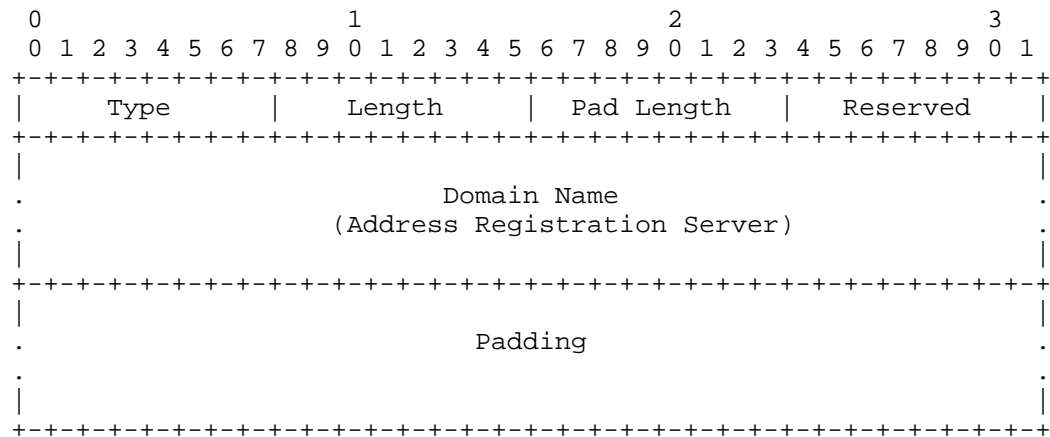
messages. In order to make use of these options, this document assumes that appropriate name resolution mechanisms (see Section 6.1.1 of [RFC1123]) are available on the host.

After receiving a message containing an address registration solicitation option, a host implementing this specification SHOULD register its self-generated addresses, if any, to the announced address registration server. The solicitation options MAY include the IPv6 address(es) of address registration server.

In principle, hosts need to receive a prefix from either RA message [RFC4861] or DHCPv6 message [I-D.ietf-dhc-host-gen-id] so that they can generate an IPv6 address by themselves. The Address Registration Solicitation options are expected to be propagated along with prefix assignment information.

4.1. ND Address Registration Solicitation Option

The ND Address Registration Solicitation Option allows routers to propagate the solicitation for hosts to register their self-generated address. This option also carries the fully qualified domain name of the address registration server. This option SHOULD be propagated together with the Prefix Information Option, Section 4.6.2, [RFC4861]. The format of the ND Address Registration Solicitation Option is described as follows:



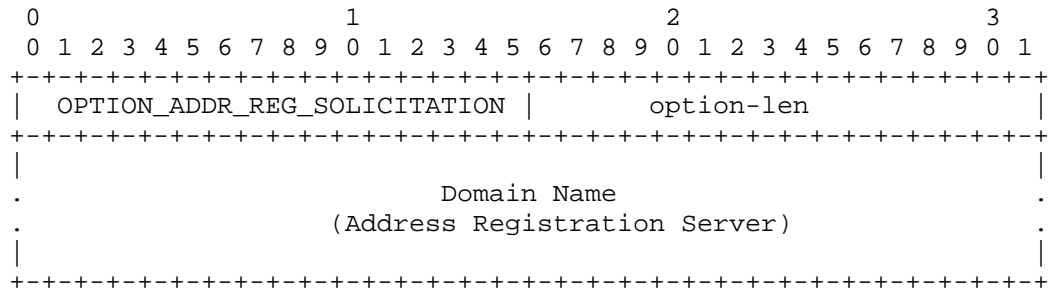
Fields:

Type	TBA1
Length	The length of the option in units of 8 octets, including the Type and Length fields. The value 0 is invalid. The receiver MUST discard a message that contains this value.
Pad Length	The number of padding octets beyond the end of the Domain Name field but within the length specified by the Length field.
Reserved	Padding bits. It is for future use also. The value MUST be initialized to zero by the sender, and MUST be ignored by the receiver.
Domain Name	Fully qualified domain name of the announced address registration server. The domain name is encoded as specified in Section 8 of [RFC3315].
Padding	A variable-length field making the option length a multiple of 8, containing as many octets as specified in the Pad Length field. Padding octets MUST be set to zero by senders and ignored by receivers.

4.2. DHCPv6 Address Registration Solicitation Option

The DHCPv6 Address Registration Solicitation Option allows DHCPv6 server to propagate the solicitation for hosts to register their self-generated address. This option also carries a domain name of

the appointed address registration server. This option SHOULD be propagated together with DHCPv6 Prefix Information Option, Section 5, [I-D.ietf-dhc-host-gen-id]. The format of the DHCPv6 Address Registration Solicitation Option is described as follows:



option-code OPTION_ADDR_REG_SOLICITATION (TBA2).

option-len Length of this option in octets (not including option-code and option-len).

Domain Name A fully qualified domain name of the appointed address registration server. The domain name is encoded as specified in Section 8 of [RFC3315].

5. DHCPv6 Address Registration Procedure

The DHCPv6 protocol is reused as the address registration protocol while a DHCPv6 server can play the role of an address registration server. The IA_NA DHCPv6 option [RFC3315] is reused in order to fulfill the address registration interactions.

5.1. DHCPv6 Address Registration Request

The host with one or more self-generated addresses sends a DHCPv6 Request message to the address registration server received in the address registration solicitation option.

The DHCPv6 Request message SHOULD contain at least one IA_NA option. The IA_NA option SHOULD contain at least one IA Address option. The host 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.

After receiving this address registration request, the address registration server MUST register the requested address in its

address database, which may further be used by other network functions, such as DNS, network access control lists, etc. The address registration server SHOULD also assign the lifetimes for these registered addresses.

The centrally managed address database contains both self-generated addresses and the DHCPv6 assigned addresses. They MAY be marked and treated differently in the database.

5.2. DHCPv6 Address Registration Acknowledge

The address registration server then sends a Reply message as the response to registration requests. The DHCPv6 Reply message SHOULD contain at least one IA_NA option. The IA_NA option SHOULD contain at least one IA Address option. The server 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 following the rules defined in Section 22 of [RFC3315].

After receiving the acknowledgement from the server, the host can use the registered address to access the network. It SHOULD use the values in the preferred and valid lifetime fields of the received message to determine the preferred and valid lifetimes of the address.

Please note that the host MAY continue to use expired address, such as Upper-Layer Identifiers (ULID) in Shim6 protocol [RFC5533], etc. but the network could potentially refuse the network access from such addresses.

6. Security Considerations

An attacker may use a faked address registration request option to indicate hosts reports their address to a malicious server and collect the user information. An attacker may also register large number of fake addresses with the network in order to overwhelm the address registration server. In either case, these attacks may be prevented by using Secure Neighbor Discovery [RFC3971] if the RA Address Registration Request Option is used, and the AUTH option [RFC3315] or Secure DHCPv6 [I-D.ietf-dhc-secure-dhcpv6] if the DHCPv6 Address Registration Request Option is used.

7. IANA Considerations

This document defines a new IPv6 Neighbor Discovery option, the Address Registration Solicitation Option (TBA1) described in Section

4.1, that requires an allocation out of the registry defined at

<http://www.iana.org/assignments/icmpv6-parameters>

This document defines a new DHCPv6 option, the
OPTION_ADDR_REG_SOLICITATION (TBA2) described in Section 4.2, that
requires an allocation out of the registry defined at

<http://www.iana.org/assignments/dhcpv6-parameters/>

8. Acknowledgements

The authors would like to thank Ralph Droms, Ted Lemon, Bernie Volz and other members of dhc working group for their valuable comments.

9. References

9.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [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.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 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.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

9.2. Informative References

- [I-D.ietf-dhc-host-gen-id]
Jiang, S., Xia, F., and B. Sarikaya, "Prefix Assignment in DHCPv6", draft-ietf-dhc-host-gen-id-04 (work in progress), August 2012.
- [I-D.ietf-dhc-secure-dhcpv6]
Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs", draft-ietf-dhc-secure-dhcpv6-07 (work in progress), September 2012.

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

Gang Chen
China Mobile
53A, Xibianmennei Ave., Xuanwu District, Beijing
P.R. China

Phone: 86-13910710674
Email: phdgang@gmail.com

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

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

Network Working Group
Internet Draft
Intended status: Standards Track
Expires: April 11, 2013

Sheng Jiang
Sam(Zhongqi) Xia
Huawei Technologies Co., Ltd
October 10, 2012

Configuring Cryptographically Generated Addresses (CGA) using DHCPv6
draft-ietf-dhc-cga-config-dhcpv6-03

Status of this Memo

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

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

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

This Internet-Draft will expire on April 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

A Cryptographically Generated Address is an IPv6 addresses binding with a public/private key pair. However, the current CGA specifications are lack of procedures to enable proper management of the usage of CGAs. This document defines the process using DHCPv6 to manage CGAs in detail. A new DHCPv6 option is defined accordingly. This document also analyses the configuration of the parameters, which are used to generate CGAs, using DHCPv6. Although the document does not define new DHCPv6 option to carry these parameters for various reasons, the configuration procedure is described.

Table of Contents

1. Introduction	3
2. Terminology	3
3. CGA Configure Process Using DHCPv6	4
3.1. Configuration of the parameters required for the generation of CGA	4
3.2. Host requests CGA Approved to the DHCPv6 server	5
4. CGA Grant Option	7
5. Security Considerations	8
6. IANA Considerations	8
7. Acknowledgments	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Author's Addresses	10

1. Introduction

Cryptographically Generated Addresses (CGA, [RFC3972]) provide means to verify the ownership of IPv6 addresses without requiring any security infrastructure such as a certification authority.

CGAs were originally designed for SeND [RFC3971] and SeND is generally not used in the same environment as a Dynamic Host Configure Protocol for IPv6 (DHCPv6) [RFC3315] server. However, after CGA has been defined, as an independent security property, many other CGA usages have been proposed and defined, such as Site Multihoming by IPv6 Intermediation (SHIM6) [RFC5533], Enhanced Route Optimization for Mobile IPv6 [RFC4866], also using the CGA for DHCP security purpose [I-D.ietf-dhc-secure-dhcpv6], etc. The use of CGAs allows identity verification in different protocols. In these scenarios, CGAs may be used in DHCPv6-managed networks.

As [I-D.ietf-csi-dhcpv6-cga-ps] analyses, in the current specifications, there is a lack of procedures to enable proper management of the usage of CGAs. Particularly, in a DHCPv6-managed network, a new DHCPv6 option is missed, therefore, the DHCPv6 server can NOT grant the use of host-generated CGA addresses on request from the client, or reject the CGA on the basis of a too-low sec value. In order to fill this gap, a new DHCPv6 option, CGA Grant Option, is defined in this document.

This document also analyses the configuration of the parameters, which are used to generate CGAs, using DHCPv6. Although the document does not define new DHCPv6 option to carry these parameters for various reasons, the configuration procedure is described. The procedure works with existing options or future define options.

The CGA configuration procedure described in this document can work with a generic address registration mechanism, which discussed in IETF DHC Working Group, but have not been defined yet. However, even a generic address registration mechanism was defined, the CGA-specific option, CGA Grant Option, is still needed so that DHCP server can indicate hosts the recommended CGA Sec value.

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 [RFC2119].

3. CGA Configure Process Using DHCPv6

The CGA specifications [RFC3972] define the procedure to generate a CGA. However, it assumes that hosts decide by itself or have been preconfigured all CGA relevant parameters. In reality, the network management MAY want to assign/enforcement some parameters to hosts; the network management MAY also manage the use of CGAs.

Among the mechanisms in which configuration parameters could be pushed to the end hosts and/or CGA related information sent back to a central administration, we discuss the stateful configuration mechanism based on DHCPv6 in this document. Other mechanisms may also provide similar functions, but out of scope.

In this section, configuration CGA parameters and that a DHCPv6 server grants the CGA usage are described in details.

3.1. Configuration of the parameters required for the generation of CGA

Each CGA is associated with a CGA Parameters data structure, which is formed by all input parameters [RFC3972] except for Sec value that is embedded in the CGA. The CGA associated Parameters used to generate a CGA includes:

- a Public Key,
- a Subnet Prefix,
- a 3-bit security parameter, Sec. Additionally, it should be noted that the hash algorithm to be used in the generation of the CGA is also defined by the Sec value [RFC4982],
- any Extension Fields that could be used.
- Note: the modifier and the Collision Count value in the CGA Parameter data structure are generated during the CGA generation process. They do NOT need to be configured.

In a DHCPv6 managed network, a host may initiate a request for the relevant CGA configuration information needed to the DHCPv6 server. The server responds with the configuration information for the host. The Option Request Option, defined in Section 22.7 in [RFC3315], can be used for host to indicate which options the client requests from the server. For response, the requested Option should be included. The server MAY also initiatively push these parameters by attaching these option in the response messages which are initiated for other purposes.

- The Public/Private key pair is generated by hosts themselves and considered not suitable for network transmission for security reasons. The configuration of the client key pair or certificate is out of scope.

- Currently, there are convenient mechanisms for allowing an administrator to configure the subnet prefix for a host, by Router Advertisement [RFC4861, RFC4862]. However, this does not suit for the DHCP-managed network. To propagate the prefix through DHCP interactions, DHCPv6 Prefix Delegation Option [RFC3633] MAY be used. However, this option was designed to assign prefix block for routers. A new Prefix Assignment Option MAY need to be defined. Since alternative approach is existing and there are debates whether a new Prefix Assignment Option MAY is necessary, this document does not define it.

- Although the network management MAY want to enforce or configure a Sec value to the hosts, it is considered as a very dangerous action. A malicious fake server may send out a high Sec value to attack clients giving the fact that generation a CGA with a high Sec value is very computational intensive [I-D.ietf-csi-dhcpv6-cga-ps]. Another risk is that a malicious server could propagate a Sec value providing less protection than intended by the network administrator, facilitating a brute force attack against the hash, or the selection of the weakest hash algorithm available for CGA definition. A recommendation Sec value is considered as confusion information. The receiving host is lack for information to make choose whether generates a CGA according to the recommendation or not. Therefore, the document does not define a DHCPv6 option to propagate the Sec value.

- Although there is an optional Extension Fields in CGA Parameter data structure, there is NO any defined extension fields. If in the future, new Extension Fields in CGA Parameter data structure are defined, future specification may define correspondent DHCPv6 options to carry these parameters.

Upon reception of the CGA relevant parameters from DHCPv6 server, the end hosts SHOULD generate addresses compliant with the received parameters. If the parameters change, the end hosts SHOULD generate new addresses compliant with the parameters propagated.

3.2. Host requests CGA Approved to the DHCPv6 server

A CGA address is generated by the associated key pair owner, normally an end host. However, in a DHCPv6-managed network, hosts should use IPv6 global addresses only from a DHCPv6 server. The process

described below allows a host, also DHCPv6 client, uses self-generated CGAs in a DHCPv6-managed environment, by requesting the granting from a DHCPv6 server.

The client sends a CGA, which is generated by itself, to a DHCPv6 server, and requests the DHCP server to determine whether the generated CGA satisfies the requirements of the network configuration, wherein the network configuration comprises a CGA security level set by the DHCP; and generates a new CGA if the generated CGA does not satisfy the requirements of the network configuration.

Client initiation behavior

In details, a DHCPv6 client SHOULD send a DHCPv6 Request message to initiate the CGA granting process.

This DHCPv6 Request message MUST include an Option Request option, which requests the CGA Grant Option, defined in Section 4 in this document, to indicate the DHCPv6 server responses with the address granting decision.

The client MUST include one or more IA Options, either IA_NA or IA_TA, in the Request message. Each IA Option MUST include one or more IA Address Options. CGAs are carried in the IA Address Options.

Server behavior

Upon reception of the Request message, the DHCPv6 server SHOULD verify whether the client's CGAs satisfy the CGA-related configuration parameters of the network. The DHCPv6 server SHOULD NOT handle the Request which the CGA_Grant field is not all 1(FFx). The DHCPv6 server then send an acknowledgement, a Reply message, to the client to either grant the use of the CGA or decline the requested CGA. The CGA_Grant field SHOULD be set following the rule, defined in Section 4 in this document. When the requested CGA is declined, the DHCPv6 server MAY also recommend a Sec value to the client using the CGA Grant option in the DHCPv6 Reply message.

In the meantime, the DHCPv6 server MAY log the requested CGA addresses. This information MAY later be used by other network functions, such as ACL.

Client receiving behavior

Upon reception of the acknowledgement from server, the client can legally use the granted CGAs. The client SHOULD silently drop any

message that has the CGA_Grant field set any other value, but F0x, or 00x~07x. If the server declines the requested CGA, the client MAY generate a new CGA with the recommended Sec value. If the server replies with CGA-relevant parameters, the client MAY generate a new CGA accordingly.

4. CGA Grant Option

DHCPv6 CGA Grant Option is used to indicate the DHCPv6 client whether the requested address is granted or not. In the decline case, a recommended Sec value MAY be sent, too.

```

      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_ADDR_GRANT          |          option-len          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   CGA Grant   |
+-----+-----+

```

option-code

OPTION_ADDR_GRANT (TBA1).

option-len

1.

CGA_Grant

The CGA_Grant field sets all 1 (FFx) when a client requests granting from server in the DHCPv6 Request message.

In the DHCPv6 reply message, the CGA_Grant field sets F0x to indicate that the requested CGA is granted; it sets 00x to indicate that the requested Address is declined without any recommended Sec value. It sets 01x~07x to indicate that requested Address is declined and the recommended Sec value (value from 1~7).

Note: On receiving the CGA Grant Option with reject information and a recommended Sec value, the client MAY generate a new CGA with the recommended Sec value. . If choosing not use the recommended Sec value, the client MAY take the risk that it is not able to use full network capabilities. The network may consider the hosts that use CGAs with lower Sec values as unsecure users and decline some or all network services.

5. Security Considerations

The mechanisms based on DHCPv6 are all vulnerable to attacks to the DHCP client. Proper use of DHCPv6 autoconfiguration facilities [RFC3315], such as AUTH option or Secure DHCP [I-D.ietf-dhc-secure-dhcpv6] can prevent these threats, provided that a configuration token is known to both the client and the server.

IF a DHCPv6 server rejected a client CGA based on a certain Sec value, it SHOULD NOT suggest a new Sec value either equal or lower than the Sec value that has been rejected.

Note that, as expected, it is not possible to provide secure configuration of CGA without a previous configuration of security information at the client (either a trust anchor, or a DHCPv6 configuration token, etc.). However, considering that the values of these elements could be shared by the hosts in the network segment, these security elements can be configured more easily in the end hosts than its addresses.

6. IANA Considerations

This document defines two new DHCPv6 [RFC3315] options, which must be assigned Option Type values within the option numbering space for DHCPv6 messages:

The DHCPv6 CGA Grant Option, OPTION_ADDR_GRANT (TBA1), described in Section 4.

7. Acknowledgments

The authors would like to thank Marcelo Bagnulo Braun and Alberto Garcia-Martinez for been involved in the early requirement identification. Valuable comments from Bernie Volz, Ted Lemon, John Jason Brzozowski, Dujuan Gu and other DHC WG members are appreciated.

8. References

8.1. Normative References

- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997.
- [RFC3315] R. Droms, Ed., "Dynamic Host Configure Protocol for IPv6", RFC3315, July 2003.

- [RFC3633] O. Troan and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3971] J. Arkko, J. Kempf, B. Zill and P. Nikander, "SEcure Neighbor Discovery (SEND) ", RFC 3971, March 2005.
- [RFC3972] T. Aura, "Cryptographically Generated Address", RFC3972, March 2005.
- [RFC4861] T. Narten, et al., "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] S. Thomson, T. Narten and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC4862, September 2007.
- [RFC4866] J. Arkko, C. Vogt and W. Haddad, "Enhanced Route Optimization for Mobile IPv6", RFC4866, May 2007.
- [RFC4982] M. Bagnulo, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs) ", RFC4982, July 2007.
- [RFC5533] E. Nordmark and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6" RFC 5533, June 2009.

8.2. Informative References

- [I-D.ietf-csi-dhcpv6-cga-ps]
S. Jiang, S. Shen and T. Chown, "DHCPv6 and CGA Interaction: Problem Statement", draft-ietf-csi-dhcpv6-cga-ps (work in progress), February, 2012.
- [I-D.ietf-dhc-secure-dhcpv6]
S. Jiang and S. Shen, "Secure DHCPv6 Using CGAs", draft-ietf-dhc-secure-dhcpv6 (work in progress), September, 2012.

Author's Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Q14 Huawei Campus, 156 BeiQi Road,
ZhongGuan Cun, Hai-Dian District, Beijing 100085
P.R. China
Email: jiangsheng@huawei.com

Sam(Zhongqi) Xia
Huawei Technologies Co., Ltd
Q14 Huawei Campus, 156 BeiQi Road,
ZhongGuan Cun, Hai-Dian District, Beijing 100085
P.R. China
Email: xiazhongqi@huawei.com

Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

T. Mrugalski
ISC
K. Kinnear
Cisco
October 22, 2012

DHCPv6 Failover Design
draft-ietf-dhc-dhcpv6-failover-design-02

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Language	4
2. Glossary	4
3. Introduction	5
3.1. Additional Requirements	6
3.2. Features out of Scope: Load Balancing	6
4. Protocol Overview	6
4.1. Failover State Machine Overview	8
4.2. Messages	9
5. Connection Management	11
5.1. Creating Connections	11
5.2. Endpoint Identification	12
6. Resource Allocation	13
6.1. Proportional Allocation	14
6.2. Independent Allocation	16
6.3. Choosing Allocation Algorithm	16
7. Information model	17
8. Failover Mechanisms	21
8.1. Time Skew	21
8.2. Time expression	22
8.3. Lazy updates	22
8.4. MCLT concept	22
8.4.1. MCLT example	24
8.5. Unreachability detection	25
8.6. Re-allocating Leases	25
8.7. Sending Binding Update	26
8.8. Receiving Binding Update	28
8.9. Conflict Resolution	28
8.10. Acknowledging Reception	30
9. Endpoint States	30
9.1. State Machine Operation	30
9.2. State Machine Initialization	33
9.3. STARTUP State	33
9.3.1. Operation in STARTUP State	34
9.3.2. Transition Out of STARTUP State	34
9.4. PARTNER-DOWN State	35
9.4.1. Operation in PARTNER-DOWN State	35
9.4.2. Transition Out of PARTNER-DOWN State	36
9.5. RECOVER State	37
9.5.1. Operation in RECOVER State	37

9.5.2. Transition Out of RECOVER State	37
9.6. RECOVER-WAIT State	39
9.6.1. Operation in RECOVER-WAIT State	40
9.6.2. Transition Out of RECOVER-WAIT State	40
9.7. RECOVER-DONE State	40
9.7.1. Operation in RECOVER-DONE State	41
9.7.2. Transition Out of RECOVER-DONE State	41
9.8. NORMAL State	41
9.8.1. Operation in NORMAL State	41
9.8.2. Transition Out of NORMAL State	42
9.9. COMMUNICATIONS-INTERRUPTED State	43
9.9.1. Operation in COMMUNICATIONS-INTERRUPTED State	43
9.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State	44
9.10. POTENTIAL-CONFLICT State	45
9.10.1. Operation in POTENTIAL-CONFLICT State	46
9.10.2. Transition Out of POTENTIAL-CONFLICT State	46
9.11. RESOLUTION-INTERRUPTED State	47
9.11.1. Operation in RESOLUTION-INTERRUPTED State	48
9.11.2. Transition Out of RESOLUTION-INTERRUPTED State	48
9.12. CONFLICT-DONE State	48
9.12.1. Operation in CONFLICT-DONE State	48
9.12.2. Transition Out of CONFLICT-DONE State	49
10. Proposed extensions	49
10.1. Active-active mode	49
11. Dynamic DNS Considerations	50
11.1. Relationship between failover and dynamic DNS update	50
11.2. Exchanging DDNS Information	51
11.3. Adding RRs to the DNS	53
11.4. Deleting RRs from the DNS	54
11.5. Name Assignment with No Update of DNS	54
12. Reservations and failover	55
13. Security Considerations	56
14. IANA Considerations	56
15. Acknowledgements	56
16. References	57
16.1. Normative References	57
16.2. Informative References	57
Authors' Addresses	58

1. Requirements Language

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

2. Glossary

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

- o Failover endpoint - The failover protocol allows for there to be a unique failover 'endpoint' for each failover relationship in which a failover server participates. The failover relationship is defined by a relationship name, and includes the failover partner IP address, the role this server takes with respect to that partner (primary or secondary), and the prefixes associated with that relationship. Note that a single prefix can only be associated with a single failover relationship. This failover endpoint can take actions and hold unique states. Typically, there is a one failover endpoint per partner (server), although there may be more. 'Server' and 'failover endpoint' are synonymous only if the server participates in only one failover relationship. However, for the sake of simplicity 'Server' is used throughout the document to refer to a failover endpoint unless to do so would be confusing.
- o Failover transmission - all messages exchanged between partners.
- o Independent Allocation - a prefix allocation algorithm to split the available pool of resources between the primary and secondary servers that is particularly well suited for vast pools (i.e. when available resources are not expected to deplete). See Section 6.2 for details.
- o Partner - name of the other DHCPv6 server that participates in failover relationship. When the role (primary or secondary) is not important, the other server is referred to as a "failover partner" or simply partner.
- o Primary Server - First out of two DHCPv6 servers that participate in a failover relationship. In active-passive mode this is the server that handles most of the client traffic. Its failover partner is referred to as secondary server.

- o Proportional Allocation - a prefix allocation algorithm that splits the available resources (addresses or prefixes) between the primary and secondary servers that is particularly well suited for more limited resources. See Section 6.1 for details.
- o Resource - Any type of resource that is assignable using DHCPv6. Currently there are two types of such resources defined: a non-temporary IPv6 address and an IPv6 prefix. Due to the nature of temporary addresses, they are not covered by the failover mechanism. Other resource types may be defined in the future.
- o Responsive - A server that is responsive, will respond to DHCPv6 client requests.
- o Secondary Server - Second of out two DHCPv6 servers that participate in a failover relationship. Its failover partner is referred to as primary server. In active-passive mode this server typically does not handle client traffic and acts as a backup.
- o Server - A DHCPv6 server that implements DHCPv6 failover. 'Server' and 'failover endpoint' are synonymous only if the server participates in only one failover relationship.
- o Unresponsive - A server that is unresponsive will not respond to DHCPv6 client requests.

3. Introduction

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

This protocol defines active-passive mode, sometimes also called a hot standby model. This means that during normal operation one server is active (i.e. actively responds to clients' requests) while the second is passive (i.e. it does receive clients' requests, but does not respond to them and only maintains a copy of lease database and is ready to take over incoming queries in case of primary server failure). Active-active mode (i.e. both servers actively handling

clients' requests) is currently not supported for the sake of simplicity. Such mode may be defined as an extension at a later time.

The failover protocol is designed to provide lease stability for leases with lease times beyond a short period. Due to the additional overhead required, failover is not suitable for leases shorter than 30 seconds. The DHCPv6 Failover protocol MUST NOT be used for leases shorter than 30 seconds.

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

3.1. Additional Requirements

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

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

3.2. Features out of Scope: Load Balancing

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

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

4. Protocol Overview

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

connection as typical DHCPv6 messages that convey DHCPv6 options, following format defined in Section 22.1 of [RFC3315].

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

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

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

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

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

The major problem with lazy update mechanism is the case when the server crashes after sending a response to client, but before sending

the lazy update to its partner (or when communication between partners is interrupted). To solve this problem, the concept known as the Maximum Client Lead Time (initially designed for DHCPv4 failover) is used. The MCLT is the maximum amount of time that one server can extend a lease for a client's binding beyond the time known by its failover partner. See Section 8.4 for detailed description how the MCLT affects assigned lease times.

Servers verify each others availability by periodically exchanging CONTACT messages. See Section 8.5 for discussion about detecting a partner's unreachability.

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

4.1. Failover State Machine Overview

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

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

A server starts its operation in short-lived STARTUP state. A server determines its partner reachability and state and sets its own state based on that determination. It frequently returns back to the state it was in before shutdown.

During typical operation when servers maintain communication, both are in NORMAL state. In that state only the primary responds to clients' requests. A secondary server is unresponsive to DHCPv6 clients.

If a server discovers that its partner is no longer reachable, it goes to COMMUNICATIONS-INTERRUPTED state. A server must be extra cautious as it can't distinguish if its partner is down or just communication between servers is interrupted. Since communication between partners is not possible, a server must act on the assumption that its partner is up. A failover server must follow a defined procedure, in particular, it MUST NOT extend any lease more than the MCLT beyond its partner's knowledge of the lease expiration time. This imposes an additional burden on the server, in that clients will

return to the server for lease renewals more frequently than they would otherwise. Therefore it is not recommended to operate for prolonged periods in this state. Once communication is reestablished, a server may go into NORMAL, POTENTIAL-CONFLICT or PARTNER-DOWN state. It may also stay in COMMUNICATIONS-INTERRUPTED state if certain conditions are met.

Once a server is switched into PARTNER-DOWN (when auto-partner-down is used or as a result of administrative action), it can extend leases, regardless of the original server that initially granted the lease. In that state server handles leases from its own pool, but is also able to serve pool from its downed partner. MCLT restrictions no longer apply. Operation in this mode is less demanding for the server that remains operational, than in COMMUNICATIONS-INTERRUPTED state, but PARTNER-DOWN does not offer any kind of redundancy.

When a server does not have an intact lease state database (e.g. due to first time run or catastrophic failure) or detects that its partner is in PARTNER-DOWN state and additional conditions are met, it switches to RECOVER state. In that state the server acknowledges that content of its database is doubtful and it needs to refresh its database from its partner. Once this operation is complete, it switches to RECOVER-WAIT and later to RECOVER-DONE.

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

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

A Server that is being shut down sends a DISCONNECT message. See Section 4.2.

4.2. Messages

The failover protocol is centered around the message exchanges used by one server to update its partner and respond to received updates. The following list enumerates these messages.

It should be noted that no specific formats or message type values are assigned at this stage. Appropriate implementation details will

be specified in a separate protocol specification document.

- o BNDUPD - The binding update message is used to send the binding lease changes to the partner. One message may contain one or more lease updates. The partner is expected to respond with a BNDACK message.
- o BNDACK - The binding acknowledgement is used for confirmation of the received BNDUPD message. It may contain a positive or negative response (e.g. due to detected lease conflict).
- o POOLREQ - The Pool Request message is used by one server (typically secondary) to request allocation of resources (addresses or prefixes) from its partner. The partner responds with POOLRSP.
- o POOLRSP - The Pool Response message is used by one server (typically primary) to respond to its partner's request for resources allocation. One POOLRSP message may contain more than one pool.
- o UPDREQ - The update request message is used by one server to request that its partner send all binding database changes that has not been sent and confirmed already. Requested partner is expected to respond with zero or more BNDUPD messages, followed by UPDDONE that signals end of updates.
- o UPDREQALL - The update request all is used by one server to request that all binding database information be sent in order to recover from a total loss of its binding database by the requesting server. Requested server responds with zero or more BNDUPD messages, followed by UPDDONE that signal end of updates.
- o UPDDONE - The update done message is used by the responding server to indicate that all requested updates have been sent by the responding server and acked by the requesting server.
- o CONNECT - The connect message is used by the primary server to establish a high level connection with the other server, and to transmit several important configuration data items between the servers. The partner is expected to confirm by responding with CONNECTACK message.
- o CONNECTACK - The connect acknowledgement message is used by the secondary server to respond to a CONNECT message from the primary server.

- o DISCONNECT - The disconnect message is used by either server when closing a connection and shutting down. No response is required for this message.
- o STATE - The state message is used by either server to inform its partner about a change of failover state. In some cases it may be used to also inform the partner about current state, e.g. after connection is established in COMMUNICATIONS-INTERRUPTED or PARTNER-DOWN states.
- o CONTACT - The contact message is used by either server to ensure that the other server continues to see the connection as operational. It MUST be transmitted periodically over every established connection if other message traffic is not flowing, and it MAY be sent at any time.

5. Connection Management

5.1. Creating Connections

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

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

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

When a connection attempt is received, the only information that the receiving server has is the IP address of the partner initiating a connection. If it has any relationships with the connecting server for which it is a secondary server, it should just await the CONNECT message to determine which relationship this connection is to serve.

If it has no secondary relationships with the connecting server, it SHOULD drop the connection.

To summarize -- a primary server MUST use a connection that it has initiated in order to send a CONNECT message. Every server that is a

secondary server in a relationship simply listens for connection attempts from the primary server.

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

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

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

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

5.2. Endpoint Identification

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

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

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

Any failover endpoint can take actions and hold unique states.

This document frequently describes the behavior of the protocol in terms of primary and secondary servers, not primary and secondary failover endpoints. However, it is important to remember that every 'server' described in this document is in reality a failover endpoint that resides in a particular process, and that several failover endpoints may reside in the same server process.

It is not the case that there is a unique failover endpoint for each prefix that participates in a failover relationship. On one server, there is (typically) one failover endpoint per partner, regardless of how many prefixes are managed by that combination of partner and role. Conversely, on a particular server, any given prefix will be associated with exactly one failover endpoint.

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

6. Resource Allocation

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

1. Proportional Allocation - This allocation algorithm is a direct application of the algorithm defined in [dhcpv4-failover] to DHCPv6. Available resources are split between the primary and secondary servers. Released resources are always returned to the primary server. Primary and secondary servers may initiate a rebalancing procedure when disparity between resources available to each server reaches a preconfigured threshold. Only resources that are not leased to any clients are "owned" by one of the servers. This algorithm is particularly well suited for scenarios where amount of available resources is limited, as may be the case with prefix delegation. See Section 6.1 for details.
2. Independent Allocation - This allocation algorithm assumes that available resources are split between primary and secondary servers as well. In this case, however, resources are assigned to a specific server for all time, regardless if they are available or currently used. This algorithm is much simpler than proportional allocation, because resource imbalance doesn't have to be checked and there is no rebalancing for independent allocation. This algorithm is particularly well suited for scenarios where there is an abundance of available resources which is typically the case for DHCPv6 address allocation. See

Section 6.2 for details.

6.1. Proportional Allocation

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

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

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

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

The initial allocation when the servers first integrate is triggered by the POOLREQ message from the secondary to the primary. This is followed by the POOLRESP message where the primary tells the secondary how many resources it allocated to the secondary. Then, the primary sends the allocated resources to the secondary via BNDUPD messages. The POOLREQ/POOLRESP message is a trigger to the primary to perform a scan of its database and to ensure that the secondary has enough resources (based on some configured ratio).

Servers frequently have several kinds of resources available on a

particular network segment. The failover protocol assumes that both primary and secondary servers are configured in such a way that each knows the type and number of resources on every network segment participating in the failover protocol. The primary server is responsible for allocating the secondary server the correct proportion of available resources of each kind, and the secondary server is responsible for being configured in such a way that it can tell the kind of every resource based solely on the IP or prefix address itself.

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

Available resources can be delegated back to the primary server in certain cases. BNDUPD will contain state FREE for leases that were previously in FREE_BACKUP state.

The POOLREQ/POOLRESP message exchange initiated by the secondary is valid at any time, and the primary server SHOULD, whenever it receives the POOLREQ message, scan its database of prefixes and determine if the secondary needs more resources from any of the prefixes.

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

The primary server SHOULD examine the balance of available resources between the primary and secondary for a particular prefix whenever the number of available resources for either the primary or secondary changes by more than a configured limit. The primary server SHOULD adjust the available resource balance as required to ensure the configured resource balance, excepting that the primary server SHOULD employ some threshold mechanism to such a balance adjustment in order to minimize the overhead of maintaining this balance.

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

The primary server can, at any time, send an available resource to the secondary using a BNDUPD with the state BACKUP. The primary server can attempt to take an available resource away from the secondary by sending a BNDUPD with the state FREE. If the secondary

accepts the BNDUPD, then it is now available to the PRIMARY and not available to the secondary. Of course, the secondary MUST reject that BNDUPD if it has already used that resource for a DHCP client.

6.2. Independent Allocation

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

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

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

6.3. Choosing Allocation Algorithm

All implementations MUST support proportional allocation algorithm and SHOULD support independent allocation. If the implementation implement both and let the user configure it, the default algorithm used SHOULD be proportional allocation algorithm.

Proportional allocation mechanism is more flexible as it can dynamically rebalance available resources between servers. That balance includes additional burden for the servers and generates more

traffic between servers. Proportional algorithm can be considered as managing available resources more efficiently than independent. That is important aspect when working in a network that is nearing address and/or prefix depletion.

Independent allocation can be used when the number of available resources are large and there is no realistic danger of running out of resources. Use of the independent allocation makes communication between partners simpler.

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

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

7. Information model

In most DHCP servers a resource (an IP address or a prefix) can take on several different binding-status values, sometimes also called lease states. While no two DHCP servers probably have exactly the same possible binding-status values, the DHCP RFC enforces some commonality among the general semantics of the binding-status values used by various DHCP server implementations.

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

The lease binding-status values defined for the failover protocol are listed below. Unless otherwise noted below, there MAY be client

information associated with each of these binding-status value.

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

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

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

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

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

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

ABANDONED -- indicates that a lease is considered unusable by the DHCP system. The primary reason for entering such state is reception of **DECLINE** message for said lease. Client identification **MUST NOT** appear.

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

The lease state machine has been presented in Figure 1. Most states are stationary, i.e. the lease stays in a given state until external event triggers transition to another state. The only transitive state is FREE*. Once it is reached, the the state machine immediately transitions to either FREE or FREE_BACKUP state.

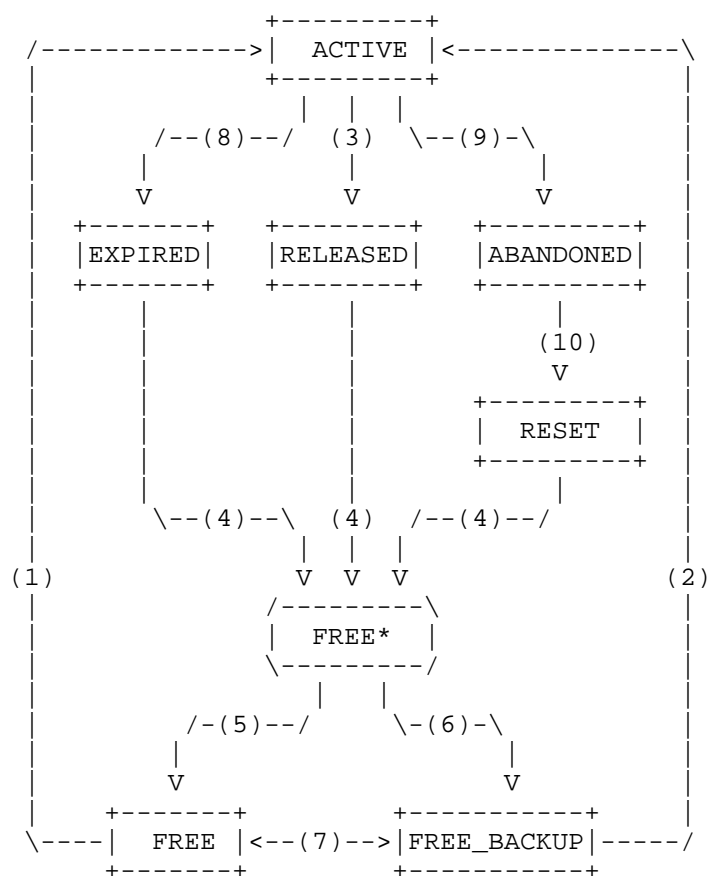


Figure 1: Lease State Machine

Transitions between states are results of the following events:

1. Primary server allocates a lease.
2. Secondary server allocates a lease.
3. Client sends RELEASE and the lease is released.

4. Partner acknowledges state change. This transition MAY also occur if the server is in PARTNER-DOWN state and the MCLT has passed since the entry in RELEASED, EXPIRED, or RESET states.
5. The lease belongs to a pool that is governed by the proportional allocation, or independent allocation is used and this lease belongs to primary server.
6. The lease belongs to a pool that is governed by the independent allocation and the lease belongs to the secondary server.
7. Pool rebalance event occurs (POOLREQ/POOLRSP messages are exchanged). Addresses (or prefixes) belonging to the primary server can be assigned to the secondary server pool (transition from FREE to FREE_BACKUP) or vice versa.
8. The lease is expired.
9. DECLINE message is received or a lease is deemed unusable for other reasons.
10. An administrative action is taken to recover an abandoned lease back to usable state. This transition MAY occur due to an implementation specific handling on ABANDONED resource. One possible example of such use is a Neighbor Discovery or ICMP Echo check if the address is still in use.

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

Pool owner		Primary	Secondary
Algorithm			
Proportional		FREE	FREE
Independent		FREE	FREE_BACKUP

Figure 2: FREE* State Transitions

In case of servers operating in active-passive mode, while a majority of the resources are owned by the primary server, the secondary server will need a portion of the resources to serve new clients

while operating in COMMUNICATION-INTERRUPTED state and also in PARTNER-DOWN state before it can take over the entire address pool (after the expiry of MCLT).

The secondary server cannot simply take over the entire resource pool immediately, since we have to handle the case where both servers are able to communicate with DHCP clients, but unable to communicate with each other.

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

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

8. Failover Mechanisms

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

8.1. Time Skew

Partners exchange information about known lease states. To reliably compare a known lease state with an update received from a partner, servers must be able to reliably compare the times stored in the known lease state with the times received in the update. Although a simple approach would be to require both partners to use synchronized time, e.g. by using NTP, such a service may not always be available in some scenarios that failover expects to cover. Therefore a mechanism to measure and track relative time differences between servers is necessary. To do so, each message **MUST** contain information about the time of the transmission in the time context of the transmitter. The transmitting server **MUST** set this as close to the actual transmission as possible. The receiving partner **MUST** store its own timestamp of reception as close to the actual reception

as possible. The received timestamp information is then compared with local timestamp.

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

8.2. Time expression

Timestamps are expressed as number of seconds since midnight (UTC), January 1, 2000, modulo 2^{32} . Note: that is the same approach as used in creation of DUID-LLT (see Section 9.2 of [RFC3315]).

Time differences are expressed in seconds and are signed.

8.3. Lazy updates

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

Although the lazy update mechanism does not introduce additional delays in server response times, it introduces other difficulties. The key problem with lazy update is that when a server fails after updating a client with a particular lease time and before updating its partner, the partner will believe that a lease has expired even though the client still retains a valid lease on that address or prefix.

8.4. MCLT concept

In order to handle problem introduced by lazy updates (see Section 8.3), a period of time known as the "Maximum Client Lead Time" (MCLT) is defined and must be known to both the primary and secondary servers. Proper use of this time interval places an upper bound on the difference allowed between the lease time provided to a DHCPv6 client by a server and the lease time known by that server's failover partner.

The MCLT is typically much less than the lease time that a server has

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

The fundamental relationship on which much of the correctness of this protocol depends is that the lease expiration time known to a DHCPv6 client **MUST NOT** under any circumstances be more than the maximum client lead time (MCLT) greater than the potential expiration time known to a server's partner.

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

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

The different times are:

desired valid lifetime:

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

actual valid lifetime:

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

potential valid lifetime:

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

acknowledged potential valid lifetime:

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

8.4.1. MCLT example

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

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

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

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

When the DHCPv6 client attempts to renew at T1 (approximately one half an hour from the start of the lease), the primary server again determines the desired valid lifetime, which is still 3 days. It then compares this with the original acknowledged potential valid lifetime (3 days + 1/2 hour) and adjusts for the time passed since the secondary was last updated (1/2 hour). Thus the time remaining of the acknowledged potential valid interval is 3 days. Adding the MCLT to this yields 3 days plus 1 hour, which is more than the

desired valid lifetime of 3 days. So the client is renewed for the desired valid lifetime -- 3 days.

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

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

8.5. Unreachability detection

Each partner maintains an FO_SEND timer for each partner connection. The FO_SEND timer is reset every time any message is transmitted. If the timer reaches the FO_SEND_MAX value, a CONTACT message is transmitted and timer is reset. The CONTACT message may be transmitted at any time.

8.6. Re-allocating Leases

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

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

the resource.

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

An "available" resource on a server may be allocated to any client. A resource which was leased to a client and which expired or was released by that client would take on a new state, EXPIRED or RELEASED respectively. The partner server would then be notified that this resource was EXPIRED or RELEASED through a BNDUPD. When the sending server received the BNDACK for that resource showing it was FREE, it would move the resource from EXPIRED or RELEASED to FREE, and it would be available for allocation by the primary server to any clients.

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

8.7. Sending Binding Update

This and the following section is written as though every BNDUPD message contains only a single binding update transaction in order to reduce the complexity of the discussion. Note that while a server MAY generate BNDUPD messages with multiple binding update transactions, every server MUST be able to process a BNDUPD message which contains multiple binding update transactions and generate the corresponding BNDACK messages with status for multiple binding update transactions.

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

1. resource type - non-temporary address or a prefix. Resource type can be indicated by the container that conveys the actual resource (e.g. an IA_NA option indicates non-temporary IPv6 address).
2. resource information - the actual address or prefix. That is conveyed using the appropriate option, e.g. an IAADDR for an address or an IAPREFIX for prefix.
3. valid life time requested by client

4. valid life time sent to client
5. IAID - Identity Association used by the client, while obtaining a given lease. (Note1: one client may use many IAIDs simulatenously. Note2: IAID for IA, TA and PD are orthogonal number spaces.)
6. Next Expected Client Transmission - time interval since Client Last Transmission Time, when a response from a client is expected.
7. potential valid life time - a lifetime that the server is willing to set if there were no MCLT/failover restrictions imposed.
8. preferred life time sent to client - the actual value sent back to the client
9. CLTT - Client Last Transaction Time, a timestamp of the last received transmission from a client
10. Client DUID

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

1. assigned FQDN name, defined in [RFC4704]
2. Options Requested by the client, i.e. content of the ORO
3. Remote-ID, defined in [RFC4649]
4. Relay-ID, defined in [RFC5460], section 5.4.1
5. Link-layer address
[I-D.ietf-dhc-dhcpv6-client-link-layer-addr-opt]
6. Any other options the updating partner deems useful.

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

8.8. Receiving Binding Update

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

There are three sorts of conflicts:

1. Two clients, one resource - This is the duplicate resource allocation conflict. There two different clients each allocated the same resource. See Section 8.9.
2. Two resources, one client conflict - This conflict exists when a client on one server is associated with a one resource, and on the other server with a different resource in the same or related subnet. This does not refer to the case where a single client has resources in multiple different subnets or administrative domains, but rather the case where on the same subnet the client has a lease on one IP address in one server and on a different IP address on the other server.
This conflict may or may not be a problem for a given DHCP server implementation and policy. If implementations and policies allow, both resources can be assigned to a given client. In the event that a DHCP server requires that a DHCP client have only one outstanding lease of a given type, the conflict **MUST** be resolved by accepting the lease which has the latest CLTT.
3. binding-status conflict - This is normal conflict, where one server is updating the other with newer information. See Section 8.9 for details of how to resolve these conflicts.

8.9. Conflict Resolution

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

When analyzing a BNDUPD message from a partner server, if there is insufficient information in the BNDUPD to process it, then reject the BNDUPD with reject-reason 3: "Missing binding information".

If the resource in the BNDUPD is not a resource associated with the failover endpoint which received the BNDUPD message, then reject it with reject-reason 1: "Illegal IP address (not part of any address pool)".

Every BNDUPD message SHOULD contain a client-last-transaction-time option, which MUST, if it appears, be the time that the server last interacted with the DHCP client. It MUST NOT be, for instance, the time that the lease on an IP address expired. If there has been no interaction with the DHCP client in question (or there is no DHCP client presently associated with this resource), then there will be no client-last-transaction-time option in the BNDUPD message.

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

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

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

Figure 3: Conflict Resolution

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

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

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

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

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

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

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

8.10. Acknowledging Reception

9. Endpoint States

9.1. State Machine Operation

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

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

- o Responsiveness -- the server is either responsive to DHCPv6 client requests or it is not.
- o Allocation Pool -- which pool of addresses (or prefixes) can be used for allocation on receipt of a SOLICIT message.

- o MCLT -- ensure that valid lifetimes are not beyond what the partner has acked plus the MCLT (or not).

A server will transition from one failover state to another based on the specific values held by the following state variables:

- o Current failover state.
- o Communications status (OK or not OK).
- o Partner's failover state (if known).

Several events can cause the transition from one failover state to another.

- o Change in communications status (OK or not OK).
- o Change in partner's failover state.
- o Receipt of particular messages.
- o Expiration of timers.

Whenever either of the last two of the above state variables changes state, the state machine is invoked, which may then trigger a change in the current failover state. Thus, whenever the communications status changes, the state machine processing is invoked. This may or may not result in a change in the current failover state.

Whenever a server transitions to a new failover state, the new state MUST be communicated to its failover partner in a STATE message if the communications status is OK. In addition, whenever a server makes a transition into a new state, it MUST record the new state, its current understanding of its partner's state, and the time at which it entered the new state in stable storage.

The following state transition diagram gives a condensed view of the state machine. If there is a difference between the words describing a particular state and the diagram below, the words should be considered authoritative.

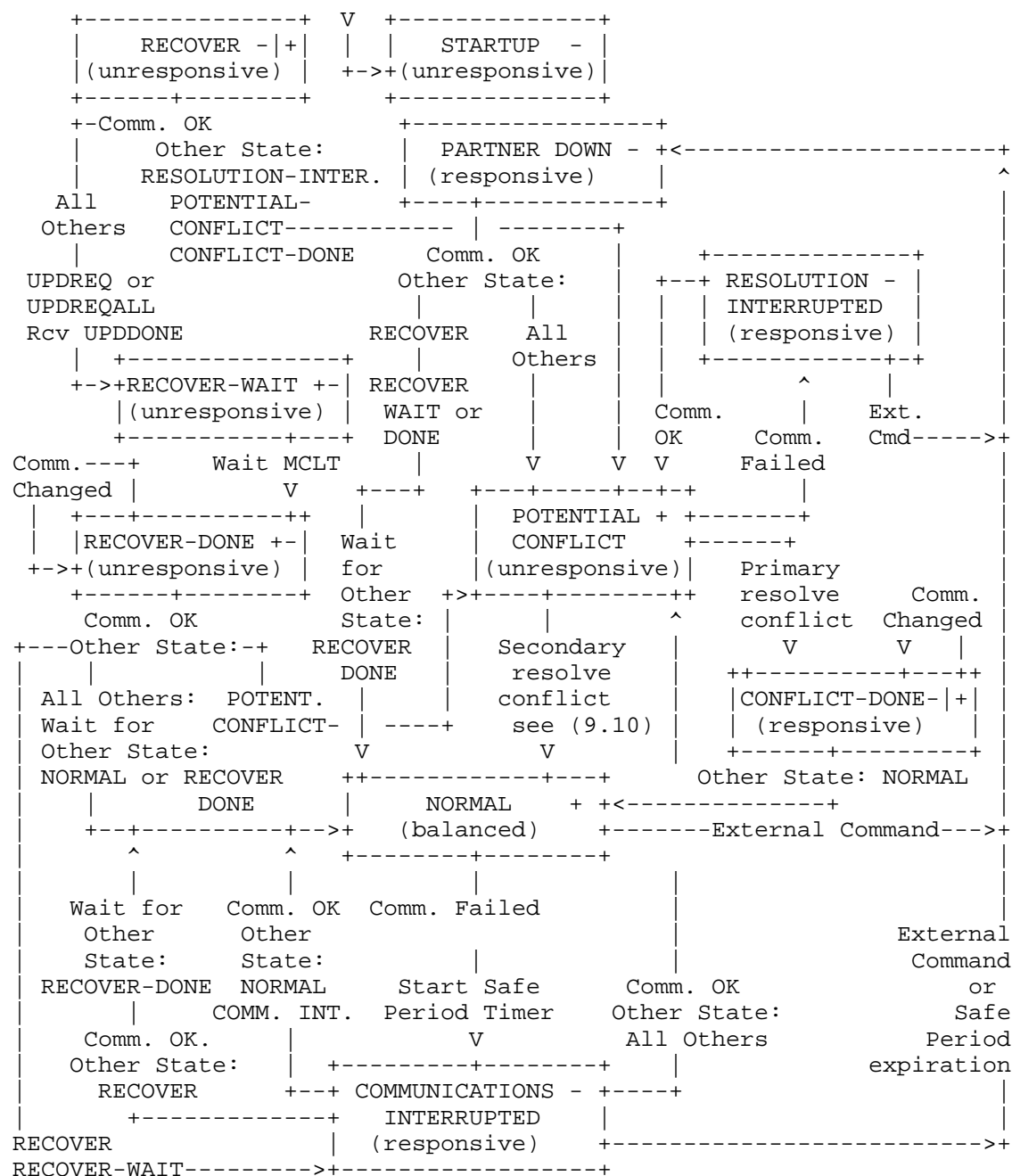


Figure 4: Failover Endpoint State Machine

9.2. State Machine Initialization

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

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

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

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

9.3. STARTUP State

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

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

obscure other information.

9.3.1. Operation in STARTUP State

The server **MUST NOT** be responsive in STARTUP state.

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

9.3.2. Transition Out of STARTUP State

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

Step 1:

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

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

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

Step 2:

If the previous state is one where communications was "OK", then set the previous state to the state that is the result of the communications failed state transition (if such transition exists -- some states don't have a communications failed state transition, since they allow both communications OK and failed).

Step 3:

Start the STARTUP state timer. The time that a server remains in the

STARTUP state (absent any communications with its partner) is implementation dependent but SHOULD be short. It SHOULD be long enough for a TCP connection to be created to a heavily loaded partner across a slow network.

Step 4:

Attempt to create a TCP connection to the failover partner.

Step 5:

Wait for "communications OK".

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

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

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

Step 6:

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

9.4. PARTNER-DOWN State

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

9.4.1. Operation in PARTNER-DOWN State

The server MUST be responsive in PARTNER-DOWN state.

It will allow renewal of all outstanding leases on IP addresses. For those IP addresses for which the server is using proportional allocation, it will allocate IP addresses from its own pool, and after a fixed period of time (the MCLT interval) has elapsed from

entry into PARTNER-DOWN state, it will allocate IP addresses from the set of all available IP addresses.

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

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

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

In the unlikely case, when there are two servers operating in a PARTNER-DOWN state, there is a chance of duplicate leases assigned. This leads to a POTENTIAL-CONFLICT (unresponsive) state when they re-establish contact. The duplicate lease issue can be postponed to a large extent by the server granting new leases first from its own pool. Therefore the server operating in PARTNER-DOWN state MUST use its own pool first for new leases before assigning any leases from its downed partner pool.

9.4.2. Transition Out of PARTNER-DOWN State

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

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

If the STARTUP bit is not set in the server-flags option of a STATE message received from its partner, then a server in PARTNER-DOWN state takes the following actions based on the state of the partner

as received in a STATE message (either immediately after establishing communications or at any time later when a new state is received)

If the partner is in:

NORMAL, COMMUNICATIONS-INTERRUPTED, PARTNER-DOWN, POTENTIAL-CONFLICT, RESOLUTION-INTERRUPTED, or CONFLICT-DONE state

transition to POTENTIAL-CONFLICT state

If the partner is in:

RECOVER, RECOVER-WAIT state

stay in PARTNER-DOWN state

If the partner is in:

RECOVER-DONE state

transition into NORMAL state

9.5. RECOVER State

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

9.5.1. Operation in RECOVER State

The server MUST NOT be responsive in RECOVER state.

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

9.5.2. Transition Out of RECOVER State

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

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

communicating with it, it MUST send an UPDREQALL message, otherwise it MUST send an UPDREQ message.

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

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

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

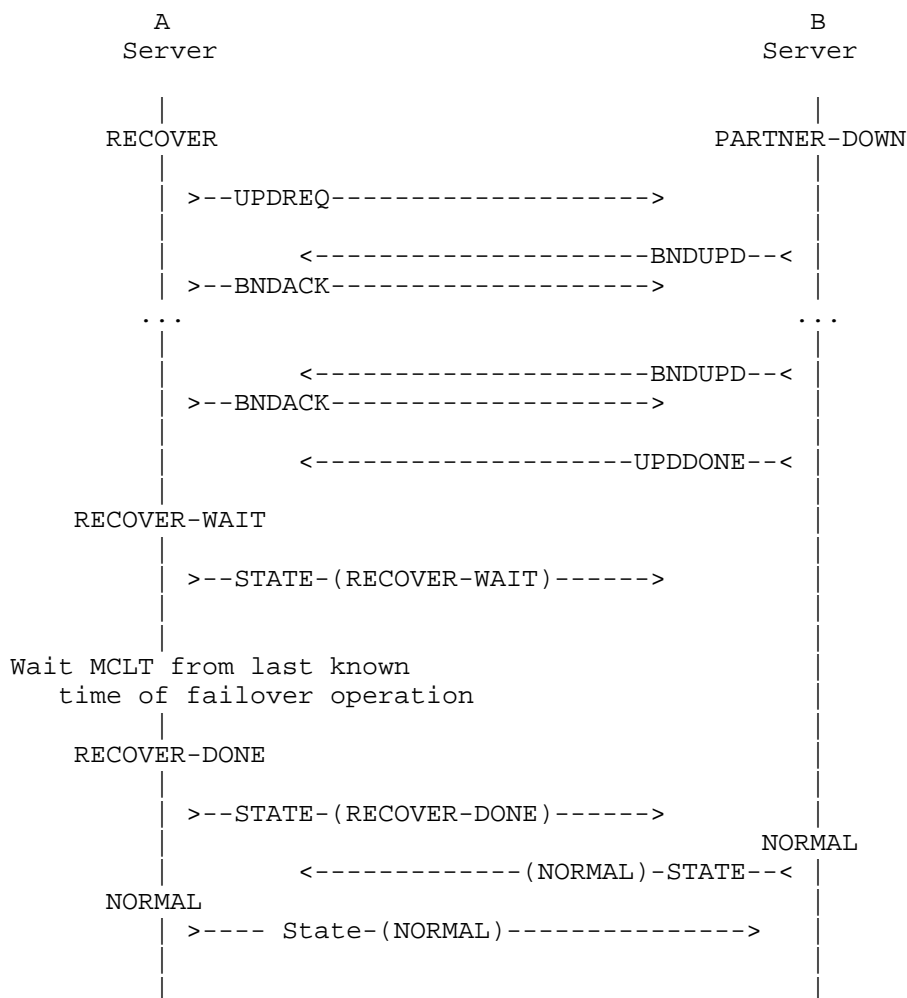


Figure 5: Transition out of RECOVER state

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

9.6. RECOVER-WAIT State

This state indicates that the server has done an UPDREQ or UPDREQALL and has received the UPDDONE message indicating that it has received all outstanding binding update information. In the RECOVER-WAIT state the server will wait for the MCLT in order to ensure that any

processing that this server might have done prior to losing its stable storage will not cause future difficulties.

9.6.1. Operation in RECOVER-WAIT State

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

9.6.2. Transition Out of RECOVER-WAIT State

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

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

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

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

If communications fails while a server is in RECOVER-WAIT state, it has no effect on the operation of this state. The server **SHOULD** continue to operate its timer, and the timer expires during the period where communications with the other server have failed, then the server **SHOULD** transition to RECOVER-DONE state. This is rare -- failover state transitions are not usually made while communications are interrupted, but in this case there is no reason to inhibit the timer.

9.7. RECOVER-DONE State

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

9.7.1. Operation in RECOVER-DONE State

A server in RECOVER-DONE state MUST respond only to DHCPREQUEST/RENEWAL and DHCPREQUEST/REBINDING DHCP messages.

9.7.2. Transition Out of RECOVER-DONE State

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

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

9.8. NORMAL State

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

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

When the above process is complete, if the server entering NORMAL state is a secondary server, then it will request IP addresses for allocation using the POOLREQ message.

9.8.1. Operation in NORMAL State

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

Lease time calculations

As discussed in Section 8.4, the lease interval given to a DHCP client can never be more than the MCLT greater than the most recently received potential- expiration-time from the failover partner or the current time, whichever is later.

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

Lazy update of partner server

After sending an REPLY that includes lease update to a client, the server servicing a DHCP client request attempts to update its partner with the new binding information. Server transmits both

desired valid lifetime and actual valid lifetime.

Reallocation of IP addresses between clients

Whenever a client binding is released or expires, a BNDUPD message must be sent to the partner, setting the binding state to RELEASED or EXPIRED. However, until a BNDACK is received for this message, the IP address cannot be allocated to another client. It cannot be allocated to the same client again if a BNDUPD was sent, otherwise it can. See Section 8.6.

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

9.8.2. Transition Out of NORMAL State

If an external command is received by a server in NORMAL state informing it that its partner is down, then transition into PARTNER-DOWN state. Generally, this would be an unusual situation, where some external agency knew the partner server was down. Using the command in this case would be appropriate if the polling interval and timeout were long.

If a server in NORMAL state fails to receive acks to messages sent to its partner for an implementation dependent period of time, it MAY move into COMMUNICATIONS-INTERRUPTED state. This situation might occur if the partner server was capable of maintaining the TCP connection between the server and also capable of sending a CONTACT message every tSend seconds, but was (for some reason) incapable of processing BNDUPD messages.

If the communications is determined to not be "ok" (as defined in Section 8.5), then transition into COMMUNICATIONS-INTERRUPTED state.

If a server in NORMAL state receives any messages from its partner where the partner has changed state from that expected by the server in NORMAL state, then the server should transition into COMMUNICATIONS-INTERRUPTED state and take the appropriate state transition from there. For example, it would be expected for the partner to transition from POTENTIAL-CONFLICT into NORMAL state, but not for the partner to transition from NORMAL into POTENTIAL-CONFLICT state.

If a server in NORMAL state receives a DISCONNECT message from its partner, the server should transition into COMMUNICATIONS-INTERRUPTED state.

9.9. COMMUNICATIONS-INTERRUPTED State

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

When a server enters COMMUNICATIONS-INTERRUPTED state, if it has been configured to support an automatic transition out of COMMUNICATIONS-INTERRUPTED state and into PARTNER-DOWN state (i.e., a "safe period" has been configured, see section 10), then a timer MUST be started for the length of the configured safe period.

A server transitioning into the COMMUNICATIONS-INTERRUPTED state from the NORMAL state SHOULD raise some alarm condition to alert administrative staff to a potential problem in the DHCP subsystem.

9.9.1. Operation in COMMUNICATIONS-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new leases, each server allocates from its own pool, where the primary MUST allocate only FREE resources (addresses or prefixes), and the secondary MUST allocate only FREE_BACKUP resources (addresses or prefixes). When responding to RENEW messages, each server will allow continued renewal of a DHCP client's current lease on an IP address or prefix irrespective of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the maximum client lead time (MCLT) beyond the latest of: 1) the potential valid lifetime already acknowledged by the other server, or 2) the actual valid lifetime sent to the DHCPv6 client, or 3) the potential valid lifetime received from the partner server.

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

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

9.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State

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

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

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

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

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

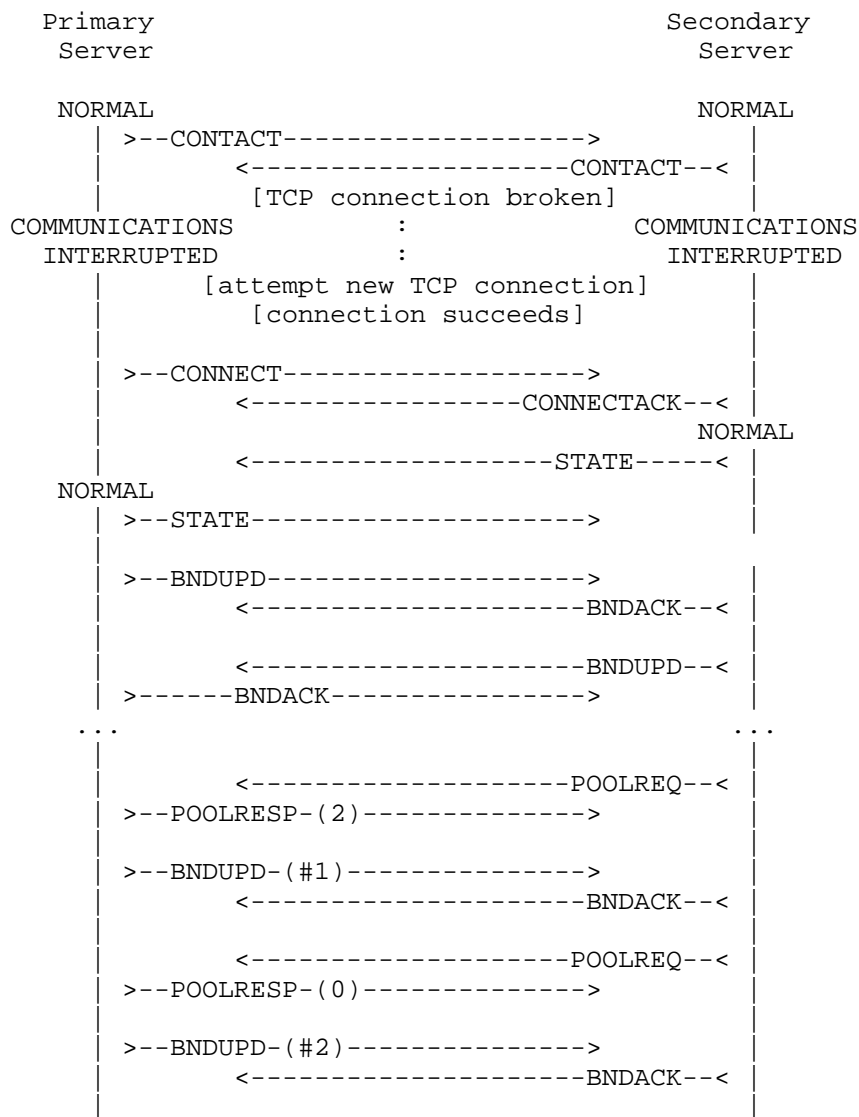


Figure 6: Transition from NORMAL to COMMUNICATIONS-INTERRUPTED and back (example with 2 addresses allocated to secondary)

9.10. POTENTIAL-CONFLICT State

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

the same resource has been offered and accepted by two different clients.

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

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

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

9.10.1. Operation in POTENTIAL-CONFLICT State

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

9.10.2. Transition Out of POTENTIAL-CONFLICT State

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

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

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

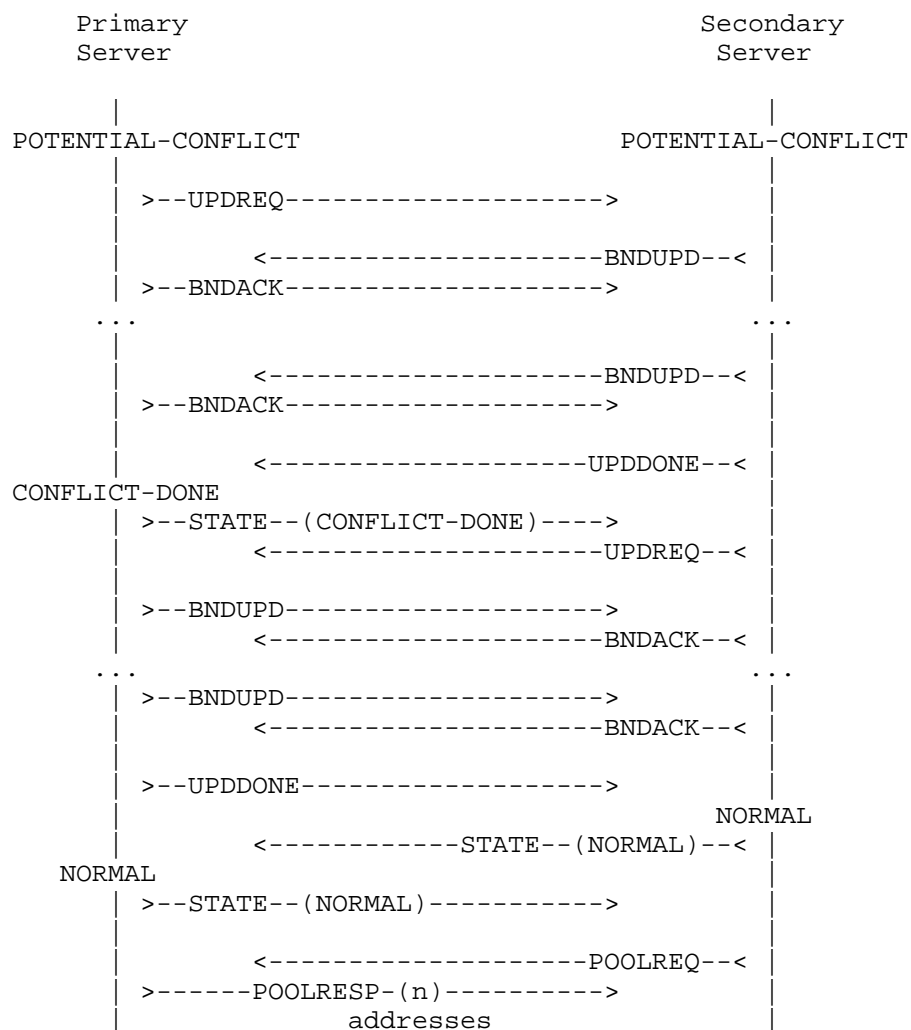


Figure 7: Transition out of POTENTIAL-CONFFLICT

9.11. RESOLUTION-INTERRUPTED State

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

If the servers remained in POTENTIAL-CONFFLICT while communications was interrupted, neither server would be responsive to DHCP client requests, and if one server had crashed, then there might be no server able to process DHCP requests.

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

9.11.1. Operation in RESOLUTION-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new resources (addresses or prefixes), each server SHOULD allocate from its own pool (if that can be determined), where the primary SHOULD allocate only FREE resources, and the secondary SHOULD allocate only BACKUP resources. When responding to renewal requests, each server will allow continued renewal of a DHCP client's current lease independent of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the maximum client lead time (MCLT) beyond the latest of: 1) the potential valid lifetime already acknowledged by the other server or 2) the lease-expiration-time or 3) potential valid lifetime received from the partner server.

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

9.11.2. Transition Out of RESOLUTION-INTERRUPTED State

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

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

9.12. CONFLICT-DONE State

This state indicates that during the process where the two servers are attempting to re-integrate with each other, the primary server has received all of the updates from the secondary server. It make a transition into CONFLICT-DONE state in order that it may be totally responsive to the client load, as opposed to NORMAL state where it would be in a "balanced" responsive state, running the load balancing algorithm.

9.12.1. Operation in CONFLICT-DONE State

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

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

9.12.2. Transition Out of CONFLICT-DONE State

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

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

10. Proposed extensions

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

10.1. Active-active mode

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

Discussion: Do DHCPv6 clients actually do this? DHCPv4 clients were rumored to wait for a "while" to accept the best offer, but to a first approximation, they all take the first offer they receive that is even acceptable.

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

Discussion: The possibility of setting both servers preference to an equal value could theoretically work as a crude attempt to provide load balancing. It wouldn't do much good on its own, as one (faster) server could be chosen more frequently (assuming that with equal preference sets clients will pick first responding server, which is not mandated by DHCPv6). We could design a simple mechanism of

dynamically updating preference depending on usage of available resources. This concept hasn't been investigated in detail yet.

11. Dynamic DNS Considerations

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

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

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

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

11.1. Relationship between failover and dynamic DNS update

The failover protocol describes the conditions under which each failover server may renew a lease to its current DHCP client, and describes the conditions under which it may grant a lease to a new

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

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

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

11.2. Exchanging DDNS Information

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

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

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

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

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

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

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

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

11.3. Adding RRs to the DNS

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

1. When the lease was granted before the server was configured to perform DDNS updates, the server MAY be configured to perform updates when it next renews existing leases. The server which granted the lease is the server which should initiate the DDNS update.
2. If a server is in PARTNER-DOWN state, it can conclude that its partner is no longer attempting to perform an update for the existing client. If the remaining server has not recorded that an update for the binding has been successfully completed, the

server MAY initiate a DDNS update. It MAY initiate this update immediately upon entry to PARTNER-DOWN state, it may perform this in the background, or it MAY initiate this update upon next hearing from the DHCP client.

11.4. Deleting RRs from the DNS

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

A server not in PARTNER-DOWN state "makes a resource FREE" when it initiates a BNDUPD with a binding-status of FREE, FREE_BACKUP, EXPIRED, or RELEASED. Its partner confirms this status by acking that BNDUPD, and upon receipt of the BNDACK the server has "made the resource FREE". Conversely, a server in PARTNER-DOWN state "makes a resource FREE" when it sets the binding-status to FREE, since in PARTNER-DOWN state no communications is required with the partner.

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

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

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

11.5. Name Assignment with No Update of DNS

In some cases, a DHCP server is configured to return a name to the DHCPv6 client but not enter that name into the DNS. This is typically a name that it has discovered or generated from information it has received from the client. In this case this name information SHOULD be communicated to the failover partner, if only to ensure

that they will return the same name in the event the partner becomes the server to which the DHCPv6 client begins to interact.

12. Reservations and failover

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

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

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

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

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

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

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

13. Security Considerations

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

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

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

TODO: Security considerations section contains loose notes and will be transformed into consistent text once the core design solidifies.

14. IANA Considerations

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

15. Acknowledgements

This document extensively uses concepts, definitions and other parts of [dhcpv4-failover] document. Authors would like to thank Shawn Rother, Greg Rabil, and Bernie Volz for their significant involvement and contributions. Authors would like to thank VithalPrasad Gaitonde for his insightful comments.

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

09-00 (Future Internet Engineering Project).

16. References

16.1. Normative References

- [I-D.ietf-dhc-dhcpv6-client-link-layer-addr-opt]
Halwasia, G., Systems, C., and W. Dec, "Client Link-layer Address Option in DHCPv6",
draft-ietf-dhc-dhcpv6-client-link-layer-addr-opt-03 (work in progress), October 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC4703] Stapp, M. and B. Volz, "Resolution of Fully Qualified Domain Name (FQDN) Conflicts among Dynamic Host Configuration Protocol (DHCP) Clients", RFC 4703, October 2006.
- [RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, October 2006.

16.2. Informative References

- [I-D.ietf-dhc-dhcpv6-failover-requirements]
Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Requirements",
draft-ietf-dhc-dhcpv6-failover-requirements-02 (work in progress), September 2012.
- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
- [RFC4649] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Remote-ID Option", RFC 4649, August 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.

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

Authors' Addresses

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

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

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

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

Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Informational
Expires: March 11, 2013

T. Mrugalski
ISC
K. Kinnear
Cisco
September 7, 2012

DHCPv6 Failover Requirements
draft-ietf-dhc-dhcpv6-failover-requirements-02

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on March 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Language	4
2. Introduction	4
3. Definitions	4
4. Scope of work	5
4.1. Alternatives to Failover	6
4.1.1. Short-lived addresses	6
4.1.2. Redundant servers	6
4.1.3. Distributed databases	7
4.1.4. Load Balancing	7
5. Failover Scenarios	7
5.1. Hot Standby Model	7
5.2. Geographically Distributed Failover	8
5.3. Load balancing	8
5.4. 1-to-1, m-to-1 and m-to-m models	8
5.5. Split prefixes	8
5.6. Long lived connections	9
5.7. Partial server communication loss	9
6. Principles of DHCPv6 Failover	9
6.1. Failure modes	9
6.1.1. Server Failure	9
6.1.2. Network partition	10
6.2. Synchronization mechanisms	11
6.2.1. Lockstep	11
6.2.2. Lazy updates	11
7. DHCPv4 and DHCPv6 Failover Comparison	12
8. DHCPv6 Failover Requirements	12
8.1. Features out of scope	14
9. Related work	14
9.1. DHCPv4 failover concepts	14
9.1.1. Goals of DHCPv4 Failover	14
9.1.2. Goals lead to Concepts	15
9.1.3. Use of the MCLT in practice	16
9.1.4. Network Partition Events	17
9.1.5. Conflict Resolution	18
9.1.6. Load Balancing	18
9.2. DHCPv6 Redundancy Considerations	18
10. Security Considerations	18
11. IANA Considerations	19
12. Acknowledgements	19
13. References	19
13.1. Normative References	19
13.2. Informative References	20
Authors' Addresses	20

1. Requirements Language

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

2. Introduction

The DHCPv6 protocol, defined in [RFC3315] allows for multiple servers to be operating on a single network, however it does not define how the servers can share the same address and prefix delegation pools and allow a client to seamlessly extend its existing leases when the original server is down. [RFC3315] provides for these capabilities, but does not document how the servers cooperate and communicate to provide this capability. Some sites are interested in running multiple servers in such a way as to provide redundancy in case of server failure. In order for this to work reliably, the cooperating primary and secondary servers must maintain a consistent database of the lease information.

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

3. Definitions

This section defines terms that are relevant to DHCPv6 failover.

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

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

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

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

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

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

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

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

Partner - A "partner", for the purpose of this document, refers to a failover server, typically the other failover server in a failover relationship.

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

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

4. Scope of work

In order to fit within the IETF process effectively and efficiently, the standardization effort for DHCPv6 failover is expected to proceed with the creation of documents of increasing specificity. It begins with this document specifying the requirements for DHCPv6 failover ("requirements document"). Later documents are expected to address the design of the DHCPv6 failover protocol ("design document"), and if sufficient interest exists, the protocol details required to

implement the DHCPv6 failover protocol itself ("protocol document"). The goal of this partitioning is, in part, to ease the validation, review, and approval of the DHCPv6 failover protocol by presenting it in comprehensible parts to the larger community.

Additional documents describing extensions may also be defined.

DHCPv6 Failover requirements are presented in Section 8.

4.1. Alternatives to Failover

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

4.1.1. Short-lived addresses

There are cases when IPv6 addresses are used only for a short time, but there is a need to have high degree of confidence that those addresses will be served. A notable example is a PXE scenario where hosts require an address during boot. Address and possibly other configuration parameters are used during boot process and are discarded afterwards. Any lack of available DHCPv6 service at this time may render the devices unbootable.

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

4.1.2. Redundant servers

In some cases the desire to deploy failover is motivated by high availability, i.e. to continue providing services despite server failure. If there are no additional requirements, that goal may be fulfilled with simply deploying two or more independent servers on the same link.

There are several well documented approaches how such deployment works. They are discussed in detail in [I-D.ietf-dhc-dhcpv6-redundancy-consider]. Each of those approaches is simpler to deploy and maintain than full failover.

4.1.3. Distributed databases

Some servers may allow their lease database to be stored in external databases. Another possible alternative to failover is to configure two servers to connect to the same distributed database.

Care should be taken to understand how inconsistencies are solved in such database backends and how such conflict resolutions affect DHCPv6 server operation.

It is also essential to use only databases that are really distributed and do not have single point of failure themselves. Otherwise the single point of failure is only moved to a different location (database rather than DHCPv6 server). Such a configuration does not improve redundancy, but significantly complicates deployment.

4.1.4. Load Balancing

Sometimes the desire to deploy more than one server is based on the assumption that they will share the client traffic. Administrators that are interested in such a capability are advised to deploy a load balancing mechanism, defined in [I-D.kostur-dhc-loadbv6].

5. Failover Scenarios

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

5.1. Hot Standby Model

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

5.2. Geographically Distributed Failover

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

5.3. Load balancing

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

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

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

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

5.5. Split prefixes

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

5.6. Long lived connections

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

5.7. Partial server communication loss

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

6. Principles of DHCPv6 Failover

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

6.1. Failure modes

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

6.1.1. Server Failure

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

When working under the assumption that this is the only type of

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

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

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

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

6.1.1.2. Network partition

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

If this failure mode is taken into consideration, a server can't assume that partner unreachability automatically means that its partner is down. They must consider the fact that the partner may continue operating and interacting with a subset of the clients. The only valid assumption is that partner also detected the network partition event and follows procedures specified for such a situation.

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

This failure mode is more likely (though still rare) in the situation where two servers are in physically distant locations with multiple

network elements between them. This is the case in geographically distributed failover (see Section 5.2).

6.2. Synchronization mechanisms

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

6.2.1. Lockstep

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

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

6.2.2. Lazy updates

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

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

7. DHCPv4 and DHCPv6 Failover Comparison

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

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

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

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

8. DHCPv6 Failover Requirements

This section summarizes the requirements for DHCPv6 failover.

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

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

1. The number of supported partners MUST be exactly two, i.e. there are at most two servers that are aware of a specific lease.
2. For each prefix or address pool, a server MUST NOT participate in more than one failover relationship.
3. The defined protocol MUST support the m-to-1 model (i.e. one server may form more than one relationship), but an implementation MAY choose to implement only the 1-to-1 model (i.e. everything from one server is backed on another).
4. One partner MUST be able to continue serving leases offered by the other partner. This property is also sometimes called 'lease stability'. The failure of either failover partner SHOULD pose minimal or no impact on client connectivity. In particular, it MUST NOT force the client to change addresses and/or change prefixes delegated to it. Long-lived connections MUST NOT be disturbed.
5. Prefix delegation MUST be supported.
6. Use of the failover protocol MUST NOT introduce significant performance impact on server response times. Therefore synchronization between partners MUST be done using some form of lazy updates (see Section 6.2.2).
7. The pair of failover servers MUST be able to recover from a server down failure (see Section 6.1.1).
8. The pair of failover servers MUST be able to recover from a network partition event (see Section 6.1.2).
9. The design MUST allow secure communication between the failover partners.
10. The definition of extensions to this core protocol SHOULD be allowed, when possible.

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

Despite the lack of standardization of DHCPv4 failover, the coexistence of DHCPv4 and DHCPv6 failover MAY be taken into consideration. In particular, certain features that are common for

both IPv4 and IPv6, like DNS Update mechanism SHOULD be taken into consideration.

8.1. Features out of scope

The following features are explicitly out of scope.

1. Load Balancing - a capability is considered an extension and MAY be defined in a separate document. It MUST NOT be part of the core protocol, but rather defined as an extension. The primary reason for this is the desire to limit core protocol complexity. Load Balancing is likely to be defined as an extension. See [I-D.kostur-dhc-loadbv6].
2. Configuration synchronization - two failover partners are expected to maintain the same configuration. Mismatched configuration between partners is a frequent problem in failover solutions. Unfortunately, that is an open-ended problem, since different servers have very different config data models.
3. m-to-m model (see section Section 5.4)
4. servers participating in multiple failover relationships for any given pool.

9. Related work

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

9.1. DHCPv4 failover concepts

9.1.1. Goals of DHCPv4 Failover

1. Provide a high availability DHCP service by leveraging the hooks built into DHCPv4 [RFC2131] and its usual implementation to support multiple servers able to respond to client requests. These hooks are:
 - (a) The broadcast of DHCPDISCOVER requests.
 - (b) The transition from unicast for DHCPREQUEST/RENEW to broadcast for DHCPREQUEST/REBIND.

- (c) The usual implementation of DHCPv4 relay agents to allow forwarding of DHCPv4 requests to multiple different DHCPv4 servers.
- 2. Produce a minimal impact on performance of the DHCPv4 server.
- 3. Prevent duplicate IP address allocation even in the event of a network partition.
- 4. Allow multiple failover relationships per server.
- 5. Standardize only the minimum necessary to provide a high availability DHCP service. In particular, avoid standardizing the interchange of configuration information.

9.1.2. Goals lead to Concepts

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

The steps in this chain of reasoning are as follows:

1. To avoid the major performance impact that a lockstep update of a failover partner would inflict, use a lazy update approach (see Section 6.2.2).
2. When using lazy update, there is always the problem that the failover server could crash after it has responded to some number of DHCPv4 clients and before it has updated its partner with the lease information it provided to those clients.
3. Thus, when one failover server cannot communicate with another failover server, it cannot know what that other failover server has told any of its DHCPv4 clients.

This creates an obvious problem.

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

This is done by ensuring that no DHCPv4 server participating in a failover relationship will ever offer a lease time to any DHCPv4

client that is more than an agreed-upon value beyond that known by its failover partner.

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

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

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

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

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

9.1.3. Use of the MCLT in practice

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

This tension is resolved by anticipating the need to extend lease times when communicating with the failover partner. The first lease

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

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

9.1.4. Network Partition Events

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

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

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

9.1.5. Conflict Resolution

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

9.1.6. Load Balancing

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

9.2. DHCPv6 Redundancy Considerations

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

10. Security Considerations

The design MUST allow secure communication between the failover partners. This requirement applies to the specification only, i.e. the design must include a way to secure communication. It does not mandate such security be employed, just that it be available.

The design of a security protection MUST take into consideration the fact that the protocol MUST operate despite network failures. In particular, relying on any external infrastructure (e.g. any form of Certificate Authority) is discouraged (or at least must be optional).

The security considerations for the design itself will be discussed in the design document.

11. IANA Considerations

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

12. Acknowledgements

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

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

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
- [RFC3074] Volz, B., Gonczi, S., Lemon, T., and R. Stevens, "DHC Load Balancing Algorithm", RFC 3074, February 2001.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, October 2006.

13.2. Informative References

- [I-D.ietf-dhc-dhcpv6-redundancy-consider]
Tremblay, J., Brzozowski, J., Chen, J., and T. Mrugalski,
"DHCPv6 Redundancy Deployment Considerations",
draft-ietf-dhc-dhcpv6-redundancy-consider-02 (work in
progress), October 2011.
- [I-D.kostur-dhc-loadbv6]
Kostur, A., "DHC Load Balancing Algorithm for DHCPv6",
draft-kostur-dhc-loadbv6-00 (work in progress),
August 2012.
- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound,
"Dynamic Updates in the Domain Name System (DNS UPDATE)",
RFC 2136, April 1997.
- [dhcpv4-failover]
Droms, R., Kinnear, K., Stapp, M., Volz, B., Gonczi, S.,
Rabil, G., Dooley, M., and A. Kapur, "DHCP Failover
Protocol", draft-ietf-dhc-failover-12 (work in progress),
March 2003.

Authors' Addresses

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

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

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 3, 2013

S. Jiang, Ed.
F. Xia
B. Sarikaya
Huawei Technologies
August 30, 2012

Prefix Assignment in DHCPv6
draft-ietf-dhc-host-gen-id-04

Abstract

This document introduces a generic host-oriented prefix assignment mechanism using DHCPv6. In this new address configuration procedure, the prefix is assigned from a DHCPv6 server to hosts through DHCPv6 message exchanging while the interface identifiers are independently generated by the hosts. It enables both integral address assignment and self-generated addresses in one single mechanism, DHCPv6. It also enables stateless address configuration without RA attendance.

Status of this Memo

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

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

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

This Internet-Draft will expire on March 3, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Applicability	4
4. Address Auto-configuration	5
5. DHCPv6 Operation	5
6. DHCPv6 IA_PA Option	7
6.1. Identity Association for Prefix Assignment Option	7
6.2. IA_PA Prefix Option	9
7. IANA consideration	9
8. Security Considerations	9
9. Acknowledgements	9
10. References	9
10.1. Normative References	9
10.2. Informative references	10
Authors' Addresses	11

1. Introduction

A host IPv6 address is combined by a prefix and an interface identifier. Currently, there are two mechanisms to configure a host IPv6 address. [RFC3315] describes the operation of address assignment by a DHCPv6 server. The operation assumes that the server is responsible for the assignment of an integral address which includes both prefix and interface identifier parts as described in [RFC4291]. In the Stateless Address Autoconfiguration (SLAAC, [RFC4862]) model, the interface Identifier is generated by the host itself while the prefix is configured through Router Advertisement message defined in [RFC4861].

However, in a DHCPv6-managed network, assigning 128-bit address is insufficient. Some hosts may want to use self-generated address, which are combined by prefixes obtained from network configuration and interface identifiers generated by hosts. The applicable user cases include CGA [RFC3972], modified EUI-64 interface identifier [EUI-64], temporary addresses for privacy [RFC4941] and etc.

In these scenarios, the address configuration procedure has to be splitted in two methods: integral address assignment through DHCPv6 and prefix announcement by RA advertisement. Some ISPs desire to manage address configuration using one set of protocol, rather than mixture of DHCPv6 and Neighbor Discovery.

There are also some network environments in that prefix announcement through RAs may not be the best choice. For example, hosts may connect through tunnels, either layer 2 tunnels or layer 3 tunnels.

While a RA is only able to announce prefix on a single link, DHCPv6 configuration can be used to manage multiple links by setup DHCPv6 relay.

Up to now, there is no mechanism for host-oriented prefix assignment in DHCPv6. [RFC3633] defines Prefix Delegation options providing a mechanism for automated delegation of IPv6 prefixes using the DHCPv6. This mechanism is intended for delegating a long-lived prefix from a delegating router to a requesting router. This mechanism "is not bound to the assignment of IP addresses or other configuration information to hosts" [RFC3633]. It delegates prefixes to a routable device for itself use only. It does not support the host-generated interface identifiers model, in which prefix(es) need to be propagated to hosts.

This document introduces a generic prefix assignment mechanism using DHCPv6. In this new address configuration procedure, the prefix is propagated from a DHCPv6 server to hosts through DHCPv6 message

exchanging while the interface identifiers are independently generated by the hosts. It enables both integral address assignment and self-generated addresses in one single mechanism, DHCPv6. Note, in many scenarios, Neighbor Discovery [RFC4861] is still needed for routing and reachability. In other scenarios, this mechanism enables stateless address configuration while RA absents.

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].

The terminology in this document is mainly based on the definitions in [RFC3315] and [RFC3633].

Prefix assignment: a DHCPv6 server propagates prefix information to hosts in unicast model.

3. Applicability

In point-to-point link model, DHCPv6 operation with host-generated interface identifier, described in this document, may be used. [RFC4968] provides different IPv6 link models that are suitable for 802.16 based networks and a point-to-point link model is recommended. Also, 3GPP and 3GPP2 have earlier adopted the point-to-point link model based on the recommendations in [RFC3314]. In this model, one prefix can only be assigned to one interface of a host (mobile station) and different hosts (mobile stations) can't share a prefix. The unique prefix can be used to identify the host. It is not necessary for a DHCPv6 server to generate an interface identifier for the host. The host may generate its interface identifier as described in [RFC4941]. An interface identifier could even be generated via random number generation.

[RFC3972] defines Cryptographically Generated Addresses (CGA), which is generated from a giving prefix and a public signature key. For security reasons, it is only proper to be generated the user, the host itself. It requests a prefix before the interface identifier can be computed.

Modified EUI-64 interface identifier [EUI-64] is also typically generated by hosts. [RFC4941] has defined temporary addresses for privacy purposes. The temporary addresses is also generated by hosts using random algorithm.

The DHCPv6 operations defined in this document supports abovementioned address methods, and the host-generated addresses that may defined in the future.

4. Address Auto-configuration

Router Advertisements in ND [RFC4861] allow routers to inform hosts how to perform Address Auto-configuration. For example, routers can specify whether hosts should use DHCPv6 and/or stateless address configuration. In Router Advertisement message, M and O bits are used for indication of address auto-configuration mode.

Whatever address auto-configuration mode a host uses, the following two parts are necessary for the host to formulate it's IPv6 address:

- o A prefix. "A bit string that consists of some number of initial bits of an address" [RFC4861]. The prefixes can be announced through Router Advertisement message. Prefix assignment from a DHCPv6 server is not currently support.
- o An interface identifier. "From address autoconfiguration's perspective, an interface identifier is a bit string of known length" [RFC4862]. Modified EUI-64 interface identifier [EUI-64] is a widely-used host generated interface identifier. It generates interface identifier from the host MAC address. The interface identifier of CGA [RFC3972] is generated by computing a preifx that will be used to form the CGA and a cryptographic hash of a public key of a host. The host is responsible for interface identifier generation.

In the ND-managed environment, RA is used to assign the prefix.

So far, there is no mechanism to support the scenario that prefixes are managed by a DHCPv6 server. This document targets to meet this gap. The DHCPv6 operation defined in this document enables the DHCPv6 server to assign a prefix, rather than a integral address, to the host, so that the host can obtain an IPv6 address by combining the prefix with its own generated interface identifier. It enables the auto address configuration through DHCPv6.

5. DHCPv6 Operation

Figure 1 shows the operation of separating prefix assignment and interface identifier generation in the DHCPv6.

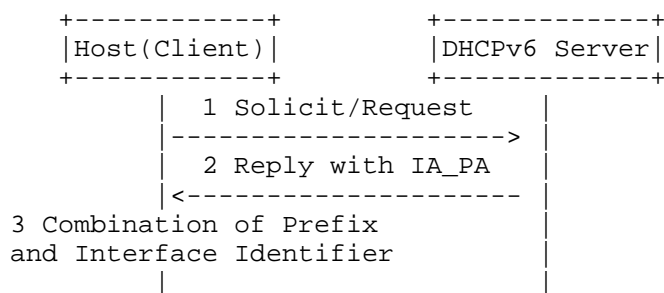


Figure 1: DHCPv6 Operation

1. A host uses a Solicit message to discover DHCPv6 servers. Indications of information requests can be included in the Solicit message or a Request message after discovery procedure. If a host that wants to use host generated addresses, it SHOULD request prefix assignment explicitly by including an IA_PA in a Solicit or a Request message, in which an IAID is provided by the host.
2. The DHCPv6 server assigns one or more prefixes to the host in the Reply messages responding to the prefix requests from the hosts. A server MUST return the same set of prefixes for the same IA_PA (as identified by the IAID) as long as those prefixes are still valid. After the lifetimes of the prefixes in an IA_TA have expired, the IAID may be reused to identify a new IA_PA with new prefix. If there is not a proper prefix available, a NoPrefixAvail (defined in [RFC3633]) status-code is returned to the host and the procedure is terminated.
3. The host generates an interface identifier and formulates a combined IPv6 address by concatenating the assigned prefix and the self-generated interface identifier.

After the host generates an IPv6 address using the above procedure, the host may send a Request message to the DHCPv6 server in order to confirm the usage of the new address. The confirmation procedure may be completed together with the address registration procedure [I-D.ietf-dhc-addr-registration]. However, the confirmation procedure is out of scope.

When the host reaches T1 or T2 defined in Section 6.1, it SHOULD use the same message exchanges, as described in section 18, "DHCP Client-Initiated Configuration Exchange" of [RFC3315], to obtain or update prefix(es) from a DHCPv6 server.

A DHCPv6 server MAY initiatively send a reconfiguration message to the host, as described in section 19, "DHCP Server-Initiated Configuration Exchange" of [RFC3315], to cause prefix(es) information

update.

If an IA_PA capable client connects to a network, and the DHCPv6 server is not IA_PA capable, the Solicit or Request message with IA_PA Option will result in no Reply, Reply without IA_PAs, or Reply with a Status Code containing UnspecFail. The client MAY decide the network does not support IA_PA immediately or after a period of soliciting (with limited retransmissions times). Then, it MAY "failover" to IA_NA/IA_TA requests.

6. DHCPv6 IA_PA Option

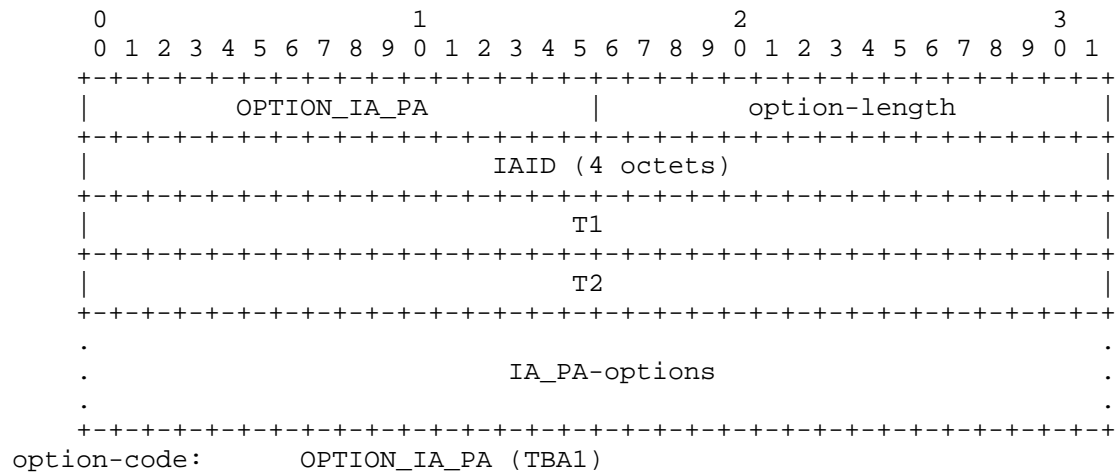
In this section, one new option is defined, Identity Association for Prefix Assignment Option . The format of this new DHCPv6 IA_PA Option has been deliberately designed to be the same with IA_PD option[RFC3633]. The IA_PD Prefix and IA Address sub-options from IA_PD option are also reused. However, the two options are different on the semantics and usage models.

Comparing with Prefix Information Option in ND, Section 4.6.2 of [RFC4861], the IA_PA option does not provide L flag and A flag. The A (autonomous address-configuration flag) isn't need obviously because the IA_PA is implicit for stateless address configuration. Because the IA_PA is only address relevant, it does not relevant to reachability or routing and the DHCPv6 server may not sure the on-link state. So L (on-Link) flag is not include. The DHCPv6 client should treat the prefix as same as L flag not set, which makes no statement about on-link or off-link properties of the prefix.

6.1. Identity Association for Prefix Assignment Option

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

The format of the IA_PA option is:



option-length: 12 + length of IA_PA-options field.

IAID: The unique identifier for this IA_PA; the IAID must be unique among the identifiers for all of this host's IA_PAs. The number space for IA_PA IAIDs is separate from the number spaces for IA_TA and IA_NA IAIDs

T1: The time at which the host should contact the DHCPv6 server from which the prefixes in the IA_PA were obtained to extend the lifetimes of the prefixes assigned to the IA_PA; T1 is a time duration relative to the current time expressed in units of seconds.

T2: The time at which the host should contact any available DHCPv6 server to extend the lifetimes of the prefixes assigned to the IA_PA; T2 is a time duration relative to the current time expressed in units of seconds.

IA_PA-options: Options associated with this IA_PA.

The details of the fields are similar to the IA_PD option description in [RFC3633]. The difference is here a DHCPv6 server and a host involved, while a delegating router and requesting router involved in [RFC3633].

6.2. IA_PA Prefix Option

OPTION_IAPREFIX (26) "IA_PD Prefix Option" defined in Section 10 of [RFC3633] is reused.

Originally, the option is used for conveying prefix information between a delegating router and a requesting router. Here the IA_PD Prefix option is used to specify IPv6 address prefixes associated with an IA_PA in Section 6.1. The IA_PD Prefix option must be encapsulated in the IA_PA-options field of an IA_PA option.

Note, the PD_EXCLUDE option [RFC6603] SHOULD NOT be encapsulated in the IAPREFIX options that are encapsulated in an IA_PA.

7. IANA consideration

This document defines a new DHCPv6 [RFC3315] option, which must be assigned Option Type values within the option numbering space for DHCPv6 messages:

The OPTION_IA_PA Option (TBA1), described in Section 6.1.

8. Security Considerations

Security considerations in DHCPv6 are described in [RFC3315].

To guard against attacks through prefix assignment, a host and a DHCPv6 server SHOULD use DHCPv6 authentication as described in Section 21, "Authentication of DHCP messages" of [RFC3315] or Secure DHCPv6 [I-D.ietf-dhc-secure-dhcpv6] .

9. Acknowledgements

The authors would like to thanks Suresh Krishnan, Ted Lemon, Bing Liu, Andre Kostur, Gaurav Halwasia, Bernie Volz and other members of DHC WG for their valuable comments.

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.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [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.
- [RFC6603] Korhonen, J., Savolainen, T., Krishnan, S., and O. Troan, "Prefix Exclude Option for DHCPv6-based Prefix Delegation", RFC 6603, May 2012.

10.2. Informative references

- [RFC3314] Wasserman, M., "Recommendations for IPv6 in Third Generation Partnership Project (3GPP) Standards", RFC 3314, September 2002.
- [RFC4968] Madanapalli, S., "Analysis of IPv6 Link Models for 802.16 Based Networks", RFC 4968, August 2007.
- [I-D.ietf-dhc-secure-dhcpv6]
Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs", draft-ietf-dhc-secure-dhcpv6-06 (work in progress), March 2012.
- [I-D.ietf-dhc-addr-registration]
Jiang, S. and G. Chen, "A Generic IPv6 Addresses Registration Solution Using DHCPv6", draft-ietf-dhc-addr-registration-00 (work in progress), May 2012.

- [EUI-64] "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>", March 1997.

Authors' Addresses

Sheng Jiang (editor)
Huawei Technologies
Q14, Huawei Campus, No.156, BeiQing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

Frank Xia
Huawei Technologies
1700 Alma Dr. Suite 500
Plano, TX 75075

Email: xiayangsong@huawei.com

Behcet Sarikaya
Huawei Technologies
1700 Alma Dr. Suite 500
Plano, TX 75075

Email: sarikaya@ieee.org

Dynamic Host Configuration Working
Group
Internet-Draft
Intended status: Informational
Expires: December 21, 2012

D. Hankins
Google
T. Mrugalski
M. Siodelski
ISC
S. Jiang
Huawei Technologies Co., Ltd
S. Krishnan
Ericsson
June 19, 2012

Guidelines for Creating New DHCPv6 Options
draft-ietf-dhc-option-guidelines-08

Abstract

This document provides guidance to prospective DHCPv6 Option developers to help them creating option formats that are easily adoptable by existing DHCPv6 software.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 21, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Language	3
2. Introduction	3
3. When to Use DHCPv6	4
4. General Principles	4
5. Reusing Other Options	5
5.1. Option with IPv6 addresses	5
5.2. Option with 32-bit integer value	6
5.3. Option with 16-bit integer value	7
5.4. Option with 8-bit integer value	7
5.5. Option with variable length data	7
5.6. Option with DNS Wire Format Domain Name List	8
6. Avoid Conditional Formatting	9
7. Avoid Aliasing	9
8. Suboptions in DHCPv6	10
9. Additional States Considered Harmful	11
10. Is DHCPv6 dynamic?	11
11. Multiple provisioning domains	11
12. Considerations for Creating New Formats	12
13. Option Size	12
14. Clients Request their Options	13
15. Security Considerations	13
16. IANA Considerations	15
17. References	15
17.1. Informative References	15
17.2. Informative References	16
Authors' Addresses	16

1. Requirements Language

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

2. Introduction

Most protocol developers ask themselves if a protocol will work, or work efficiently. These are important questions, but another less frequently considered question is whether the proposed protocol presents itself needless barriers to adoption by deployed software.

DHCPv6 [RFC3315] software implementors are not merely faced with the task of handling a given option's format on the wire. The option must fit into every stage of the system's process, starting with the user interface used to enter the configuration upto the machine interfaces where configuration is ultimately consumed.

Another frequently overlooked aspect of rapid adoption is whether the option requires operators to be intimately familiar with the option's internal format in order to use it? Most DHCPv6 software provides a facility for handling unknown options at the time of publication. The handling of such options usually needs to be manually configured by the operator. But if doing so requires extensive reading (more than can be covered in a simple FAQ for example), it inhibits adoption.

So although a given solution would work, and might even be space, time, or aesthetically optimal, a given option is presented with a series of ever-worsening challenges to be adopted;

- o If it doesn't fit neatly into existing config files.
- o If it requires new source code changes to be adopted, and hence upgrades of deployed software.
- o If it does not share its deployment fate in a general manner with other options, standing alone in requiring code changes or reworking configuration file syntaxes.

There are many things DHCPv6 option creators can do to avoid the pitfalls in this list entirely, or failing that, to make software implementors lives easier and improve its chances for widespread adoption.

3. When to Use DHCPv6

Principally, DHCPv6 carries configuration parameters for its clients. Any knob, dial, slider, or checkbox on the client system, such as "my domain name servers", "my hostname", or even "my shutdown temperature" are candidates for being configured by DHCPv6.

The presence of such a knob isn't enough, because DHCPv6 also presents the extension of an administrative domain - the operator of the network to which the client is currently attached. Someone runs not only the local switching network infrastructure that the client is directly (or wirelessly) attached to, but the various methods of accessing the external Internet via local assist services that network must also provide (such as domain name servers, or routers). This means that in addition to the existence of a configuration parameter, one must also ask themselves if it is reasonable for this parameter to be set by the directly attached network's administrators.

Note that the client still reserves the right to ignore values received via DHCPv6 (for example, due to having a value manually configured by its own operator). Bear in mind that doing so might cause the client to be rejected network attachment privileges, and this is one main reason for the use of DHCPv6 in corporate enterprises.

4. General Principles

The primary guiding principle to follow in order to enhance an option's adoptability is simplification. More specifically, the option should be created in such a way that does not require any new or special case software to support. If old software currently deployed and in the field can adopt the option through supplied configuration facilities then it's fairly certain that new software can easily formally adopt it.

There are at least two classes of DHCPv6 options: A bulk class of options which are provided explicitly to carry data from one side of the DHCPv6 exchange to the other (such as nameservers, domain names, or time servers), and a protocol class of options which require special processing on the part of the DHCPv6 software or are used during special processing (such as the Fully Qualified Domain Name (FQDN) option [RFC4704]), and so forth; these options carry data that is the result of a routine in some DHCPv6 software.

The guidelines laid out here should be applied in a relaxed manner for the protocol class of options. Wherever special case code is

already required to adopt the DHCPv6 option, it is substantially more reasonable to format the option in a less generic fashion, if there are measurable benefits to doing so.

5. Reusing Other Options

The easiest approach to manufacturing trivially deployable DHCPv6 Options is to assemble the option out of whatever common fragments fit - possibly allowing a group of fragments to repeat to fill the remaining space (if present) and so provide multiple values. Place all fixed size values at the start of the option, and any variable/indeterminate sized value at the tail end of the option.

This estimates that implementations will be able to reuse code paths designed to support the other options.

There is a tradeoff between the adoptability of previously defined option formats, and the advantages that new or specialized formats can provide. In general, it is usually preferable to reuse previously used option formats.

However, it isn't very practical to consider the bulk of DHCPv6 options already allocated, and consider which of those solve a similar problem. So, the following list of common option format fragments is provided as a shorthand. Please note that it is not complete in terms of exemplifying every option format ever devised...it is only a list of option format fragments which are used in two or more options.

5.1. Option with IPv6 addresses

This option format is used to carry one or many IPv6 addresses:

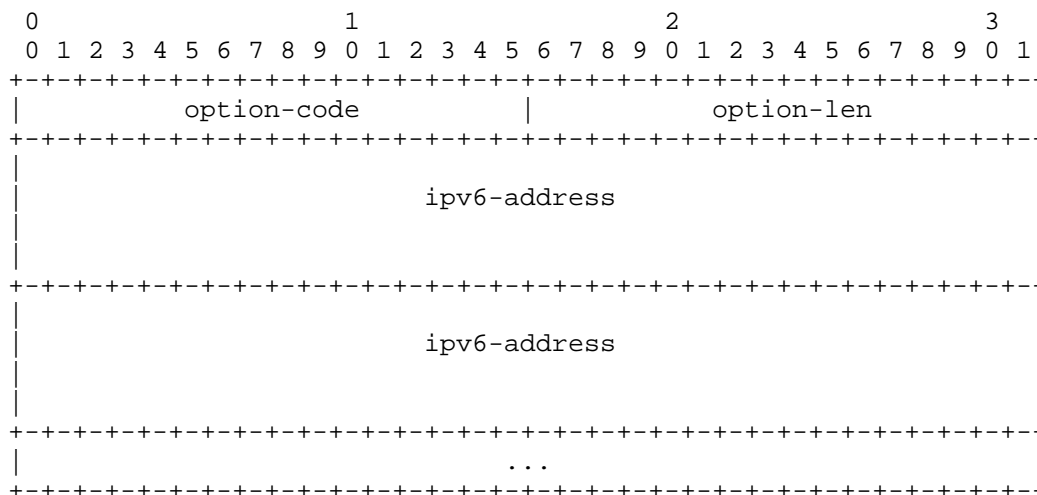


Figure 1: Option with IPv6 address

Examples of use:

- o DHCPv6 server unicast address [RFC3315]
- o SIP Servers IPv6 Address List [RFC3319]
- o DNS Recursive Name Server [RFC3646]
- o NIS Servers [RFC3898]
- o SNTP Servers [RFC4075]
- o Broadcast and Multicast Service Controller IPv6 Address Option for DHCPv6 [RFC4280]

5.2. Option with 32-bit integer value

This option format can be used to carry 32 bit-signed or unsigned integer value:

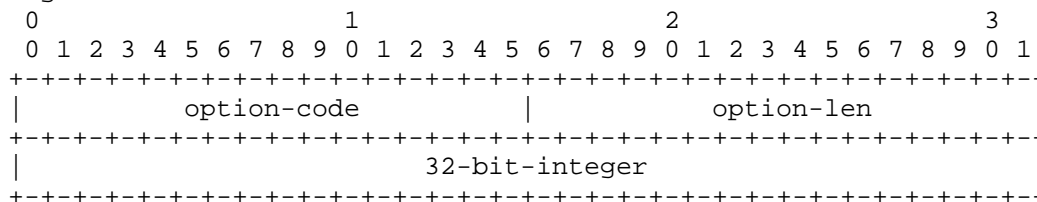


Figure 2: Option with 32-bit-integer value

Examples of use:

- o Information Refresh Time [RFC4242]

5.3. Option with 16-bit integer value

This option format can be used to carry 16-bit signed or unsigned integer values:

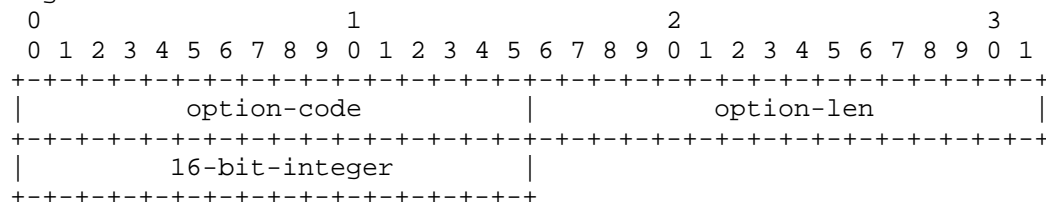


Figure 3: Option with 16-bit integer value

Examples of use:

- o Elapsed Time [RFC3315]

5.4. Option with 8-bit integer value

This option format can be used to carry 8-bit integer values:

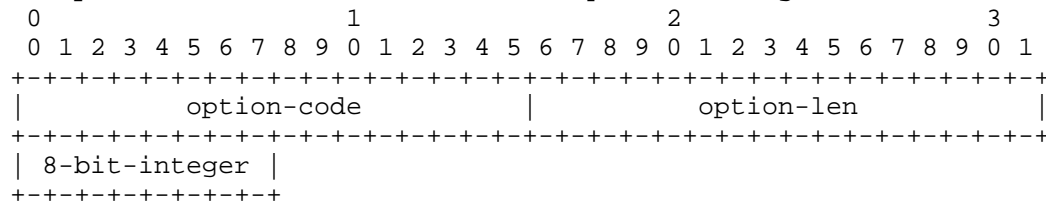


Figure 4: Option with 8-bit integer value

Examples of use:

- o DHCPv6 Preference [RFC3315]

5.5. Option with variable length data

This option can be used to carry variable length data of any kind. Internal representation of carried data is option specific. Some of the existing DHCPv6 options use NVT-ASCII strings to encode: filenames, host or domain names, protocol features or textual messages such as verbose error indicators.

This option format provides a lot of flexibility to pass data of

almost any kind. Though, whenever possible it is highly recommended to use more specialized options, with field types better matching carried data types.

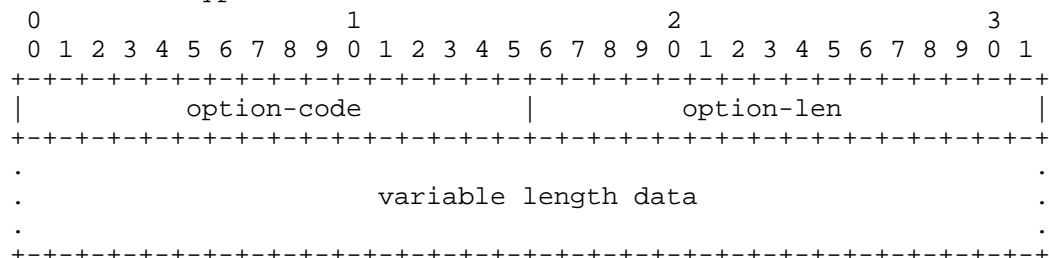


Figure 5: Option with variable length data

Examples of use:

- o Client Identifier [RFC3315]
- o Server Identifier [RFC3315]
- o Boot File URL [RFC5970]

5.6. Option with DNS Wire Format Domain Name List

This option is used to carry 'domain search' lists or any host or domain name:

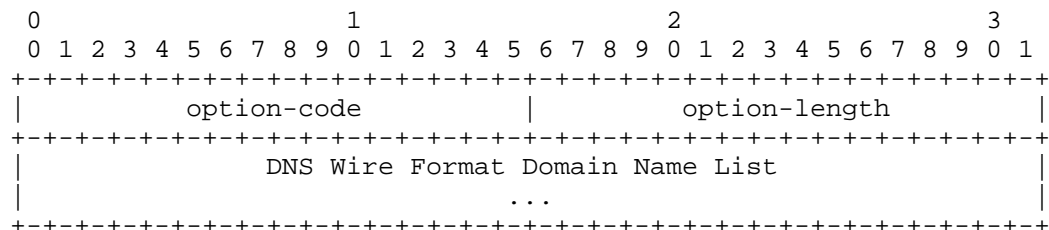


Figure 6: Option with DNS Wire Format Domain Name List

Examples of use:

- o SIP Servers Domain Name List [RFC3319]
- o NIS Domain Name [RFC3898]

6. Avoid Conditional Formatting

Placing an octet at the start of the option which informs the software how to process the remaining octets of the option may appear simple to the casual observer. But the only conditional formatting methods that are in widespread use today are 'protocol' class options. So conditional formatting requires new code to be written, as well as introduces an implementation problem; as it requires that all speakers implement all current and future conditional formats.

Conditional formatting is not recommended, except in cases where the DHCPv6 option has already been deployed experimentally, and all but one conditional format is deprecated.

7. Avoid Aliasing

Options are said to be aliases of each other if they provide input to the same configuration parameter. A commonly proposed example is to configure the location of some new service ("my foo server") using a binary IP address, a domain name field, and a URL. This kind of aliasing is undesirable, and is not recommended.

In this case, where three different formats are supposed, it more than triples the work of the software involved, requiring support for not merely one format, but support to produce and digest all three. Furthermore, code development and testing must cover all possible combinations of defined formats. Since clients cannot predict what values the server will provide, they must request all formats... so in the case where the server is configured with all formats, DHCPv6 option space is wasted on option contents that are redundant.

It also becomes unclear which types of values are mandatory, and how configuring some of the options may influence the others. For example, if an operator configures the URL only, should the server synthesize a domain name and IP address?

A single configuration value on a host is probably presented to the operator (or other software on the machine) in a single field or channel. If that channel has a natural format, then any alternative formats merely make more work for intervening software in providing conversions.

So the best advice is to choose the one method that best fulfills the requirements, be that for simplicity (such as with an IP address and port pair), late binding (such as with DNS), or completeness (such as with a URL).

On the specific subject of desiring to configure a value using a FQDN instead of a binary IP address, note that most DHCPv6 server implementations will happily accept a Domain Name entered by the administrator, and use DNS resolution to render binary IP addresses in DHCPv6 replies to clients. Consequently, consider the extra packet overhead incurred on the client's end to perform DNS resolution itself. The client may be operating on a battery and packet transmission is a non-trivial use of power, and the extra RTT delays the client must endure before the service is configured are at least two factors to consider in making a decision on format.

8. Suboptions in DHCPv6

Most options are conveyed in a DHCPv6 message directly. Although there is no codified normative language for such options, they are often referred to as top-level options. Many options may include other options. Such inner options are often referred to as sub-options. It should be noted that, contrary to DHCPv4, there is no shortage of option numbers. Therefore all options share a common option space. For example option type 1 meant different things in DHCPv4, depending if it was located in top-level or inside of Relay Agent Information option. There is no such ambiguity in DHCPv6.

Such encapsulation mechanism is not limited to one level. There is at least one defined option that is encapsulated twice: Identity Association for Prefix Delegation (IA_PD, defined in [RFC3633], section 9) conveys IA Prefix (IAPREFIX, defined in [RFC3633], section 10). Such delegated prefix may contain an excluded prefix range that is represented by PD_EXCLUDE option that is conveyed as sub-option inside IAPREFIX (PD_EXCLUDE, defined in [RFC6603]). It seems awkward to refer to such options as sub-sub-option, therefore "sub-option" term is typically used, regardless of the nesting level.

When defining configuration means for more complex mechanisms, it may be tempting to simply use sub-options. That should usually be avoided, as it increases complexity of the parser. It is much easier, faster and less error prone to parse larger number of options on a single (top-level) scope, than parse options on several scopes. The use of sub-options should be avoided as much as possible but it is better to use sub-options rather than conditional formatting.

It should be noted that currently there is no clear way defined for requesting sub-options. Most known implementations are simply using top-level ORO for requesting both top-level options and sub-options.

9. Additional States Considered Harmful

DHCP is a protocol designed for provisioning nodes. Less experienced protocol designers often assume that it is easy to define an option that will convey a different parameter for each node in a network. Such problems arose during designs of MAP [I-D.mdt-software-map-dhcp-option] and 4rd [I-D.ietf-software-4rd]. While it would be easier for provisioned nodes to get ready to use per node option values, such requirement puts exceedingly large loads on the server side. Alternatives should be considered, if possible. As an example, [I-D.mdt-software-map-dhcp-option] was designed in a way that all nodes are provisioned with the same set of MAP options and each provisioned node uses its unique address and delegated prefix to generate node-specific information. Such solution does not introduce any additional state for the server and therefore scales better.

It also should be noted that contrary to DHCPv4, DHCPv6 keeps several timers for renewals. Each IA_NA (addresses) and IA_PD (prefixes) contains T1 and T2 timers that designate time after which client will initiate renewal. Those timers apply only to its own IA containers. For renewing other parameters, please use Information Refresh Time Option (defined in [RFC4242]). Introducing additional timers make deployment unnecessarily complex. Please try to avoid it.

10. Is DHCPv6 dynamic?

DHCPv6 stands for Dynamic Host Configuration Protocol for IPv6. Contrary to its name, in many contexts it is not dynamic. While designing DHCPv6 options, it is worth noting that there is no reliable way to instantly notify clients that something has happened, e.g. parameter value has changed. There is a RECONFIGURE mechanism, but it has several serious drawbacks that makes its use difficult. First, its support is optional and many client implementations do not support it. To use reconfigure mechanism, server must use its secret nonce. That means that provisioning server is the only one that can initiate reconfiguration. Other servers do not know it and cannot trigger reconfiguration. Therefore the only reliable way for clients to refresh their configuration is to wait till T1 expires.

11. Multiple provisioning domains

In some cases there could be more than one DHCPv6 server on a link, with each provisioning a different set of parameters. One notable example of such case is a home network with a connection to two independent ISPs.

DHCPv6 was not initially designed with multiple provisioning domains. Although [RFC3315] states that a client that receives more than one ADVERTISE message, may respond to one or more, such capability was never observed in any known implementations. Existing clients will pick one server and will continue configuration process with that server, ignoring all other servers.

This is a generic DHCP protocol issue and should not be dealt within each option separately. This issue is better dealt with using a protocol-level solution and fixing this problem should not be attempted on a per option basis.

12. Considerations for Creating New Formats

If the option simply will not fit into any existing work by using fragments, the last recourse is to create a new format to fit.

When doing so, it is not enough to gauge whether or not the option format will work in the context of the option presently being considered. It is equally important to consider if the new format's fragments might reasonably have any other uses, and if so, to create the option with the foreknowledge that its parts may later become a common fragment.

One specific consideration to evaluate is whether or not options of a similar format would need to have multiple or single values encoded (whatever differs from the current option), and how that might be accomplished in a similar format.

13. Option Size

DHCPv6 [RFC3315] allows for packet sizes up to 64KB. First, through its use of link-local addresses, it steps aside many of the deployment problems that plague DHCPv4, and is actually an UDP over IPv6 based protocol (compared to DHCPv4, which is mostly UDP over IPv4 protocol, but with layer 2 hacks). Second, RFC 3315 explicitly refers readers to RFC 2460 Section 5, which describes an MTU of 1280 octets and a minimum fragment reassembly of 1500 octets. It's feasible to suggest that DHCPv6 is capable of having larger options deployed over it, and at least no common upper limit is yet known to have been encoded by its implementors. It is impossible to describe any fixed limit that cleanly divides those too big from the workable.

It is advantageous to prefer option formats which contain the desired information in the smallest form factor that satisfies the requirements.

DHCPv6 does allow for multiple instances of a given option, and they are treated as distinct values following the defined format, however this feature is generally preferred to be restricted to protocol class features (such as the IA_* series of options). In such cases, it is better to define an option as an array if it is possible. It is recommended to clarify (with normative language) whether a given DHCPv6 option may appear once or multiple times.

14. Clients Request their Options

The DHCPv6 Option Request Option (OPTION_ORO) [RFC3315], is an option that serves two purposes - to inform the server what options the client supports and is willing to consume.

It doesn't make sense for some options to appear on this Option Request Option, such as those formed by elements of the protocol's internal workings, or are formed on either end by DHCPv6-level software engaged in some exchange of information. When in doubt, it is prudent to assume that any new option must be present on the relevant option request list if the client desires to receive it.

It is a frequent mistake of option draft authors, then, to create text that implies that a server will simply provide the new option, and clients will digest it. Generally, it's best to also specify that clients **MUST** place the new option code on the relevant list option, clients **MAY** include the new option in their packets to servers with hints as to values they desire, and servers **MAY** respond with the option contents (if they have been so configured).

Creators of DHCPv6 options **MUST NOT** require special ordering of options either in the relevant request option, or in the order of options within the packet. Although it is reasonable to expect that options will be processed in the order they appear in ORO, server software is not required to sort DHCPv6 options into the same order in reply messages. It should be noted that any requirement regarding option ordering will break down most existing implementations, as "order is not important" was one of the design principles of DHCPv6 and many implementations follow it. For example, there are existing implementations that use hash maps for storing options, so forcing any particular order is not feasible without great deal of work. If options must be processed in any specific order (e.g. due to inter-dependency), use of option encapsulation should be considered.

15. Security Considerations

DHCPv6 does have an Authentication mechanism ([RFC3315]) that makes

it possible for DHCPv6 software to discriminate between authentic endpoints and men in the middle. Other authentication mechanisms may optionally be deployed. For example, the Secure DHCPv6 [I-D.ietf-dhc-secure-dhcpv6], based on Cryptographically Generated Addresses (CGA) [RFC3972], can provide source address ownership validation, message origin authentication and message integrity without requiring symmetric key pairs or supporting from any key management system. However, as of now, the mechanism is not widely deployed. It also does not provide end-to-end encryption.

So, while creating a new option, it is prudent to assume that the DHCPv6 packet contents are always transmitted in the clear, and actual production use of the software will probably be vulnerable at least to man-in-the-middle attacks from within the network, even where the network itself is protected from external attacks by firewalls. In particular, some DHCPv6 message exchanges are transmitted to multicast addresses that are likely broadcast anyway.

If an option is of a specific fixed length, it is useful to remind the implementer of the option data's full length. This is easily done by declaring the specific value of the 'length' tag of the option. This helps to gently remind implementers to validate option length before digesting them into likewise fixed length regions of memory or stack.

If an option may be of variable size (such as having indeterminate length fields, such as domain names or text strings), it is advisable to explicitly remind the implementor to be aware of the potential for long options. Either define a reasonable upper limit (and suggest validating it), or explicitly remind the implementor that an option may be exceptionally long (to be prepared to handle errors rather than truncate values).

For some option contents, out of bound values may be used to breach security. An IP address field might be made to carry a loopback address, or local broadcast address, and depending on the protocol this may lead to undesirable results. A domain name field may be filled with contrived contents that exceed the limitations placed upon domain name formatting... as this value is possibly delivered to "internal configuration" records of the system, it may be implicitly trusted without being validated.

So it behooves an option's definition to contain any validation measures as can reasonably be made.

16. IANA Considerations

This document has no actions for IANA.

17. References

17.1. Informative References

- [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.
- [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.
- [RFC3319] Schulzrinne, H. and B. Volz, "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers", RFC 3319, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.
- [RFC3898] Kalusivalingam, V., "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3898, October 2004.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4075] Kalusivalingam, V., "Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6", RFC 4075, May 2005.
- [RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, November 2005.
- [RFC4280] Chowdhury, K., Yegani, P., and L. Madour, "Dynamic Host Configuration Protocol (DHCP) Options for Broadcast and Multicast Control Servers", RFC 4280, November 2005.

- [RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, October 2006.
- [RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6 Options for Network Boot", RFC 5970, September 2010.
- [RFC6603] Korhonen, J., Savolainen, T., Krishnan, S., and O. Troan, "Prefix Exclude Option for DHCPv6-based Prefix Delegation", RFC 6603, May 2012.

17.2. Informative References

- [I-D.ietf-dhc-secure-dhcpv6]
Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs",
draft-ietf-dhc-secure-dhcpv6-06 (work in progress),
March 2012.
- [I-D.ietf-softwire-4rd]
Despres, R., Penno, R., Lee, Y., Chen, G., and S. Jiang,
"IPv4 Residual Deployment via IPv6 - a unified Stateless
Solution (4rd)", draft-ietf-softwire-4rd-00 (work in
progress), May 2012.
- [I-D.mdt-softwire-map-dhcp-option]
Mrugalski, T., Boucadair, M., Deng, X., Troan, O., and C.
Bao, "DHCPv6 Options for Mapping of Address and Port",
draft-mdt-softwire-map-dhcp-option-02 (work in progress),
January 2012.

Authors' Addresses

David W. Hankins
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Email: dhankins@google.com

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

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

Marcin Siodelski

Phone:
Email: msiodelski@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

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

Email: suresh.krishnan@ericsson.com

dhc
Internet-Draft
Intended status: Standards Track
Expires: March 15, 2013

G. Liu
Y. Tu
C. Zhu
ZTE Corporation
W. Hendericks
D. Derksen
L. Thiebaut
Alcatel-Lucent
September 11, 2012

DHCP Options for 3GPP Service
draft-liu-dhc-3gpp-option-02.txt

Abstract

This document defines a new option that can be used by DHCP clients in their signaling sent to DHCP servers, when those clients need to associate a request for an IP address or IPv6 prefix with a given 3GPP packet service. It is intended for scenarios of interworking between a non-3GPP access and a 3GPP network.

Status of this Memo

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

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

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

This Internet-Draft will expire on March 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Convention & Terminology	4
3. Problem Statement	5
4. 3GPP-Service Option Format	6
4.1. APN Sub-option Format	7
4.2. Non Seamless WLAN Offload (3GPP-Service-Type)Sub-option Format	7
5. DHCP Client Behavior	10
6. DHCP Server Behavior	11
7. Security Considerations	12
8. IANA Considerations	13
9. Normative References	14
Authors' Addresses	15

1. Introduction

The Dynamic Host Configuration Protocol (DHCP) is built on a client-server model, where designated DHCP server allocate network addresses and deliver configuration parameters to dynamically configured hosts. The changes to [RFC2131][RFC3315] defined in this document defines the use of 3GPP specific option transferred to DHCP server by DHCP client.

2. Convention & 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 [RFC 2119].

The terminology in this document is based on the definitions in [RFC2131][RFC3315], in addition to the ones specified in this section

3GPP 3rd Generation Partnership Project

TS Technical Specification

EPC Evolved Packet Core

APN Access Point Name

PDN Packet Data Network

GTP GPRS Tunnelling Protocol

PMIP Proxy Mobile IP

3. Problem Statement

In 3GPP TS 23.402, 3GPP UE can access the 3GPP EPC through non-3GPP IP access.

When the non 3GPP access is Trusted, there is no need for the 3GPP UE to establish a Layer 3 tunnel (IPSec [RFC4301], DSMIPv6 [RFC 5555]) to access the EPC as it can rely upon the non 3GPP access security mechanisms. In this case, the 3GPP UE may send DHCP signaling to non 3GPP access to acquire an IP address. However, the 3GPP UE may wish to associate its request for an IP address with IP services provided by the 3GPP EPC or may conversely explicitly require that its traffic is not sent to the EPC but offloaded, even though the UE has been authenticated via its 3GPP credentials (the service is called "NSO" i.e. Non Seamless Offload).

The APN (Access Point Name) is the parameter by which a 3GPP UE signals the kind of EPC service it desires. Based on the value of this parameter, a 3GPP IP Edge (a PDN GW) is selected and the PDN GW is able to determine the IP features to deliver to the traffic exchanged by the UE as part of this APN.

It should be noted that the set of IP configuration parameters that the UE may receive via a DHCP response (e.g. the DNS server(s) address) may also be influenced by the value of the APN.

Thus when requesting an IP address via DHCP, a 3GPP UE should be able to indicate:

- o Whether this IP address is for NSO or whether it is for an EPC service.
- o If the IP address is for an EPC service, the value of the APN corresponding to the IP services reached when using this IP address. The UE may also request an EPC service without providing an APN value; in this case, the network uses a default APN value.

Such DHCP request may be sent when the UE requests access to EPC via non 3GPP access, or when the UE performs mobility from a 3GPP access to a non 3GPP access.

This document is intended to define a new DHCP option for "3GPP Service".

4. 3GPP-Service Option Format

A DHCP client within a 3GPP based device sets the "3GPP-Service" Option in the DHCP request it sends to the network to indicate the desired 3GPP service associated with that request.

A DHCP server willing to indicate it has taken into account the parameters / sub-options of a "3GPP-Service" Option included by the client in a DHCP request mirrors these parameters in the DHCP signaling it sends back to the client.

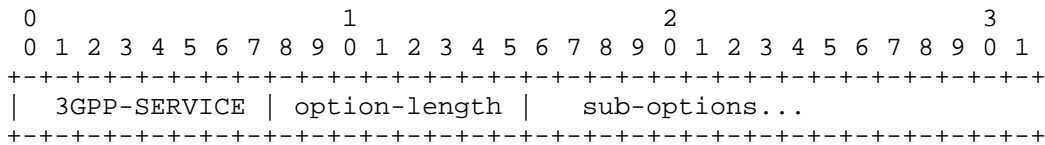


Figure 1: 3GPP Service Option Format for DHCPv4

option-code	3GPP-Service (TBD)
option-len	The number of octets in the option, minimum 1.
Sub-options	The "3GPP-Service" Option may contain one or more Sub-options further defined in this document

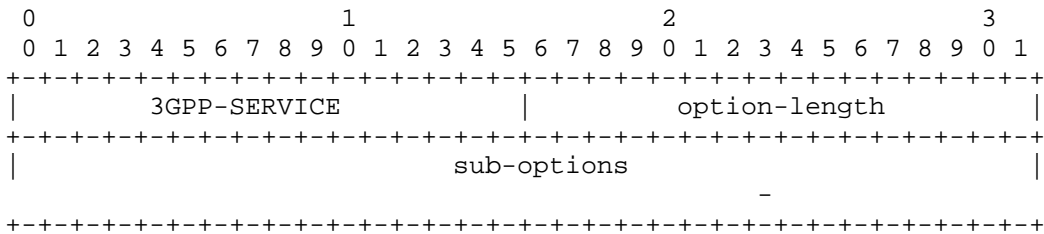


Figure 2: 3GPP Service Option Format for DHCPv6

option-code	3GPP-Service (TBD)
option-len	The number of octets in the option, minimum 1.
Sub-options	The "3GPP-Service" Option may contain one or more Sub-options further defined in this document

4.1. APN Sub-option Format

The purpose of 3GPP-APN (Access Point Name) Sub-option is sent by UE to network for identifying the packet data network associated with the DHCP message. The APN is defined in 3GPP TS 23.401[TS23401].

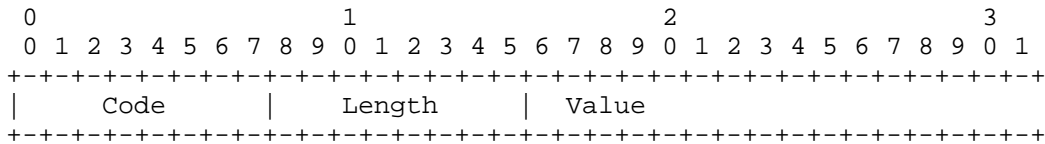


Figure 3: APN Sub-option Format for DHCPv4

Code3GPP-APN(TBD).

LengthThe number of octets in the option, minimum 1.

ValueThe APN value, as defined in 3GPP TS 24.008 [TS24008] section 1

0.5.6.1.

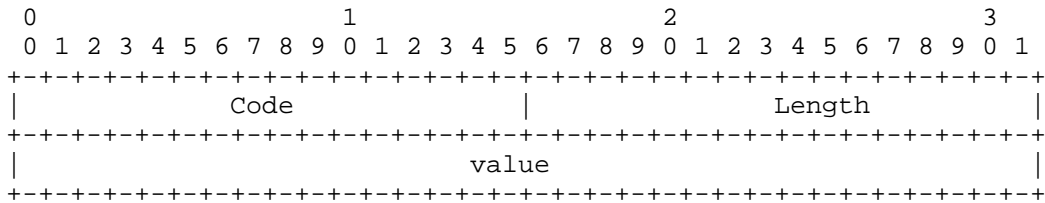


Figure 4: APN Sub-option Format for DHCPv6

Code3GPP-APN(TBD).

LengthThe number of octets in the option, minimum 1.

ValueThe APN value, as defined in 3GPP TS 24.008 [TS24008] section 1

0.5.6.1.

The 3GPP-APN Sub-option SHOULD only be present once in a DHCP message. If it is present more than once, the value of the last occurrence of the option supersedes the value of the other occurrences of this sub-option.

4.2. Non Seamless WLAN Offload (3GPP-Service-Type)Sub-option Format

The purpose of 3GPP-Service-Type (Non Seamless Offload) Sub-option is to identify whether the DHCP message is to be associated with a Non Seamless Offload service or with an EPC service.

The 3GPP Non Seamless Offload service is defined in 3GPP TS 23.402 [TS23402].

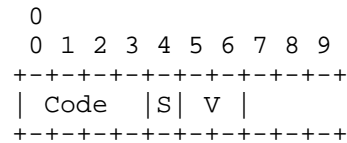


Figure 5: 3GPP-Service-Type Sub-option Format for DHCPv4

Code3GPP-Service-Type (TBD).

SBit 4 of octet 1 is spare and shall be coded as zero.

VThe 3GPP-Service-Type value, the defination is as follows,

Bits0 1

0 1NSO

0 0EPC

All other values are reserved.

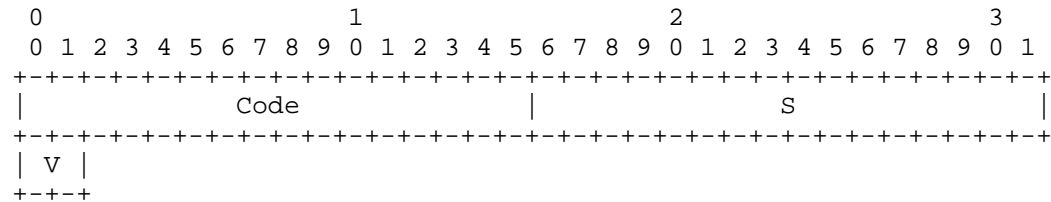


Figure 6: 3GPP-Service-Type Sub-option Format for DHCPv6

Code3GPP-Service-Type (TBD).

SThis is spare and shall be coded as zero.

VThe 3GPP-Service-Type value, the defination of is as follows,

Bits0 1

0 1NSO

0 0EPC

All other values are reserved.

The presence of the 3GPP-Service-Type Sub-option in a DHCP message indicates whether the DHCP signalling is associated with a NSO service or an EPC service rendered to the UE.

The 3GPP-Service-Type Sub-option SHOULD NOT be provided in the same

DHCP message than the 3GPP-APN Sub-option, when the 3GPP-Service-Type sub-option includes NSO indication. If present, the 3GPP-APN Sub-option SHOULD be provided in the same DHCP message than the 3GPP-Service-Type Sub-option, when the 3GPP-Service-Type sub-option includes EPC indication. NSO indication and EPC indication are exclusive.

If the 3GPP-Service-Type Sub-option includes NSO indication, and if 3GPP-APN Sub-option is included, the 3GPP-APN Sub-option is ignored.

5. DHCP Client Behavior

For DHCPv4 protocol, a DHCP client may include a "3GPP-Service" option in the option payload of DHCPDISCOVER and DHCP REQUEST messages being sent toward a DHCP server in order to associate its request with IP services provided by the 3GPP EPC or to explicitly require that its traffic is not sent to the EPC but offloaded

For DHCPv6 protocol, DHCP client may include a "3GPP-Service" option in the option payload of SOLICIT message being sent toward a DHCP server in order to associate its request with IP services provided by the 3GPP EPC or to explicitly require that its traffic is not sent to the EPC but offloaded.

If the answer from the DHCP server does not include a "3GPP-Service" option, the DHCP client assumes that the answer is not associated with an EPC service.

6. DHCP Server Behavior

A DHCP server that receives a "3GPP-Service" option from a DHCP client but does not support this option, MUST ignore this option and MUST NOT provide this option in the signaling it sends back towards the DHCP client.

Conversely, when a DHCP server has taken this option into account it MUST provide this option in the signaling it sends back towards the DHCP client.

A DHCP server supporting this option SHOULD take the content of this option into account when trying to serve the DHCP client. This may involve making sure relevant signaling (e.g. GTP, PMIP [RFC5213]) is sent to an EPC gateway determined using the parameters of the "3GPP-Service" option.

7. Security Considerations

Security considerations in DHCPv4 are described in [RFC3118].

Security considerations in DHCPv6 are described in [RFC3315].

8. IANA Considerations

IANA is requested to assign one new DHCP option code defined in section 5.

9. 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.
- [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.
- [TS23401] 3GPP, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network(E-UTRAN) access", 2011.
- [TS23402] 3GPP, "Architecture enhancements for non-3GPP accesses", 2011.
- [TS24008] 3GPP, "Mobile radio interface Layer 3 specification; Core network protocols", 2010.

Authors' Addresses

Guoyan Liu
ZTE Corporation
No.68 Zijinghua Avenue, Yuhuatai District
Nanjing
China

Phone: +86-25-5287-1362
Email: liu.guoyan@zte.com.cn

Yangwei Tu
ZTE Corporation
No.68 Zijinghua Avenue, Yuhuatai District
Nanjing
China

Phone: +86-25-5287-1362
Email: tu.yangwei@zte.com.cn

Chunhui Zhu
ZTE Corporation
No.68 Zijinghua Avenue, Yuhuatai District
Nanjing
China

Phone: +86-25-5287-4634
Email: zhu.chunhui@zte.com.cn

Wim Hendericks
Alcatel-Lucent

Email: Wim.Henderickx@alcatel-lucent.com

Daniel Derksen
Alcatel-Lucent

Email: Daniel.Derksen@alcatel-lucent.com

Laurent Thiebaut
Alcatel-Lucent

Email: laurent.thiebaut@alcatel-lucent.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 10, 2013

Q. Sun
Tsinghua University
Y. Lee
Comcast
Q. Sun
China Telecom
G. Bajko
Nokia
M. Boucadair
France Telecom
October 7, 2012

Dynamic Host Configuration Protocol (DHCP) Option for Port Set
Assignment
draft-sun-dhc-port-set-option-00

Abstract

Because of the exhaustion of the IPv4 address space, several techniques have been proposed to share the same IPv4 address among several uses. As an alternative to introducing a level of NAT in the provider's core network, this document provides a mechanism to assign non-overlapping port set to users assigned with the same IPv4 address: Port Set DHCPv4 Option.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. DHCPv4 Port Set Option	3
3.1. Port Set Option Format	3
3.2. Port Set Option Example	4
4. Server Behavior	5
5. Client Behavior	5
6. DHCP Unicast Considerations	5
6.1. Server Behavior	6
6.2. Client Behavior	6
7. Security Consideration	6
7.1. Denial-of-Service	6
7.2. Port Randomization	6
8. IANA Consideration	7
9. Contributors List	7
10. References	7
10.1. Normative References	7
10.2. Informative References	8

1. Introduction

Currently some large ISPs still have a large enough IPv4 address pool to be able to allocate public IPv4 addresses for their subscribers. However, due to the exhaustion of the global IPv4 address space, these ISP expect the situation is unsustainable and they will not be able anymore to assign to every requesting host a public IPv4 address.

Two solutions have been proposed so far: (1) Deploy Network Address Translation (NAT) or (2) Allocate the same public IPv4 address with non-overlapped port sets directly to multiple connected devices (which can be CPEs or end hosts). This document focuses on the second solution.

This document describes a new DHCPv4 option which allows the DHCPv4 server to assign a set of ports to a user device during the IPv4 address provisioning process. By assigning the same IPv4 address with non-overlapped port sets to multiple clients, the clients can share the IPv4 address and continue to deliver IPv4 services to subscribers.

The Port Set Option described in this document can be used in various deployment scenarios, some of which are described in [RFC6346]

2. Requirements Language

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

3. DHCPv4 Port Set Option

3.1. Port Set Option Format

The format of Port Set Option is shown in Figure 1.

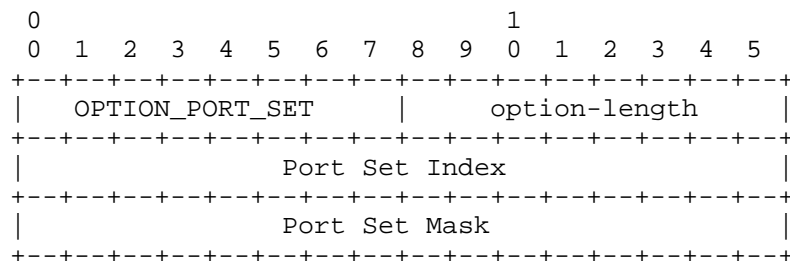


Figure 1 Port Set Option Format

- o option-code: OPTION_PORT_SET (TBD)
- o option-length: An 8-bit field indicating the length of the option excluding the 'Option Code' and the 'Option Length' fields. In this option, the option-length is 4 octets.
- o Port Set Index: Port Set Index identifies a set of ports assigned to a device. The first k bits on the left of the 2-octet field is the Port Set Index value, with the rest of the field right padding zeros.
- o Port Set Mask: Port Set Mask indicates the position of the bits used to build the mask. The first k bits on the left is padding ones while the remained (16-k) bits of the 2-octet field on the right is padding zeros.

In the context of Port Set Option, the port number should consist of port set prefix and port number suffix. The port set prefix can be got from Port Set Index and Port Set Mask, while port number suffix can change continuously. The format of port number is shown in Figure 2.

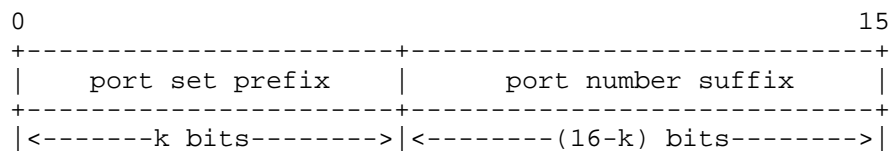


Figure 2 Bit Representation of a port number

In order to exclude the system ports ([I-D.ietf-tsvwg-iana-ports]) or ports saved by SPs, the former port-sets that contains well-known ports SHOULD NOT be assigned.

For example: If k is 10 (the left 10 bits of Port Set Mask is '1'), the first 16 port sets is located in well-known port space, which should not be allocated. Or,

For example: If k is 4 (the left 4 bits of Port Set Mask is '1'), the first port set (0 - 4095) contains the well-know port space. It should be perceived as well.

3.2. Port Set Option Example

The Port Set Option is used to specify one contiguous port set pertaining to the given IP address.

Concretely, this option is used to notify a remote DHCP client about

the port set prefix to be applied when selecting a port value as a source port. The Port Set Option is used to infer a set of allowed contiguous port values. Two port numbers are said to belong to the same Port Set if and only if, they have the same port set prefix.

The following Port Set Index and Port Set Mask are conveyed using DHCP to assign a contiguous port set with excluding well-know ports (with Port Set Index not zero):

Port Set Index: 0001 0100 0000 0000 (5120)

Port Set Mask: 1111 1100 0000 0000 (64512)

The device will get a contiguous port set: 5120 - 6143

4. Server Behavior

The server will not reply with the option until the client has explicitly listed the option code in the Parameter Request List (Option 55).

Server MUST reply with Port Set Option if the client requested `OPTION_PORT_SET` in its Parameter Request List. The server MUST run an address & port-set pool which plays the same role as address pool in regular DHCP server. The address and port-set pool MUST follow the Port-Mask-format port-set.

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

5. Client Behavior

The DHCP client applying for the a port-set MUST include either the `OPTION_PORT_SET` code in the Parameter Request List (Option 55). The client will retrieve a Port Set Option and use the Port Set Index and Port Set Mask to perform the port mask algorithm to get the contiguous port set. The client renews or releases the DHCP lease with the port set.

6. DHCP Unicast Considerations

DHCP messages could be unicasted over UDP port 67. In the context of address sharing, not all the ports are available to the clients. The server cannot use unicast to send the DHCP message to a client which originated the DHCP request. To mitigate this problem, we propose to use the broadcast address (0.0.0.0) when the server replies to the

client. Broadcast address is special and won't be assigned to any client.

6.1. Server Behavior

DHCP server MUST set broadcast bit of the 'flags' field in DHCP messages (Figure 2 of [RFC2131]) when allocating port sets. And DHCP server MUST NOT unicast responses to DHCP client. In order to identify the DHCP responses are sent to which client, client identifier [I-D.ietf-dhc-client-id] is used. DHCP server MUST return client identifier.

6.2. Client Behavior

DHCP client MUST validate client identifier, as specified in [I-D.ietf-dhc-client-id]. DHCP client MUST NOT unicast requests to server: all requests are broadcast. This includes lease renewals. In the case of DHCP relay agent, it will broadcast the server responses to clients.

In some deployment scenarios, DHCP messages containing the proposed DHCP option can be conveyed by other forwarding carrier than IPv4, saying IPv6 [I-D.ietf-dhc-dhcpv4-over-ipv6], etc. The server has to manage to forward DHCP responses to right client.

7. Security Consideration

7.1. Denial-of-Service

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

7.2. Port Randomization

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

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

TCP against Blind In-Window Attacks, use IPv6).

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

8. IANA Consideration

IANA is kindly requested to allocate DHCP option code to the OPTION_PORT_SET. The code should be added to the DHCP option code space.

9. Contributors List

Many thanks for valuable comments and great efforts from the following contributors:

Peng Wu
Tsinghua University

Email: peng-wu@foxmail.com

Teemu Savolainen
Nokia

Email: teemu.savolainen@nokia.com

Ted Lemon
Nominum, Inc.

Email: mellon@nominum.com

Tina Tsou
Huawei Technologies

Email: tena@huawei.com

Pierre Levis
France Telecom

Email: pierre.levis@orange.com

10. References

10.1. Normative References

[RFC1918]

Rekhter, Y., Moskowitz, R.,
Karrenberg, D., Groot, G., and E.
Lear, "Address Allocation for

- Private Internets", BCP 5, RFC 1918, February 1996.
- [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.
- [RFC4925] Li, X., Dawkins, S., Ward, D., and A. Durand, "Softwire Problem Statement", RFC 4925, July 2007.
- [RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, August 2010.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.
- [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.
- [RFC6346] Bush, R., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, August 2011.

10.2. Informative References

- [I-D.bajko-pripaddrassign] Bajko, G., Savolainen, T.,

- Boucadair, M., and P. Levis, "Port Restricted IP Address Assignment", draft-bajko-pripaddrassign-04 (work in progress), April 2012.
- [I-D.ietf-dhc-client-id] Swamy, N., Halwasia, G., and S. Unit, "Client Identifier Option in DHCP Server Replies", draft-ietf-dhc-client-id-06 (work in progress), October 2012.
- [I-D.ietf-dhc-dhcpv4-over-ipv6] Cui, Y., Wu, P., Wu, J., and T. Lemon, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-05 (work in progress), September 2012.
- [I-D.ietf-tsvwg-iana-ports] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", draft-ietf-tsvwg-iana-ports-10 (work in progress), February 2011.
- [I-D.vixie-dnsexst-dns0x20] Vixie, P. and D. Dagon, "Use of Bit 0x20 in DNS Labels to Improve Transaction Identity", draft-vixie-dnsexst-dns0x20-00 (work in progress), March 2008.

Authors' Addresses

Qi Sun
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R.China

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

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

Phone:
EMail: yiu_lee@cable.comcast.com

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

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

Gabor Bajko
Nokia

Phone:
EMail: gabor.Bajko@nokia.com

Mohamed Boucadair
France Telecom
2330 Central Expressway
Rennes 35000
France

Phone:
EMail: mohamed.boucadair@orange.com

