

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 4861 (if approved)
Intended status: Standards Track
Expires: June 17, 2012

F. Gont
UK CPNI
December 15, 2011

Managing the Address Generation Policy for Stateless Address
Autoconfiguration in IPv6
draft-gont-6man-managing-slaac-policy-00

Abstract

This document describes an operational problem that arises due to the impossibility of managing the address generation policy employed by hosts participating in IPv6 Stateless Address Autoconfiguration (SLAAC). Additionally, it specifies a new field in the Prefix Information option of Router Advertisement messages, such that routers can advertise, for each network prefix included in a Router Advertisement message, the desired address generation policy to be used for SLAAC.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 June 17, 2012.

Copyright Notice

Copyright (c) 2011 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. Updating the Prefix Information option	5
3. Router specification	7
3.1. Router configuration	7
3.2. Router operation	7
4. Host specification	8
4.1. Host configuration	8
4.2. Host Operation	8
5. IANA Considerations	10
6. Privacy Considerations	11
7. Security Considerations	12
8. Acknowledgements	13
9. References	14
9.1. Normative References	14
9.2. Informative References	14
Appendix A. Changes from previous versions of the document (to be removed by the RFC Editor before publication of this document as a RFC	16
A.1. Changes from draft-gont-6man-managing-privacy-extensions-01	16
Author's Address	17

1. Introduction

[RFC4862] specifies the Stateless Address Autoconfiguration (SLAAC) for IPv6, which typically results in hosts configuring one or more addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) that typically embeds a hardware address (using the Modified EUI-64 Format [RFC4291]).

Since the identifiers (e.g. Ethernet MAC addresses) typically used for those addresses are usually globally unique, the IPv6 addresses generated as specified in [RFC4291] can be leveraged to track and correlate the activity of a node, thus negatively affecting the privacy of users.

The "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941] were introduced to difficult the task of eavesdroppers and other information collectors to correlate the activities of a node, and basically result in random Interface Identifiers that are typically more difficult to leverage than their Modified EUI-64 Format counterpart. Some flavor of these "Privacy Extensions" have been implemented in a variety of systems, some of which (notably Microsoft Windows Vista and Microsoft Windows 7) enable them by default.

The impossibility of managing the address generation policy employed for SLAAC poses a problem when a site desires or requires a specific policy for the generation of IPv6 addresses. For example, some operating systems (notably FreeBSD) implement "Privacy Extensions", but do not enable them by default. And since there is currently no mechanism in IPv6 to convey the desired address-generation policy, administrators have no option other than manual configuration to enable such extensions. On the other hand, some implementations (notably Windows Vista and Windows 7) that enable "Privacy Extensions" by default might need to be deployed on sites that require the use of stable addresses (e.g. those resulting from Modified EUI-64 Format Identifiers [RFC4291]), for the ease of correlating network activities or enforcing simple access controls. However, since there is currently no mechanism to convey the desired address-generation policy for SLAAC, an administrator would need to manually-configure each of the attached nodes such that they employ the desired address generation policy.

Depending on manual configuration for enabling a specific and homogeneous address-generation policy may result in a lot of work on the side of the administrator, but may also be difficult to implement, particularly when considering mobile nodes such as laptops and mobile phones [Broersma]. Additionally, the lack of a mechanism for conveying address-generation policy information might preclude

the use of some technologies, such as "Privacy Extensions" [RFC4941], which are desirable in most general environments (e.g., a typical home network, or an Internet cafe), but are currently only enabled as a result of manual configuration.

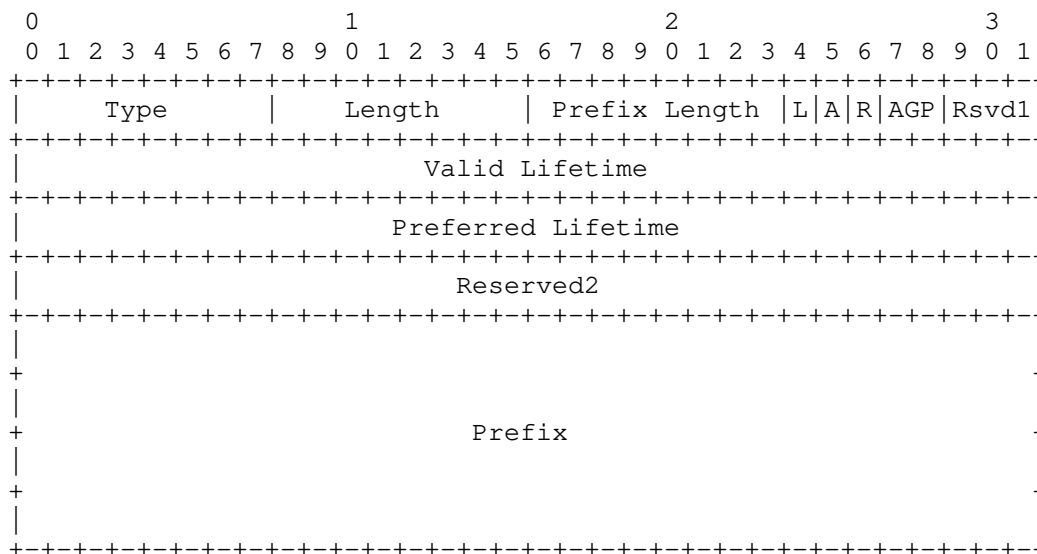
This document specifies a new field in the Prefix Information option of Router Advertisement messages, such that routers can advertise, for each network prefix to be used for SLAAC, the desired policy for the generation of IPv6 addresses. The policy information is simply "advisory" information, in the sense that hosts still have the final word on which address generation policy they use.

The aforementioned policy information basically indicates whether "stable" or "temporary" addresses are desired. We note that while the only address generation policies that have so far been standardized by the IETF are those based on e.g. IEEE identifiers and "Privacy Extensions for SLAAC", other address generation policies are possible. For example, [STABLE-PRIV] describes an address generation policy which results in interface identifiers that are stable for each prefix used for SLAAC, but that change from one autoconfiguration prefix to another.

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. Updating the Prefix Information option

The syntax of the Prefix Information option is updated as follows:



An additional field, the two-bit "AGP" (Address Generation Policy) field, is specified for the Prefix Information option. The semantics of each of the possible values are:

00:

No specific advice is provided for the generation of addresses for this prefix.

01:

When generating addresses for this prefix, the resulting addresses SHOULD be stable (i.e., not temporary). The resulting stable addresses may be based on Modified EUI-64 Format Identifiers [RFC4291], the stable private identifiers proposed in [STABLE-PRIV], or any other address generation policy specified in the future which results in IPv6 addresses that are stable/constant for that autoconfiguration prefix/subnet.

10:

When generating addresses for this prefix, temporary addresses SHOULD be employed. The resulting addresses may be based on "Privacy Extensions for SLAAC" [RFC4941], or on any other policy which results in temporary Interface Identifiers. Address generation policies that result in stable addresses (such as those

specified in [RFC4941] and [STABLE-PRIV]) SHOULD NOT be used for this prefix.

11:

Unused (reserved for future use). The special value "11" is reserved for future extensions, and MUST NOT be set by routers implementing this specification. Hosts that implement this specification MUST interpret the special value "11" in the same way as "00" (i.e., no specific advice is provided for address generation).

Note: The "R" bit was specified by [RFC3775]. The Rsvd1 field corresponds to the remaining reserved bits, and thus MUST be set to zero by the sender of this option, and ignored by the receiver.

Since the "AGP" bits correspond to a previously "reserved" field, implementations that predate this specification should be setting the AGP field to "00" when sending the option, and ignoring the AGP bits upon receipt.

3. Router specification

3.1. Router configuration

This section specifies a variable that routers implementing this specification MUST support:

DesiredAddressPolicy:

This variable specifies the desired address generation policy for IPv6 addresses resulting from SLAAC. As of this specification, possible values are: "Default", "TemporaryAddresses", and "StableAddresses". This variable SHOULD default to "Default".

3.2. Router operation

A router sending a Prefix Information option MUST set the AGP bits according to the value of the variable DesiredAddressPolicy. The following table specifies which values must be used for the AGP field depending on the value of the DesiredAddressPolicy variable.

DesiredAddressPolicy	AGP field
Default	00
StableAddresses	01
TemporaryAddresses	10

Table 1: Correspondence between DesiredAddressPolicy and AGP bits

4. Host specification

4.1. Host configuration

This section specifies two new variables that hosts implementing this specification MUST support:

AddressPolicyConfiguration:

This variable specifies whether the host should honor the advice conveyed in the AGP field of the received Prefix Information options. There are two possible values for this variable: "Enabled" and "Disabled". This variable SHOULD default to "Enabled".

DefaultAddressPolicy:

This variable specifies the default IPv6 address generation policy that will be employed if AddressPolicyConfiguration is set to "Disabled", or if "AddressPolicyConfiguration" is set to "Enabled" but the AGP field of the received Prefix Information option is set to "00" (i.e., no specific advice is provided for the generation of addresses for this prefix). As of this specification, possible values are "TemporaryAddresses" (e.g. for [RFC4941]) and "StableAddresses" (for [RFC4291] or [STABLE-PRIV]). This variable SHOULD default to "TemporaryAddresses".

A host willing to ignore the advice of the router regarding which policy to use to generate IPv6 addresses MAY do so by setting AddressPolicyConfiguration to "Disabled".

It should be noted that the aforementioned variables might have different granularities. For example, a host could specify a set of EnableAddressPolicy and DefaultAddressPolicy variables on a global basis, on a "per network interface type" basis, on a "per wireless network" basis, etc.

4.2. Host Operation

When generating addresses for the prefix contained in a "Prefix Information Option", hosts implementing this specification MUST proceed as follows:

- o If AddressPolicyConfiguration is set to "Enabled", the host SHOULD employ only the policy specified by the AGP field when generating addresses for this prefix. If no specific advice is provided (i.e., the AGP field is set to "00"), the host SHOULD employ only the policy specified by the DefaultAddressPolicy variable when generating addresses for this prefix.

- o If AddressPolicyConfiguration is Disabled, the host SHOULD employ only the policy specified by the DefaultAddressPolicy variable when generating addresses for this prefix.

5. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

6. Privacy Considerations

As discussed in [RFC4941], IPv6 addresses generated using the Modified EUI-64 Format Identifiers [RFC4291] allow tracking of nodes across networks, since the resulting Interface-ID is a globally-unique value that will remain constant across all networks that the node may connect to.

As specified in Section 3 and Section 4 of this document, the default value for the AGP bits of a Prefix Information option is "01" (no specific advice is provided for the generation of addresses for this prefix), and the default address generation policy (DefaultAddressPolicy) for hosts is set to "TemporaryAddresses". This means that, unless the router or host default settings are overridden, the default settings resulting from this specification will enable the use of "temporary addresses" (such as those specified in [RFC4941]).

Nevertheless, it should be noted that the mechanism specified in this document simply provides the means for a router to convey *advisory* information regarding the desired policy for generating IPv6 addresses when SLAAC is employed: this specification allows hosts to ignore the aforementioned advice when deemed appropriate (by setting AddressPolicyConfiguration to "Disabled"). For example, hosts very concerned with the privacy implications of using interface identifiers that remain constant across networks may set AddressPolicyConfiguration to "Disabled" and DefaultAddressPolicy to "TemporaryAddresses" when connecting to untrusted networks, such that Temporary Addresses (such as those specified in [RFC4941]) are always employed (despite the advice provided by the local router).

Finally, we note that the value and effectiveness of some variants of Temporary Addresses (such as that specified in [RFC4941]) have been questioned in a number of studies [I-D.dupont-ipv6-rfc3041harmful] [Escudero] [CPNI-IPv6]. However, this document does not take a stance about their value and effectiveness.

7. Security Considerations

An attacker could exploit the mechanism specified in this document to cause hosts in a given subnet to disable "Temporary Addresses", thus usually leading to the generation of Interface Identifiers that embed the underlying hardware address (e.g. using Modified EUI-64 Format Identifiers), instead. Thus, in such cases, the privacy of the victim hosts that would have enabled the Privacy Extensions could possibly be reduced.

However, some considerations should be made about such possible attack. Firstly, such an attack would require from an attacker the same effort as any other Neighbor Discovery attack based on crafted Router Advertisement messages [RFC3756] [CPNI-IPv6], most of which would be far more interesting for an attacker than this possible attack vector. For example, a (possibly malicious) router could still cause a host to use Modified EUI-64 Format Identifiers if DHCPv6 [RFC3315] is required for address configuration, and the DHCPv6 server selects the IPv6 addresses to be leased to hosts based on e.g. the source link-layer address of the DHCP requests. Secondly, while the the only policy for generating stable IPv6 addresses that has so far been standardized by the IETF is that based on e.g. IEEE identifiers, there are other possible policies, such as that proposed in [STABLE-PRIV], that lead to stable addresses. That is, use of "stable" identifiers does not necessarily imply that such identifiers remain stable/constant across networks.

In those cases in which the network itself is trusted, but users connected to the same network are not, the possible options for mitigating this and other attack vectors based on crafted Router Advertisement messages include the deployment of the so-called "Router Advertisement guard" mechanism [RFC6104], with the implementation guidelines described in [I-D.gont-v6ops-ra-guard-evasion]. Additionally, SEND (SEcure Neighbor Discovery) [RFC3971] could be potentially deployed to mitigate these and other Neighbor Discovery attacks. However, a number of issues (such as the requirement for Public Key Infrastructure) difficult the deployment of SEND in most general network scenarios [CPNI-IPv6].

8. Acknowledgements

The author would like to thank (in alphabetical order) Mikael Abrahamsson, Ran Atkinson, Brian Carpenter, Christian Huitema, Mark Smith, Mark Townsley, and James Woodyatt, for providing valuable comments on [I-D.gont-6man-managing-privacy-extensions], on which this document is based.

Fernando Gont would like to thank CPNI (<http://www.cpni.gov.uk>) for their continued support.

9. References

9.1. Normative References

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, 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.
- [RFC6104] Chown, T. and S. Venaas, "Rogue IPv6 Router Advertisement Problem Statement", RFC 6104, February 2011.

9.2. Informative References

- [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.
- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [I-D.gont-v6ops-ra-guard-evasion] Gont, F. and U. CPNI, "IPv6 Router Advertisement Guard (RA-Guard) Evasion", draft-gont-v6ops-ra-guard-evasion-01 (work in progress), June 2011.

[I-D.gont-6man-managing-privacy-extensions]

Gont, F. and R. Broersma, "Managing the Use of Privacy Extensions for Stateless Address Autoconfiguration in IPv6", draft-gont-6man-managing-privacy-extensions-01 (work in progress), March 2011.

[I-D.dupont-ipv6-rfc3041harmful]

Dupont, F. and P. Savola, "RFC 3041 Considered Harmful", draft-dupont-ipv6-rfc3041harmful-05 (work in progress), June 2004.

[STABLE-PRIV]

Gont, F., "A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC)", Work in Progress, December 2011, <<http://tools.ietf.org/html/draft-gont-6man-stable-privacy-addresses>>.

[Broersma]

Broersma, R., "IPv6 Everywhere: Living with a Fully IPv6-enabled environment", Australian IPv6 Summit 2010, Melbourne, VIC Australia, October 2010, <http://www.ipv6.org.au/summit/talks/Ron_Broersma.pdf>.

[Escudero]

Escudero, A., "PRIVACY EXTENSIONS FOR STATELESS ADDRESS AUTOCONFIGURATION IN IPV6 - "REQUIREMENTS FOR UNOBSERVABILITY", RVK02, Stockholm, 2002, <<http://web.it.kth.se/~aep/PhD/docs/paper3-rvk2002.pdf>>.

[CPNI-IPv6]

Gont, F., "Security Assessment of the Internet Protocol version 6 (IPv6)", UK Centre for the Protection of National Infrastructure, (available on request).

Appendix A. Changes from previous versions of the document (to be removed by the RFC Editor before publication of this document as a RFC)

A.1. Changes from draft-gont-6man-managing-privacy-extensions-01

- o The address-generation policy information has been changed from "'Modified EUI-64' vs. 'Privacy Addresses'" to "'stable addresses' vs. 'temporary addresses'", thus noting that more than one possible policy exists for each category.
- o The appendix on possible alternative specifications for the AGP bits has been removed.
- o The document now focuses on a mechanism that would enable increased use of "temporary addresses" (such as "Privacy Extensions").

Author's Address

Fernando Gont
UK CPNI

Email: fgont@si6networks.com
URI: <http://www.cpni.gov.uk>

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 2460, 5722 (if approved)
Intended status: Standards Track
Expires: September 30, 2012

F. Gont
UK CPNI
March 29, 2012

Security Implications of Predictable Fragment Identification Values
draft-gont-6man-predictable-fragment-id-02

Abstract

IPv6 specifies the Fragment Header, which is employed for the fragmentation and reassembly mechanisms. The Fragment Header contains an "Identification" field which, together with the IPv6 Source Address and the IPv6 Destination Address of the packet, identifies fragments that correspond to the same original datagram, such that they can be reassembled together at the receiving host. The only requirement for setting the "Identification" value is that it must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address. Some implementations simply use a global counter for setting the Fragment Identification field, thus leading to predictable values. This document analyzes the security implications of predictable Identification values, and updates RFC 2460 specifying additional requirements for setting the Fragment Identification, such that the aforementioned security implications are mitigated.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 September 30, 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. Introduction	3
2. Security Implications of Predictable Fragment Identification values	4
3. Updating RFC 2460	7
4. Constraints for the selection of Fragment Identification Values	8
5. Algorithms for Selecting Fragment Identification Values	9
5.1. Per-destination counter (initialized to a random value)	9
5.2. Randomized Identification values	10
5.3. Hash-based Fragment Identification selection algorithm	10
6. IANA Considerations	13
7. Security Considerations	14
8. Acknowledgements	15
9. References	16
9.1. Normative References	16
9.2. Informative References	16
Appendix A. Information leakage produced by vulnerable implementations	18
Appendix B. Changes from previous versions of the document (to be removed by the RFC Editor before publication of this document as a RFC)	20
B.1. Changes from draft-gont-6man-predictable-fragment-id-01	20
B.2. Changes from draft-gont-6man-predictable-fragment-id-00	20
Author's Address	21

1. Introduction

IPv6 specifies the Fragment Header, which is employed for the fragmentation and reassembly mechanisms. The Fragment Header contains an "Identification" field which, together with the IPv6 Source Address and the IPv6 Destination Address of the packet, identifies fragments that correspond to the same original datagram, such that they can be reassembled together at the receiving host. The only requirement for setting the "Identification" value is that it must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address.

The most trivial algorithm to avoid reusing Fragment Identification values too quickly is to maintain a global counter that is incremented for each fragmented packet that is sent. However, this trivial algorithm leads to predictable Identification values, which can be leveraged for performing a variety of attacks.

Section 2 of this document analyzes the security implications of predictable Identification values. Section 3 updates RFC 2460 by adding the requirement that Identification values not be predictable by an off-path attacker. Section 4 discusses constraints in the possible algorithms for selecting Fragment Identification values. Finally, Section 5 specifies a number of algorithms that could be used for generating Identification values.

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. Security Implications of Predictable Fragment Identification values

Predictable Identification values result in an information leakage that can be exploited in a number of ways. Among others, they may potentially be exploited to:

- o determine the packet rate at which a given system is transmitting information,
- o perform stealth port scans to a third-party,
- o uncover the rules of a number of firewalls,
- o count the number of systems behind a middle-box, or,
- o perform a Denial of Service (DoS) attack

[CPNI-IPv6] contains a detailed analysis of possible vulnerabilities introduced by predictable Fragment Identification values. In summary, their security implications are very similar to those of predictable Identification values in IPv4.

[Sanfilippo1998a] originally pointed out how the IPv4 Identification field could be examined to determine the packet rate at which a given system is transmitting information. Later, [Sanfilippo1998b] described how a system with such an implementation could be used to perform a stealth port scan to a third (victim) host. [Sanfilippo1999] explained how to exploit this implementation strategy to uncover the rules of a number of firewalls. [Bellovin2002] explains how the IPv4 Identification field can be exploited to count the number of systems behind a NAT. [Fyodor2004] is an entire paper on most (if not all) the ways to exploit the information provided by the Identification field of the IPv4 header (and these results apply in a similar way to IPv6).

One key difference between the IPv4 case and the IPv6 case is that in IPv4 the Identification field is part of the fixed IPv4 header (and thus usually set for all packets), while in IPv6 the Identification field is set only in those packets that employ a Fragment Header. As a result, successful exploitation of the Identification field against communication instances with arbitrary destinations depends on two different factors:

- o IPv6 implementations using predictable Identification values, and,
- o the ability of the attacker to cause the victim host to fragment packets destined to other nodes

As noted in the previous section, some implementations are known to use predictable identification values.

For example, Linux 2.6.38-8 sets the Identification field according to a global counter that is incremented by one for each datagram that is sent with a fragment header (either a single fragment or as multiple fragments).

Finally, we note that an attacker could cause a victim host to fragment its outgoing packets by sending it a forged ICMPv6 'Packet Too Big' error message advertising a Next-Hop MTU smaller than 1280 bytes.

RFC 1981 [RFC1981] states that when an ICMPv6 Packet Too Big error message with an MTU smaller than 1280 bytes is received, the receiving host is not required to reduce the Path-MTU for the corresponding destination address, but must simply include a Fragment Header in all subsequent packets sent to that destination. In order to make sure that the forged ICMPv6 Packet Too Big error message triggers fragmentation at the victim host, the attacker could set the MTU field of the error message to a value smaller than 1280 bytes. Since the minimum IPv6 MTU is 1280 bytes, such value would always be smaller than the Path-MTU in use for that destination.

There are a few issues that should be considered, though:

- o In all the implementations the author is aware of, an attacker can only cause the victim to enable fragmentation on a per-destination basis. That is, the victim will use fragmentation only for those packets sent to the Source Address of IPv6 packet embedded in the payload of the ICMPv6 Packet Too Big error message.

Section 5.2 of [RFC1981] notes that an implementation could maintain a single system-wide PMTU value to be used for all packets originating from that nodes. Clearly, such an implementations would exacerbate the problem of any attacks based on PMTUD [RFC5927] or IPv6 fragmentation.

- o If the victim node implements some of the counter-measures for ICMP attacks described in RFC 5927 [RFC5927], it might be difficult for an attacker to cause the victim node to use fragmentation for its outgoing packets.

Some implementations do not incorporate countermeasures for attacks based on ICMPv6 error messages. For example, Linux 2.6.38-8 does not even require received ICMPv6 error messages to correspond to ongoing communication instances.

Implementations that employ predictable Identification values and also fail to include countermeasures against attacks based on ICMPv6 error messages will be vulnerable to attacks similar to those based on the IPv4 Identification field for IPv4 networks, such as the stealth port-scanning technique described in [Sanfilippo1998b].

One possible way in which predictable Identification values could be leveraged for performing a Denial of Service (DoS) attack is as follows: once the Identification value currently in use at the victim host has been learned, the attacker would send a forged ICMPv6 Packet Too Big error message to the victim host, with the IPv6 Destination Address of the embedded IPv6 packet set to the IPv6 address of a third-party host with which the victim is communicating. This ICMPv6 Packet Too Big error message would cause any packets sent from the victim to the third-party host to include a Fragment Header. The attacker would then send forged IPv6 fragments to the third-party host, with their IPv6 Source Address set to that of the victim host, and with the Identification field of the forged fragments set to values that would result in collisions at the third-party host. If the third-party host discards fragments that result in collisions of Identification values, the attacker could simply trash the Identification space by sending multiple forged fragments with different Identification values, such that any subsequent packets from the victim host are discarded at the third-party host as a result of the malicious fragments sent by the attacker.

For example, Linux 2.6.38-10 is vulnerable to the aforementioned issue.

[I-D.ietf-6man-ipv6-atomic-fragments] describes an improved processing of these packets that would eliminate this specific attack vector, at least in the case of TCP connections that employ the Path-MTU Discovery mechanism.

The aforementioned attack scenario is simply included to illustrate the problem of employing predictable fragment Identification values, rather than to indicate a specific attack vector that needs to be mitigated.

We note that regardless of the attacker's ability to cause a victim host to employ fragmentation when communicating with third-parties, use of predictable Identification values makes communication flows that employ fragmentation vulnerable to any fragmentation-based attacks.

3. Updating RFC 2460

Hereby we update RFC 2460 [RFC2460] as follows:

The Identification value of the Fragment Header MUST NOT be predictable by an off-path attacker.

4. Constraints for the selection of Fragment Identification Values

the "Identification" field of the Fragmentation Header is 32-bits long. However, when translators [RFC6145] are employed, the "effective" length of the IPv6 Fragment Identification field is 16 bits.

[RFC6145] notes that, when translating in the IPv6-to-IPv4 direction, "if there is a Fragment Header in the IPv6 packet, the last 16 bits of its value MUST be used for the IPv4 identification value". This means that the high-order 16 bits are effectively ignored.

As a result, at least during the IPv6/IPv4 transition/co-existence phase, it is probably safer to assume that only the last 16 bits of the IPv6 Fragment Identification may be used in some cases.

Regarding the selection of Fragment Identification values, the only requirement specified in [RFC2460] is that the Fragment Identification must be different than that of any other fragmented packet sent recently with the same Source Address and Destination Address.

Failure to comply with that requirement might lead to the interoperability problems discussed in [RFC4963].

From a security standpoint, unpredictable Identification values are desirable. However, this is somewhat at odds with the "re-use" requirements specified in [RFC2460].

Finally, since Fragment Identification values need to be selected for each outgoing datagram that requires fragmentation, the performance aspect should be considered when choosing an algorithm for the selection of Fragment Identification values.

5. Algorithms for Selecting Fragment Identification Values

This section specifies a number of algorithms that MAY be used for selecting Fragment Identification values.

5.1. Per-destination counter (initialized to a random value)

1. Whenever a packet must be sent with a Fragment Header, the sending host should perform a look-up in the Destinations Cache an entry corresponding to the intended Destination Address.
2. If such an entry exists, it contains the last Fragment Identification value used for that Destination. Therefore, such value should be incremented by 1, and used for setting the Fragment Identification value of the outgoing packet. Additionally, the updated value should be recorded in the corresponding entry of the Destination Cache.
3. If such an entry does not exist, it should be created, and the "Identification" value for that destination should be initialized with a random value (e.g., with a pseudorandom number generator), and used for setting the Identification field of the Fragment Header of the outgoing packet.

The advantages of this algorithm are:

- o It is simple to implement, with the only complexity residing in the Pseudo-Random Number Generator (PRNG) used to initialize the "Identification" value contained in each entry of the Destinations Cache.
- o The "Identification" re-use frequency will typically be lower than that achieved by a global counter (when sending traffic to multiple destinations), since this algorithm uses per-destination counters (rather than a single system-wide counter).
- o It has good performance properties (once the corresponding entry in the Destinations Cache has been created, each subsequent "Identification" value simply involves the increment of a counter).

The possible drawbacks of this algorithm are:

- o If as a result of resource management an entry of the Destinations Cache must be removed, the last Fragment Identification value used for that Destination is obviously lost. Thus, if subsequent traffic to that destination causes the aforementioned entry to be re-created, the Fragment Identification value will be randomized,

thus possibly leading to Fragment Identification "collisions".

- o Since the Fragment Identification values are predictable by the destination host, a vulnerable host might possible leak to third-parties the Fragment Identification values used by other hosts to send traffic to it (i.e., Host B could leak to Host C the Fragment Identification values that Host A is using to send packets to Host B).

Appendix A describes a scenario in which that information leakage could take place.

5.2. Randomized Identification values

Clearly, use of a Pseudo-Random Number Generator for selecting the Fragment Identification could be desirable from a security standpoint. With such a scheme, the Fragment Identification of each fragmented datagram would be selected as:

$$\text{Identification} = \text{random}()$$

where "random()" is the PRNG.

The specific properties of such scheme would clearly depend on the specific PRNG algorithm used. For example, some PRNGs may result in higher Fragment Identification reuse frequencies than others, in the same way as some PRNGs may be more expensive (in terms of processing requirements and/or implementation complexity) than others.

Discussion of the properties of possible PRNGs is considered out of the scope of this document. However, we do note that some PRNGs employed in the past by some implementations have been found to be predictable [Klein2007]. Please see [RFC4086] for randomness requirements for security.

5.3. Hash-based Fragment Identification selection algorithm

Another alternative is to implement a hash-based algorithm similar to that specified in for the selection of transport port numbers. With such a scheme, the Fragment Identification value of each fragment datagram would be selected with the expression:

$$\text{Identification} = F(\text{Src IP}, \text{Dst IP}, \text{secret1}) + \text{counter}[G(\text{src IP}, \text{Dst Pref}, \text{secret2})]$$

where:

Identification:

Identification value to be used for the fragmented datagram

F():

Hash function

Src IP:

IPv6 Source Address of the datagram to be fragmented

Dst IP:

IPv6 Destination Address of the datagram to be fragmented

secret1:

Secret data unknown to the attacker

counter[]:

System-wide array of 32-bit counters (e.g. with 8K elements or more)

G():

Hash function. May or may not be the same hash function as that used for F()

Dst Pref:

IPv6 "Destination Prefix" of datagram to be fragmented (can be assumed to be the first eight bytes of the Destination Address of such packet). Note: the "Destination Prefix" (rather than Destination Address) is used, such that the ability of an attacker of searching the "increments" space by using multiple addresses of the same subnet is reduced.

secret1:

Secret data unknown to the attacker

Note: counter[G(src IP, Dst Pref, secret2)] should be incremented by one each time an Identification value is selected.

The advantages of this algorithm are:

- o The "Identification" re-use frequency will typically be lower than that achieved by a global counter (when sending traffic to multiple destinations), since this algorithm uses multiple system-wide counters (rather than a single system-wide counter). The extent to which the re-use frequency will be lower will depend on the number of elements in counter[], and the number of other active flows that result in the same value of G() (and hence cause the same counter to be incremented for each fragmented datagram that is sent).

- o It is possible to implement the algorithm such that good performance is achieved. For example, the result of F() could be stored in the Destinations Cache (such that it need not be recomputed for each packet that must be sent) along with computed *index* for counter[].

It should be noted that if this implementation approach is followed, and an entry of the Destinations Cache must be removed as a result of resource management, the last Fragment Identification value used for that Destination will *not* be lost. This is an improvement over the algorithm specified in Section 5.1.

The possible drawbacks of this algorithm are:

- o Since the Fragment Identification values are predictable by the destination host, a vulnerable host could possibly leak to third-parties the Fragment Identification values used by other hosts to send traffic to it (i.e., Host B could leak to Host C the Fragment Identification values that Host A is using to send packets to Host B).

Appendix A describes a scenario in which that information leakage could take place. We note, however, that this algorithm makes the aforementioned attack less reliable for the attacker, since each counter could be possibly shared by multiple traffic flows (i.e., packets destined to other destinations might cause the counter to be incremented).

This algorithm might be preferable (over the one specified in Section 5.1) in those scenarios in which a node is expected to communicate with a large number of destinations, and thus it is desirable to limit the amount of information to be maintained in memory.

In such scenarios, if the algorithm specified in Section 5.1 were implemented, entries from the Destinations Cache might need to be pruned frequently, thus increasing the risk of fragment Identification collisions.

6. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

7. Security Considerations

This document discusses the security implications of predictable Fragment Identification values, and updates RFC 2460 such that Fragment Identification values are required to be unpredictable by off-path attackers, hence mitigating the aforementioned security implications.

A number of possible algorithms are specified, to provide some implementation alternatives to implementers. However, the selection of an specific algorithm that complies with Section 3 is left to implementers. We note that the selection of such an algorithm usually implies a number of trade-offs (security, performance, implementation complexity, interoperability properties, etc.).

8. Acknowledgements

The author would like to thank Ivan Arce for proposing the attack scenario described in Appendix A, and for providing valuable comments on earlier versions of this document.

The author would like to thank Dave Thaler for providing valuable comments on earlier versions of this document.

This document is based on the technical report "Security Assessment of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6] authored by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

Fernando Gont would like to thank the UK CPNI (<http://www.cpni.gov.uk>) for their continued support.

9. References

9.1. Normative References

- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, December 2009.
- [RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

9.2. Informative References

- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.
- [RFC6274] Gont, F., "Security Assessment of the Internet Protocol Version 4", RFC 6274, July 2011.
- [I-D.ietf-6man-ipv6-atomic-fragments]
Gont, F., "Processing of IPv6 "atomic" fragments",
draft-ietf-6man-ipv6-atomic-fragments-00 (work in
progress), February 2012.
- [Bellovin2002]
Bellovin, S., "A Technique for Counting NATted Hosts",
IMW'02 Nov. 6-8, 2002, Marseille, France, 2002.
- [CPNI-IPv6]
Gont, F., "Security Assessment of the Internet Protocol
version 6 (IPv6)", UK Centre for the Protection of

National Infrastructure, (available on request).

[Fyodor2004]

Fyodor, "Idle scanning and related IP ID games", 2004,
<<http://www.insecure.org/nmap/idlescan.html>>.

[Klein2007]

Klein, A., "OpenBSD DNS Cache Poisoning and Multiple O/S
Predictable IP ID Vulnerability", 2007, <[http://
www.trusteer.com/files/
OpenBSD_DNS_Cache_Poisoning_and_Multiple_OS_Predictable_IP
_ID_Vulnerability.pdf](http://www.trusteer.com/files/OpenBSD_DNS_Cache_Poisoning_and_Multiple_OS_Predictable_IP_ID_Vulnerability.pdf)>.

[Sanfilippo1998a]

Sanfilippo, S., "about the ip header id", Post to Bugtraq
mailing-list, Mon Dec 14 1998,
<<http://www.kyuzz.org/antirez/papers/ipid.html>>.

[Sanfilippo1998b]

Sanfilippo, S., "Idle scan", Post to Bugtraq mailing-list,
1998, <<http://www.kyuzz.org/antirez/papers/dumbscan.html>>.

[Sanfilippo1999]

Sanfilippo, S., "more ip id", Post to Bugtraq mailing-
list, 1999,
<<http://www.kyuzz.org/antirez/papers/moreipid.html>>.

Appendix A. Information leakage produced by vulnerable implementations

Section 2 provides a number of references describing a number of ways in which the information leakage produced by a vulnerable implementation could be leveraged by an attacker. This section describes a specific network scenario in which a vulnerable implementation could possibly leak the current Fragment Identification value in use by a third-party host to send fragmented datagrams to the vulnerable implementation.

For the most part, this section is included to illustrate how a vulnerable implementation might be leveraged to leak-out the Fragment Identification value of an otherwise secure implementation. This section might be removed in future revisions of this document.

The following scenarios assume:

- A: Is an IPv6 host that implements the recommended Fragment Identification algorithm (Section 5.1), implements [RFC5722], but does not implement [I-D.ietf-6man-ipv6-atomic-fragments].
- B: Victim node. Selected the Fragment Identification values from a global counter.
- C: Attacker. Can forge the IPv6 Source Address of his packets at will.

If the attacker sends forged SYN packets to a closed TCP port, and then fails when trying to produce a collision of Fragment Identifications (see line #4), the following packet exchange might take place:

A	B	C
#1		<----- Echo Req #1 ----->
#2		--- Echo Resp #1, FID=5000 --->
#3	<----- SYN #1, src= B ----->	
#4		<----- SYN/ACK, FID=42 src = A---
#5	----- SYN/ACK, FID=9000 ---->	
#6	<----- RST, FID= 5001 ----->	
#7	<----- RST, FID= 5002 ----->	
#8		<----- Echo Req #2 ----->
#9		---- Echo Resp #2, FID= 5003 -->

On the other hand, if the attacker succeeds to produce a collision of Fragment Identification values, the following packet exchange could take place:

```

      A                                B                                C

#1                                <----- Echo Req #1 ----->
#2                                ---- Echo Resp #1, FID=5000 ---->
#3 <----- SYN #1, src= B ----->
#4                                <-- SYN/ACK, FID=9000 src=A ---
#5 ---- SYN/ACK, FID=9000 ---->
                                ... (RFC5722) ...
#6                                <----- Echo Req #2 ----->
#7                                ---- Echo Resp #2, FID= 5001 ---->

```

Clearly, the Fragment Identification value sampled by from the second ICMPv6 Echo Response packet ("Echo Resp #2") implicitly indicates whether the Fragment Identification in the forged SYN/ACK (see line #4 in both figures) was the current Fragment Identification in use by Host A.

As a result, the attacker could employ this technique to learn the current Fragment Identification value used by host A to send packets to host B.

Appendix B. Changes from previous versions of the document (to be removed by the RFC Editor before publication of this document as a RFC)

B.1. Changes from draft-gont-6man-predictable-fragment-id-01

- o Recommendation of a specific algorithm has been removed. Nodes are just required to select the Fragment Identification values such that they are not easily predictable by off-path attackers.

B.2. Changes from draft-gont-6man-predictable-fragment-id-00

- o The constraints for the possible Fragment-Identification selection algorithm are now described in more detail (e.g., the I-D now mentioned that translators only use the last 16 bits of the IPv6 Fragment Identification).
- o The I-D now provides a discussion of possible Fragment-Identification selection algorithms, and analyzes the trade-offs.

Author's Address

Fernando Gont
UK CPNI

Email: fgont@si6networks.com
URI: <http://www.cpni.gov.uk>

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 6106 (if approved)
Intended status: Standards Track
Expires: December 17, 2012

F. Gont
SI6 Networks / UTN-FRH
P. Simerda
June 15, 2012

Current issues with DNS Configuration Options for SLAAC
draft-gont-6man-slaac-dns-config-issues-00

Abstract

RFC 6106 specifies two Neighbor Discovery options that can be included in Router Advertisement messages to convey information about DNS recursive servers and DNS Search Lists. Small lifetime values for the aforementioned options have been found to cause interoperability problems in those network scenarios in which these options are used to convey DNS-related information. This document analyzes the aforementioned problem, and formally updates RFC 6106 such that these issues are mitigated.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 17, 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. Introduction	3
2. Possible Solutions	4
2.1. Summary of possible solutions	4
2.2. Changing the Semantics of the 'Lifetime' field of RDNSS and DNSSL options	4
2.3. Changing the Default Values of the 'Lifetime' field of RDNSS and DNSSL options	6
2.4. Use Router Solicitations for active Probing	7
2.5. Sanitize the received RDNSS/DNSSL 'Lifetime' Values	7
3. Security Considerations	9
4. Acknowledgements	10
5. References	11
5.1. Normative References	11
5.2. Informative References	11
Appendix A. Additional notes regarding RFC 6106	12
Authors' Addresses	14

1. Introduction

RFC 6106 [RFC6106] specifies to Neighbor Discovery (ND) [RFC4861] options that can be included in Router Advertisement messages to convey information about DNS recursive servers and DNS Search Lists. Namely, the Recursive DNS Server (RDNSS) Option specifies the IPv6 addresses of recursive DNS servers, while the DNS Search List (DNSSL) Option specifies a "search list" to be used when trying to resolve a name by means of the DNS.

Each of this options include a "Lifetime" field which specifies the maximum time, in seconds, during which the information included in the option can be used by the receiving system. The aforementioned "Lifetime" value is set as a function of the Neighbor Discovery parameter 'MaxRtrAdvInterval', which specifies the maximum time allowed between sending unsolicited multicast Router Advertisements from an interface. The recommended bounds ($\text{MaxRtrAdvInterval} \leq \text{Lifetime} \leq 2 * \text{MaxRtrAdvInterval}$) have been found to be too short for scenarios in which some Router Advertisement messages may be lost. In such scenarios, host may fail to receive unsolicited Router Advertisements and therefore fail to refresh the expiration time of the DNS-related information previously learned through the RDNSS and DNSSL options), thus eventually discarding the aforementioned DNS-related information prematurely.

Some implementations consider the lack of DNS-related information as a hard failure, thus causing configuration restart. This situation is exacerbated in those implementations in which IPv6 connectivity and IPv4 connectivity are bound together, and hence failure in the configuration of one of them causes the whole link to be restarted.

Section 2 proposes a number of ALTERNATIVE solutions to the problem, such that the 6man wg can discuss both of them. It is expected that once the 6man wg converges on a preferred solution, the other ones will be removed from the document.

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. Possible Solutions

This section describes a number of possible ALTERNATIVE solutions to the problem, such that the 6man wg can discuss both of them. It is expected that once the 6man wg converges on a preferred solution, the other one will be removed from the document.

2.1. Summary of possible solutions

Section 2.2 proposes to change the semantics of the RDNSS/DNSSL 'Lifetime', thus fixing the 'expiration' problem outlined in Section 1 and the potential of 'network configuration oscillation'. Section 2.3 proposes to increase the 'Lifetime' value at the sending routers, fixing the "expiration" problem outlined in Section 1, but without addressing the potential of 'network configuration oscillation'. Section 2.4 proposes to send Router Solicitations when expiration of RDNSS/DNSSL information is imminent, to elicit Router Advertisement messages. Section 2.5 proposes to enforce a lower limit on the received RDNSS/DNSSL 'Lifetime' values at the client side.

2.2. Changing the Semantics of the 'Lifetime' field of RDNSS and DNSSL options

The semantics of the 'Lifetime' field of the RDNSS and DNSSL options is updated as follows:

- o The 'Lifetime' field indicates the amount of time during which the aforementioned DNS-related information is expected to be stable.
- o If the information received in a RDNSS or DNSSL option is already present in the corresponding data structures, the corresponding 'Expiration' time should be updated according to the value in the 'Lifetime' field of the received option. A 'Lifetime' of '0' causes the corresponding information to be discarded, as already specified in [RFC6106].
- o If a host has already gathered a sufficient number of RDNSS addresses (or DNS search domain names), and additional data is received while the existing entries have not yet expired, the received RDNSS addresses (or DNS search domain names) SHOULD be ignored.
- o If a host receives new RDNSS addresses (or DNS search domain names), and some of the existing entries have expired, the newly-learned information SHOULD be used to replace the expired entries.

- o A host SHOULD flush configured DNS-related information when it has any reason to believe that its network connectivity has changed in some relevant way (e.g., there has been a "link change event"). When that happens, the host MAY send a Router Solicitation message to re-learn the corresponding DNS-related information.
- o The most-recently-updated information SHOULD have higher priority over the other DNS-related information already present on the local host.

The rationale for the suggested change is as follows:

- o It is a backwards-compatible local-policy change that solves the problem described in Section 1 without requiring changes to router software or router configuration in existing deployments (over which the user is likely to have no control at all).
- o Since different RDNSS and DNSSL information could be sent by the same router in different Router Advertisement messages, the updated semantics of the 'Lifetime' parameter prevents oscillations in network configuration.

This situation could arise for a number of reasons. For example, if the desire for different 'Lifetime' values warrants the use of different RDNSS or DNSSL options, and because of packet size issues each option must be included in a separate Router Advertisement message, each burst of RAs could cause DNS-related information to be reconfigured.

Another possible scenario that could lead to the same situation is that in which there is more than one local router, and each of the local routers announces different RDNSS (or DNSSL) information. If the number of RDNSS addresses (or DNS search domain names) that the local host considers "sufficient" prevents the aggregate set of RDNSS (or DNSSL) information, the local RDNSS (or DNSSL) information would oscillate between that advertised by each of the local routers.

- o The original motivation for enforcing a short expiration timeout value was to allow mobile nodes to prefer local RDNSSes to remote RDNSSes. However, the recommendation in the last bullet above already allows for a timely update of the corresponding DNS-related information. Additionally, since it is already recommended by [RFC6106] to maintain some RDNSS addresses (or DNS search domain names) per source, in the typical scenarios in which a single router (per subnet) advertises configuration information, one of this 'slots' will be free (or have expired information) and readily available to be populated with information learned from

the new subnet to which the host has moved.

2.3. Changing the Default Values of the 'Lifetime' field of RDNSS and DNSSL options

The default RDNSS/DNSSL "Lifetime" value in current the current router solutions vary between MaxRtrAdvInterval and 2*MaxRtrAdvInterval. This means that common packet loss rates can lead to the problem described in this document.

One possible approach to mitigate this issue would be to avoid 'Lifetime' values that are on the same order as MaxRtrAdvInterval. This solution would require, of course, changes in router software.

When specifying a better default value, the following aspects should be considered:

- o IPv6 will be used on many links (including IEEE 802.11) that experience packet loss. Therefore losing a few packets in a short period of time should not invalidate DNS configuration information.
- o Unsolicited Router Advertisements sent on Ethernet networks result in packets that employ multicast Ethernet Destination Addresses. A number of network elements (including those that perform bridging between wireless networks and wired networks) have problems with multicasted Ethernet frames, thus typically leading to packet loss of some of those frames. Therefore, SLAAC implementations should be able to cope with devices that can lose several multicast packets in a row.

The default value of AdvRDNSSLifetime and AdvDNSSLLifetime MUST be at least 5*MaxRtrAdvInterval so that the probability of hosts receiving unsolicited Router Advertisements is increased.

This solution leaves out the following situations:

- o In those scenarios in which the involved routers cannot be updated this solution will not be applicable. This also limitation also applies to nomadic hosts that can connect to many different networks. This case will be discussed later in this document.
- o The affected network has a huge multicast packet loss. This unfortunately happens in some real networks.
- o This solution does not address the potential 'network configuration oscillation' issue described in Section 2.2.

2.4. Use Router Solicitations for active Probing

According to RFC 6106, hosts MAY send Router Solicitations to avoid expiry of RDNSS and DNSSL lifetimes. This technique could be employed as a "last resort" when expiration of the RDNSS and DNSSL information is imminent.

Hosts SHOULD start sending multicast Router Solicitation when most of the Lifetime of the last RDNSS server is consumed. The precise time should be a randomized value chosen from 70% to 90% of the original Lifetime to avoid bursts of packets from other hosts on the network. Hosts MAY send up to MAX_RTR_SOLICITATIONS Router Solicitation messages, each separated by at least RTR_SOLICITATION_INTERVAL seconds (please see Section 6.3.7 of [RFC4861]).

Problems of this approach:

- o If no IPv6 router responds, all previously connected hosts will repeatedly send Router Solicitations and only stop doing so when their RDNSS and DNSSL information finally expires. This could disrupt IPv4 networking in larger networks and that must be avoided.
- o If IPv6 router responds with unicast Router Advertisement, it may need to respond to many clients.
- o There is no other SLAAC information that requires or would benefit from this kind of "active probing".
- o This approach does not mitigate the potential 'network configuration oscillation' problem described in Section 2.2.

NOTE: We should either state a very good reason to specialcase DNS timeouts or deprecate Router Solicitations in RFC 6106 entirely. Deprecation is the more favorable option in our opinion.

2.5. Sanitize the received RDNSS/DNSSL 'Lifetime' Values

Another possible approach would be to enforce a lower limit on the received RDNSS/DNSSL 'Lifetime' values on the client side. The challenge this technique is the selection of a reasonable value. On the router side, it can be derived from MaxRtrAdvInterval but this value is not known to the client side.

Therefore we would have to assume some maximum value of MaxRtrAdvInterval and use it to derive minimum value of lifetimes instead of the actual MaxRtrAdvInterval.

It should be noted that this approach would not address the potential 'network configuration oscillation' issue described in Section 2.2.

3. Security Considerations

This document does not introduce any additional security considerations to those documented in the "Security Considerations" section of [RFC6106].

4. Acknowledgements

5. References

5.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.

5.2. Informative References

Appendix A. Additional notes regarding RFC 6106

Section 5.2 of [RFC6106] states:

An RDNSS address or a DNSSL domain name MUST be used only as long as both the RA router Lifetime (advertised by a Router Advertisement message [RFC4861]) and the corresponding option Lifetime have not expired.

This requirement could introduce problems in scenarios in which the router advertising the RDNSS or DNSSL options is not expected to be employed as a "default router", and hence the 'Router Lifetime' value of its Router Advertisement messages is set to 0.

As noted in Section 4.2 of [RFC4861], the Router Lifetime applies only to the router's usefulness as a default router, and it does not apply to information contained in other message fields or options.

Therefore, it would be sensible to exclude the 'Router Lifetime' information when deciding about the validity or "freshness" of the DNS-related configuration information.

Section 5.3.1 of [RFC6106] states:

When the IPv6 host has gathered a sufficient number (e.g., three) of RDNSS addresses (or DNS search domain names), it SHOULD maintain RDNSS addresses (or DNS search domain names) by the sufficient number such that the latest received RDNSS or DNSSL is more preferred to the old ones; that is, when the number of RDNSS addresses (or DNS search domain names) is already the sufficient number, the new one replaces the old one that will expire first in terms of Lifetime.

As noted earlier in this document, this policy could lead (in some scenarios) to network configuration oscillations. Therefore, it would be sensible to enforce some minimum stability of the configured information, such as that resulting from the update in Section 2.2.

Section 6.3 of [RFC6106] states:

Step (d): For each DNSSL domain name, if it does not exist in the DNS Search List, register the DNSSL domain name and Lifetime with the DNS Search List and then insert the DNSSL domain name in front of the Resolver Repository. In the case where the data structure for the DNS Search List is full of DNSSL domain name entries (that is, has more DNSSL domain names than the sufficient number discussed in Section 5.3.1), delete from the DNS Server List the

entry with the shortest Expiration-time (i.e., the entry that will expire first).

This policy could lead to the same network configuration oscillation problems as described above for the RDNSS addresses. Therefore, it would be sensible to enforce some minimum stability of the configured information, such as that resulting from the update in Section 2.2.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <http://www.si6networks.com>

Pavel Simerda

Phone: +420 775 996 256
Email: pavlix@pavlix.net

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 6106 (if approved)
Intended status: Standards Track
Expires: August 30, 2015

F. Gont
SI6 Networks / UTN-FRH
P. Simerda

W. Liu
Huawei Technologies
February 26, 2015

Current issues with DNS Configuration Options for SLAAC
draft-gont-6man-slaac-dns-config-issues-01

Abstract

RFC 6106 specifies two Neighbor Discovery options that can be included in Router Advertisement messages to convey information about DNS recursive servers and DNS Search Lists. Small lifetime values for the aforementioned options have been found to cause interoperability problems in those network scenarios in which these options are used to convey DNS-related information. This document analyzes the aforementioned problem, and formally updates RFC 6106 such that these issues are mitigated.

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 August 30, 2015.

Copyright Notice

Copyright (c) 2015 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	2
2. Changing the Semantics of the 'Lifetime' field of RDNSS and DNSSL options	3
3. Changing the Default Values of the 'Lifetime' field of RDNSS and DNSSL options	4
4. Use of Router Solicitations for active Probing	4
5. Sanitize the received RDNSS/DNSSL 'Lifetime' Values	5
6. Security Considerations	5
7. Acknowledgements	5
8. Normative References	5
Authors' Addresses	5

1. Introduction

RFC 6106 [RFC6106] specifies two Neighbor Discovery (ND) [RFC4861] options that can be included in Router Advertisement messages to convey information about DNS recursive servers and DNS Search Lists. Namely, the Recursive DNS Server (RDNSS) Option specifies the IPv6 addresses of recursive DNS servers, while the DNS Search List (DNSSL) Option specifies a "search list" to be used when trying to resolve a name by means of the DNS.

Each of this options include a "Lifetime" field which specifies the maximum time, in seconds, during which the information included in the option can be used by the receiving system. The aforementioned "Lifetime" value is set as a function of the Neighbor Discovery parameter 'MaxRtrAdvInterval', which specifies the maximum time allowed between sending unsolicited multicast Router Advertisements from an interface. The recommended bounds ($\text{MaxRtrAdvInterval} \leq \text{Lifetime} \leq 2 * \text{MaxRtrAdvInterval}$) have been found to be too short for scenarios in which some Router Advertisement messages may be lost. In such scenarios, hosts may fail to receive unsolicited Router Advertisements and therefore fail to refresh the expiration time of the DNS-related information previously learned through the RDNSS and DNSSL options), thus eventually discarding the aforementioned DNS-related information prematurely.

Some implementations consider the lack of DNS-related information as a hard failure, thus causing configuration restart. This situation

is exacerbated in those implementations in which IPv6 connectivity and IPv4 connectivity are bound together, and hence failure in the configuration of one of them causes the whole link to be restarted.

This document formally updates RFC 6106 such that this issue is mitigated.

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. Changing the Semantics of the 'Lifetime' field of RDNSS and DNSSL options

The semantics of the 'Lifetime' field of the RDNSS and DNSSL options is updated as follows:

- o The 'Lifetime' field indicates the amount of time during which the aforementioned DNS-related information is expected to be stable. A node is NOT required to discard the DNS-related information once the Lifetime expires.
- o If the information received in a RDNSS or DNSSL option is already present in the corresponding local data structures, the corresponding 'Expiration' time should be updated according to the value in the 'Lifetime' field of the received option. A 'Lifetime' of '0' causes the corresponding information to be discarded, as already specified in [RFC6106].
- o If a host has already gathered a sufficient number of RDNSS addresses (or DNS search domain names), and additional data is received while the existing entries have not yet expired, the received RDNSS addresses (or DNS search domain names) SHOULD be ignored.
- o If a host receives new RDNSS addresses (or DNS search domain names), and some of the existing entries have expired, the newly-learned information SHOULD be used to replace the expired entries.
- o A host SHOULD flush configured DNS-related information when it has any reason to believe that its network connectivity has changed in some relevant way (e.g., there has been a "link change event"). When that happens, the host MAY send a Router Solicitation message to re-learn the corresponding DNS-related information.
- o The most-recently-updated information SHOULD have higher priority over the other DNS-related information already present on the local host.

We note that the original motivation for enforcing a short expiration timeout value was to allow mobile nodes to prefer local RDNSs to remote RDNSs. However, the above rules already allow for a timely update of the corresponding DNS-related information.

3. Changing the Default Values of the 'Lifetime' field of RDNS and DNSSL options

The default RDNS/DNSSL "Lifetime" value in current the current router solutions vary between MaxRtrAdvInterval and 2*MaxRtrAdvInterval. This means that common packet loss rates can lead to the problem described in this document.

One possible approach to mitigate this issue would be to avoid 'Lifetime' values that are on the same order as MaxRtrAdvInterval. This solution would require, of course, changes in router software.

When specifying a better default value, the following aspects should be considered:

- o IPv6 will be used on many links (including IEEE 802.11) that experience packet loss. Therefore losing a few packets in a short period of time should not invalidate DNS configuration information.
- o Unsolicited Router Advertisements sent on Ethernet networks result in packets that employ multicast Ethernet Destination Addresses. A number of network elements (including those that perform bridging between wireless networks and wired networks) have problems with multicasted Ethernet frames, thus typically leading to packet loss of some of those frames. Therefore, SLAAC implementations should be able to cope with devices that can lose several multicast packets in a row.

[RFC6106] is hereby updated as follows:

The default value of AdvRDNSLifetime and AdvDNSSLifetime MUST be at least 10*MaxRtrAdvInterval so that the probability of hosts receiving unsolicited Router Advertisements is increased.

4. Use of Router Solicitations for active Probing

According to RFC 6106, hosts MAY send Router Solicitations to avoid expiry of RDNS and DNSSL lifetimes. This technique could be employed as a "last resort" when expiration of the RDNS and DNSSL information is imminent.

5. Sanitize the received RDNSS/DNSSL 'Lifetime' Values

A host that receives a RDNSS or DNSSL option that has a non-zero Lifetime smaller than $10 \times \text{MaxRtrAdvInterval}$ should employ $10 \times \text{MaxRtrAdvInterval}$ as the Lifetime value of the corresponding RDNSS or DNSSL option.

6. Security Considerations

This document does not introduce any additional security considerations to those documented in the "Security Considerations" section of [RFC6106].

7. Acknowledgements

The authors would like to thank Erik Nordmark and Mark Smith for their valuable input on the topic covered by this document.

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.

Authors' Addresses

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <http://www.si6networks.com>

Pavel Simerda

Phone: +420 775 996 256

Email: pavlix@pavlix.net

Will Liu

Huawei Technologies

Bantian, Longgang District

Shenzhen 518129

P.R. China

Email: liushucheng@huawei.com

6MAN
Internet-Draft
Updates: 3986, 4007 (if approved)
Intended status: Standards Track
Expires: January 12, 2013

B. Carpenter
Univ. of Auckland
R. Hinden
Check Point
July 11, 2012

Representing IPv6 Zone Identifiers in Address Literals and Uniform
Resource Identifiers
draft-ietf-6man-uri-zoneid-02

Abstract

This document describes how the Zone Identifier of an IPv6 scoped address can be represented in a a literal IPv6 address and in a Uniform Resource Identifier that includes such a literal address. It updates RFC 3986 and RFC 4007 accordingly.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 12, 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. Specification	4
3. Web Browsers	5
4. Security Considerations	6
5. IANA Considerations	6
6. Acknowledgements	6
7. Change log [RFC Editor: Please remove]	7
8. References	7
8.1. Normative References	7
8.2. Informative References	8
Appendix A. Alternatives Considered	8
Authors' Addresses	9

1. Introduction

[RFC3986] defined how a literal IPv6 address can be represented in the "host" part of a Uniform Resource Identifier (URI). Subsequently, [RFC4007] extended the text representation of limited-scope IPv6 addresses such that a zone identifier may be concatenated to a literal address, for purposes described in that RFC. Zone identifiers are especially useful in contexts where literal addresses are typically used, for example during fault diagnosis, when it may be essential to specify which interface is used for sending to a link local address. It should be noted that zone identifiers have purely local meaning within the host where they are defined, and they are completely meaningless for any other host. Today, they are only meaningful when attached to addresses with less than global scope, but it is possible that other uses might be defined in the future.

RFC 4007 does not specify how zone identifiers are to be represented in URIs. Practical experience has shown that this feature is useful, in particular when using a web browser for debugging with link local addresses, but as it is undefined, it is not implemented consistently in URI parsers or in browsers.

Some versions of some browsers accept the RFC 4007 syntax for scoped IPv6 addresses embedded in URIs, i.e., they have been coded to interpret the "%" sign according to RFC 4007 instead of RFC 3986. Clearly this approach is very convenient for users, although it formally breaches the syntax rules of RFC 3986. The present document defines an alternative approach that respects and extends the rules of URI syntax, and IPv6 literals in general, to be consistent.

Thus, this document updates [RFC3986] by adding syntax to allow a zone identifier to be included in a literal IPv6 address within a URI. It also updates [RFC4007], in particular by adding a second allowed delimiter for zone identifiers.

It should be noted that in other contexts than a user interface, a zone identifier is mapped into a numeric zone index or interface number. The MIB textual convention [RFC4001] and the socket interface [RFC3493] define this as a 32 bit unsigned integer. The mapping between the human-readable zone identifier string and the numeric value is a host-specific function that varies between operating systems. The present document is concerned only with the human-readable string.

Several alternative solutions were considered while this document was developed. The Appendix briefly describes the alternatives and their advantages and disadvantages.

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

2. Specification

According to RFC 4007, a zone identifier is attached to the textual representation of an IPv6 address by concatenating "%" followed by <zone_id>, where <zone_id> is a string identifying the zone of the address. However, RFC 4007 gives no precise definition of the character set allowed in <zone_id>. There are no rules or de facto standards for this. For example, the first Ethernet interface in a host might be called %0, %1, %en1, %eth0, or whatever the implementer happened to choose.

In a URI, a literal IPv6 address is always embedded between "[" and "]". This document specifies how a <zone_id> can be appended to the address. A <zone_id> SHOULD contain only ASCII characters classified in RFC 3986 as "unreserved", which conveniently excludes "]" in order to simplify parsing.

Unfortunately "%" is always treated as an escape character in a URI, and according to RFC 3986 it MUST therefore itself be escaped in a URI, in the form "%25". For this reason, "-" (hyphen) is used instead as the separator when a <zone_id> is included in a URI. Thus, the scoped address fe80::a%en1 would appear in a URI as http://[fe80::a-en1].

If an operating system uses any other characters in zone or interface identifiers that are not in the "unreserved" character set, they MUST be escaped with a "%" sign according to RFC 3986.

We now present the necessary formal syntax.

In RFC 3986, the IPv6 literal format is formally defined in ABNF [RFC5234] by the following rule:

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
```

To provide support for a zone identifier, the existing syntax of IPv6address is retained, and a zone identifier may be added optionally to any literal address. This allows flexibility for unknown future uses. The rule quoted above from RFC 3986 is replaced by three rules:

```
IP-literal = "[" ( IPv6addrz / IPvFuture ) "]"
```

```
ZoneID = 1*( unreserved / pct-encoded )
```

```
IPv6addrz = IPv6address [ "-" ZoneID ]
```

Section 11 of RFC 4007 is updated to allow "-" as well as "%" as the preceding delimiter of a ZoneID.

The rules in [RFC5952] SHOULD be applied in producing URIs.

RFC 3986 states that URIs have a global scope, but that in some cases their interpretation depends on the end-user's context. URIs including a ZoneID are to be interpreted only in the context of the host where they originate, since the ZoneID is of local significance only.

The 6man WG discussed and rejected an alternative in which the existing syntax of IPv6address would be extended by an option to add the ZoneID only for the case of link-local addresses. It was felt that the present solution offers more flexibility for future uses and is more straightforward to implement.

RFC 4007 offers guidance on how the ZoneID affects interface/address selection inside the IPv6 stack. Note that the behaviour of an IPv6 stack if passed a non-zero zone index for an address other than link-local is undefined.

3. Web Browsers

Due to the lack of a standard in this area, web browsers have been inconsistent in providing for ZoneIDs. Many have no support, but there are examples of ad hoc support. For example, older versions of Firefox allowed the use of a ZoneID preceded by an unescaped "%" character, but this was removed for consistency with RFC 3986. As another example, recent versions of Internet Explorer allow use of a ZoneID preceded by a "%" character escaped as "%25", still beyond the syntax allowed by RFC 3986. This syntax extension is in fact used internally in the Windows operating system and some of its APIs.

In recent years, web browsers have evolved considerably and now accept and parse many forms of input that are not a formal URI. Examples of this include host names, search items, bookmarks, search history, etc. For example the Google Chrome browser now calls the "address bar" the "omnibox" [chrome]. The authors believe it is feasible, and very convenient for users, if browsers also allow (in addition to the formal URI syntax defined in this document) a syntax

that will enable cut and paste. For example:

```
http://[fe80::a%en1]
```

It seems that modern browsers can be adapted to parse this because it is inside of the "[" "]"'s. This would permit the output of commands like `ping6 -w ff02::1%en1` to be "cut and pasted" into a browser address bar. Consequently this document recommends that browsers support this syntax in addition to the formal URI syntax defined above.

4. Security Considerations

The security considerations of [RFC3986] and [RFC4007] apply. In particular, this URI format creates a specific pathway by which a deceitful zone index might be communicated, as mentioned in the final security consideration of RFC 4007. It is emphasised that the format is intended only for debugging purposes, but of course this intention does not prevent misuse.

To limit this risk, implementations SHOULD NOT allow use of this format except for well-defined usages such as sending to link local addresses under prefix `fe80::/10`.

An HTTP server or proxy MUST ignore any ZoneID attached to an incoming URI, as it only has local significance at the sending host.

The addition of a choice between "%" and "-" as the delimiter preceding a ZoneID slightly complicates the string comparison issue discussed in [I-D.iab-identifier-comparison].

5. IANA Considerations

This document requests no action by IANA.

6. Acknowledgements

The lack of this format was first pointed out by Margaret Wasserman some years ago, and more recently by Kerry Lynn. A previous draft document by Martin Duerst and Bill Fenner [I-D.fenner-literal-zone] discussed this topic but was not finalised.

Valuable comments and contributions were made by Karl Auer, Carsten Bormann, Brian Haberman, Tatuya Jinmei, Tom Petch, Tomoyuki Sahara, Juergen Schoenwaelder, Dave Thaler, and Ole Troan.

Brian Carpenter was a visitor at the Computer Laboratory, Cambridge University during part of this work.

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

7. Change log [RFC Editor: Please remove]

draft-ietf-6man-uri-zoneid-02: additional WG comments, 2012-07-11.

draft-ietf-6man-uri-zoneid-01: use "-" instead of %25, listed alternatives in Appendix, according to WG debate, added suggestion for browser developers, 2012-05-29.

draft-ietf-6man-uri-zoneid-00: adopted by WG, fixed syntax to allow for % encoded characters, 2012-02-17.

draft-carpenter-6man-uri-zoneid-01: chose Option 2, removed 15 character limit, added explanation of ID/number mapping and other clarifications, 2012-02-08.

draft-carpenter-6man-uri-zoneid-00: original version, 2011-12-07.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

8.2. Informative References

- [I-D.fenner-literal-zone]
Fenner, B. and M. Duerst, "Formats for IPv6 Scope Zone Identifiers in Literal Address Formats", draft-fenner-literal-zone-02 (work in progress), October 2005.
- [I-D.iab-identifier-comparison]
Thaler, D., "Issues in Identifier Comparison for Security Purposes", draft-iab-identifier-comparison-02 (work in progress), May 2012.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [chrome] Google, "Use the address bar (omnibox)", 2012, <<http://support.google.com/chrome/bin/answer.py?answer=95440>>.

Appendix A. Alternatives Considered

1. Leave the problem unsolved.

This would mean that per-interface diagnostics would still have to be performed using ping or ping6:

```
ping fe80::a%en1
```

Advantage: works today.

Disadvantage: less convenient than using a browser.

2. Simply using the percent character.

```
http://[fe80::a%en1]
```

Advantage: allows use of browser, allows cut and paste.

Disadvantage: invalid syntax under RFC 3986; not acceptable to

URI community.

3. Escaping the escape character as allowed by RFC 3986:

`http://[fe80::a%25en1]`

Advantage: allows use of browser.

Disadvantage: ugly and confusing, doesn't allow simple cut and paste.

4. Alternative separator

`http://[fe80::a-en1]`

Advantage: allows use of browser, simple syntax

Disadvantage: Requires all IPv6 address literal parsers and generators to be updated in order to allow simple cut and paste.

Note: the initial proposal for this choice was to use an underscore as the separator, but it was noted that this becomes effectively invisible when a user interface automatically underlines URLs.

5. With the "IPvFuture" syntax left open in RFC 3986:

`http://[v6.fe80::a_en1]`

Advantage: allows use of browser.

Disadvantage: ugly and redundant, doesn't allow simple cut and paste.

Authors' Addresses

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland, 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Robert M. Hinden
Check Point Software Technologies, Inc.
800 Bridge Parkway
Redwood City, CA 94065
US

Email: bob.hinden@gmail.com

6MAN
Internet-Draft
Updates: 3986 (if approved)
Intended status: Standards Track
Expires: June 10, 2013

B. Carpenter
Univ. of Auckland
S. Cheshire
Apple Inc.
R. Hinden
Check Point
December 7, 2012

Representing IPv6 Zone Identifiers in Address Literals and Uniform
Resource Identifiers
draft-ietf-6man-uri-zoneid-06

Abstract

This document describes how the Zone Identifier of an IPv6 scoped address, as defined in RFC 4007, can be represented in a literal IPv6 address and in a Uniform Resource Identifier that includes such a literal address. It updates RFC 3986 accordingly.

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 June 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. Specification	4
3. Web Browsers	5
4. Security Considerations	6
5. IANA Considerations	6
6. Acknowledgements	6
7. Change log [RFC Editor: Please remove]	7
8. References	7
8.1. Normative References	7
8.2. Informative References	8
Appendix A. Options Considered	8
Authors' Addresses	10

1. Introduction

The Uniform Resource Identifier (URI) syntax [RFC3986] defined how a literal IPv6 address can be represented in the "host" part of a URI. A subsequent specification [RFC4007] extended the text representation of limited-scope IPv6 addresses such that a zone identifier may be concatenated to a literal address, for purposes described in that RFC. Zone identifiers are especially useful in contexts where literal addresses are typically used, for example during fault diagnosis, when it may be essential to specify which interface is used for sending to a link local address. It should be noted that zone identifiers have purely local meaning within the node where they are defined, often being the same as IPv6 interface names. They are completely meaningless for any other node. Today, they are only meaningful when attached to addresses with less than global scope, but it is possible that other uses might be defined in the future.

RFC 4007 does not specify how zone identifiers are to be represented in URIs. Practical experience has shown that this feature is useful, in particular when using a web browser for debugging with link local addresses, but as it is undefined, it is not implemented consistently in URI parsers or in browsers.

Some versions of some browsers accept the RFC 4007 syntax for scoped IPv6 addresses embedded in URIs, i.e., they have been coded to interpret the "%" sign according to RFC 4007 instead of RFC 3986. Clearly this approach is very convenient for users, although it formally breaches the syntax rules of RFC 3986. The present document defines an alternative approach that respects and extends the rules of URI syntax, and IPv6 literals in general, to be consistent.

Thus, this document updates [RFC3986] by adding syntax to allow a zone identifier to be included in a literal IPv6 address within a URI.

It should be noted that in other contexts than a user interface, a zone identifier is mapped into a numeric zone index or interface number. The MIB textual convention InetZoneIndex [RFC4001] and the socket interface [RFC3493] define this as a 32 bit unsigned integer. The mapping between the human-readable zone identifier string and the numeric value is a host-specific function that varies between operating systems. The present document is concerned only with the human-readable string.

Several alternative solutions were considered while this document was developed. The Appendix briefly describes the various options and their advantages and disadvantages.

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

2. Specification

According to RFC 4007, a zone identifier is attached to the textual representation of an IPv6 address by concatenating "%" followed by <zone_id>, where <zone_id> is a string identifying the zone of the address. However, RFC 4007 gives no precise definition of the character set allowed in <zone_id>. There are no rules or de facto standards for this. For example, the first Ethernet interface in a host might be called %0, %1, %en1, %eth0, or whatever the implementer happened to choose.

In a URI, a literal IPv6 address is always embedded between "[" and "]". This document specifies how a <zone_id> can be appended to the address. Unfortunately "%" is always treated as an escape character in a URI, and according to RFC 3986 it MUST therefore itself be percent-encoded in a URI, in the form "%25". Thus, the scoped address fe80::a%en1 would appear in a URI as http://[fe80::a%25en1].

A <zone_id> SHOULD contain only ASCII characters classified in RFC 3986 as "unreserved". This excludes characters such as "]" or even "%" which would complicate parsing. However, the syntax below does allow such characters to be percent-encoded, for compatibility with existing devices that use them.

If an operating system uses any other characters in zone or interface identifiers that are not in the "unreserved" character set, they MUST be escaped with a "%" sign according to RFC 3986.

We now present the necessary formal syntax.

In RFC 3986, the IPv6 literal format is formally defined in ABNF [RFC5234] by the following rule:

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
```

To provide support for a zone identifier, the existing syntax of IPv6address is retained, and a zone identifier may be added optionally to any literal address. This allows flexibility for unknown future uses. The rule quoted above from RFC 3986 is replaced by three rules:

```
IP-literal = "[" ( IPv6address / IPv6addrz / IPvFuture ) "]"
```

```
ZoneID = 1*( unreserved / pct-encoded )
```

```
IPv6addrz = IPv6address "%25" ZoneID
```

This syntax fills the gap that is described at the end of Section 11.7 of RFC 4007.

The rules in [RFC5952] SHOULD be applied in producing URIs.

RFC 3986 states that URIs have a global scope, but that in some cases their interpretation depends on the end-user's context. URIs including a ZoneID are to be interpreted only in the context of the host where they originate, since the ZoneID is of local significance only.

RFC 4007 offers guidance on how the ZoneID affects interface/address selection inside the IPv6 stack. Note that the behaviour of an IPv6 stack if passed a non-null zone index for an address other than link-local is undefined.

3. Web Browsers

This section discusses how web browsers might handle this syntax extension. Unfortunately there is no formal distinction between the syntax allowed in a browser's input dialogue box and the syntax allowed in URIs. For this reason, no normative statements are made in this section.

Due to the lack of defined syntax, web browsers have been inconsistent in providing for ZoneIDs. Many have no support, but there are examples of ad hoc support. For example, some versions of Firefox allowed the use of a ZoneID preceded by an unescaped "%" character, but this was removed for consistency with RFC 3986. As another example, some versions of Internet Explorer allow use of a ZoneID preceded by a "%" character escaped as "%25", still beyond the syntax allowed by RFC 3986. This syntax extension is in fact used internally in the Windows operating system and some of its APIs.

It is desirable for all browsers to recognise a ZoneID preceded by an escaped "%". In the spirit of "be liberal with what you accept", we also suggest that URI parsers accept bare "%" signs when possible (i.e., a "%" not followed by two valid and meaningful hexadecimal characters). This would make it possible for a user to copy and paste a string such as "fe80::a%en1" from the output of a "ping" command and have it work. On the other hand, "%ee1" would need to be

manually escaped as "fe80::a%25ee1" to avoid any risk of misinterpretation.

Such bare "%" signs are for user interface convenience, and need to be turned into properly escaped characters (where "%25" encodes "%") before the URI is used in any protocol or HTML document. However, URIs including a ZoneID have no meaning outside the originating node. It would therefore be highly desirable for a browser to remove the ZoneID from a URI before including that URI in an HTTP request.

The normal diagnostic usage for the ZoneID syntax will cause it to be entered in the browser's input dialogue box. Thus, URIs including a ZoneID are unlikely to be encountered in HTML documents. However, if they do (for example, in a diagnostic script coded in HTML) it would be appropriate to treat them exactly as above.

4. Security Considerations

The security considerations of [RFC3986] and [RFC4007] apply. In particular, this URI format creates a specific pathway by which a deceitful zone index might be communicated, as mentioned in the final security consideration of RFC 4007. It is emphasised that the format is intended only for debugging purposes, but of course this intention does not prevent misuse.

To limit this risk, implementations MUST NOT allow use of this format except for well-defined usages such as sending to link local addresses under prefix fe80::/10. At the time of writing, this is the only well-defined usage known.

An HTTP client, proxy or other intermediary MUST remove any ZoneID attached to an outgoing URI, as it only has local significance at the sending host.

5. IANA Considerations

This document requests no action by IANA.

6. Acknowledgements

The lack of this format was first pointed out by Margaret Wasserman some years ago, and more recently by Kerry Lynn. A previous draft document by Martin Duerst and Bill Fenner [I-D.fenner-literal-zone] discussed this topic but was not finalised.

Valuable comments and contributions were made by Karl Auer, Carsten Bormann, Benoit Claise, Stephen Farrell, Brian Haberman, Ted Hardie, Tatuya Jinmei, Yves Lafon, Barry Leiba, Radia Perlman, Tom Petch, Tomoyuki Sahara, Juergen Schoenwaelder, Dave Thaler, Martin Thomson, and Ole Troan.

Brian Carpenter was a visitor at the Computer Laboratory, Cambridge University during part of this work.

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

7. Change log [RFC Editor: Please remove]

draft-ietf-6man-uri-zoneid-06: responding to IETF Last Call and IESG comments, 2012-12-07.

draft-ietf-6man-uri-zoneid-05: tuned ABNF, clarified RFC 4007 text, 2012-11-06.

draft-ietf-6man-uri-zoneid-04: additional author, 2012-09-21.

draft-ietf-6man-uri-zoneid-03: reverted to percent-encoded model following WGLC, 2012-09-10.

draft-ietf-6man-uri-zoneid-02: additional WG comments, 2012-07-11.

draft-ietf-6man-uri-zoneid-01: use "-" instead of %25, listed alternatives in Appendix, according to WG debate, added suggestion for browser developers, 2012-05-29.

draft-ietf-6man-uri-zoneid-00: adopted by WG, fixed syntax to allow for % encoded characters, 2012-02-17.

draft-carpenter-6man-uri-zoneid-01: chose Option 2, removed 15 character limit, added explanation of ID/number mapping and other clarifications, 2012-02-08.

draft-carpenter-6man-uri-zoneid-00: original version, 2011-12-07.

8. References

8.1. Normative References

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

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

8.2. Informative References

- [I-D.fenner-literal-zone]
Fenner, B. and M. Duerst, "Formats for IPv6 Scope Zone Identifiers in Literal Address Formats", draft-fenner-literal-zone-02 (work in progress), October 2005.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [chrome] Google, "Use the address bar (omnibox)", 2012, <<http://support.google.com/chrome/bin/answer.py?answer=95440>>.

Appendix A. Options Considered

The syntax defined above allows a ZoneID to be added to any IPv6 address. The 6man WG discussed and rejected an alternative in which the existing syntax of IPv6address would be extended by an option to add the ZoneID only for the case of link-local addresses. It was felt that the present solution offers more flexibility for future uses and is more straightforward to implement.

The various syntax options considered are now briefly described.

1. Leave the problem unsolved.

This would mean that per-interface diagnostics would still have to be performed using ping or ping6:

```
ping fe80::a%en1
```

Advantage: works today.

Disadvantage: less convenient than using a browser.

2. Simply using the percent character.

```
http://[fe80::a%en1]
```

Advantage: allows use of browser, allows cut and paste.

Disadvantage: invalid syntax under RFC 3986; not acceptable to URI community.

3. Escaping the escape character as allowed by RFC 3986:

```
http://[fe80::a%25en1]
```

Advantage: allows use of browser, consistent with general URI syntax.

Disadvantage: somewhat ugly and confusing, doesn't allow simple cut and paste.

This is the option chosen for standardization.

4. Alternative separator

```
http://[fe80::a-en1]
```

Advantage: allows use of browser, simple syntax

Disadvantage: Requires all IPv6 address literal parsers and generators to be updated in order to allow simple cut and paste; inconsistent with existing tools and practice.

Note: the initial proposal for this choice was to use an underscore as the separator, but it was noted that this becomes effectively invisible when a user interface automatically underlines URLs.

5. With the "IPvFuture" syntax left open in RFC 3986:

`http://[v6.fe80::a_en1]`

Advantage: allows use of browser.

Disadvantage: ugly and redundant, doesn't allow simple cut and paste.

Authors' Addresses

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland, 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: cheshire@apple.com

Robert M. Hinden
Check Point Software Technologies, Inc.
800 Bridge Parkway
Redwood City, CA 94065
US

Email: bob.hinden@gmail.com

MBONED Working Group
Internet-Draft
Updates: 4291 (if approved)
Intended status: Standards Track
Expires: November 24, 2012

M. Boucadair, Ed.
France Telecom
J. Qin
Cisco
Y. Lee
Comcast
S. Venaas
Cisco Systems
X. Li
CERNET Center/Tsinghua
University
M. Xu
Tsinghua University
May 23, 2012

IPv6 Multicast Address Format With Embedded IPv4 Multicast Address
draft-ietf-mboned-64-multicast-address-format-02

Abstract

This document specifies an extension to the IPv6 multicast addressing architecture to be used in the context of IPv4-IPv6 interconnection. In particular, this document defines an address format for IPv4-embedded IPv6 multicast addresses. This address format can be used for IPv4-IPv6 translation or encapsulation schemes.

This document updates RFC4291.

Requirements Language

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

Status of this Memo

This Internet-Draft 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 November 24, 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. Introduction	4
2. Terminology	4
3. IPv4-Embedded IPv6 Multicast Address Format: ASM Mode	5
4. IPv4-Embedded IPv6 Multicast Address Format: SSM Mode	6
5. Textual Representation	7
6. Multicast PREFIX64	7
7. Source IPv4 Address in the IPv6 Realm	8
8. Address Translation Algorithm	8
9. Examples	9
10. IANA Considerations	9
11. Security Considerations	9
12. Acknowledgements	9
13. References	10
13.1. Normative References	10
13.2. Informative References	10
Appendix A. Design Choices	11
A.1. Why an Address Format is Needed for Multicast IPv4-IPv6 Interconnection?	11
A.2. Why Identifying an IPv4-Embedded IPv6 Multicast Address is Required?	11
A.3. Location of the IPv4 Address	12
A.4. Location of the M-bit	12
A.5. Encapsulation vs. Translation	14
Authors' Addresses	14

1. Introduction

Various solutions (e.g., [I-D.ietf-softwire-mesh-multicast], [I-D.ietf-softwire-dslite-multicast]) have been proposed to allow access to IPv4 multicast content from hosts attached to IPv6-enabled domains. Even if these solutions have distinct applicability scopes (translation vs. encapsulation) and target different use cases, they all make use of specific IPv6 multicast addresses to embed an IPv4 multicast address. Particularly, the IPv4-embedded IPv6 multicast address is used as a destination IPv6 address of multicast flows received from an IPv4-enabled domain and injected by the IPv4-IPv6 Interconnection Function into an IPv6-enabled domain. It is also used to build an IPv6 multicast state (*, G6) or (S6, G6) corresponding to their (*, G4) or (S4, G4) IPv4 counter parts by the IPv4-IPv6 Interconnection Function. [I-D.ietf-mboned-v4v6-mcast-ps] provides more discussion about issues related to IPv4/IPv6 multicast.

This document specifies an extension to the IPv6 multicast addressing architecture to be used in the context of IPv4-IPv6 interconnection. In particular, this document defines an address format for IPv4-embedded IPv6 multicast addresses. This address format can be used for IPv4-IPv6 translation or encapsulation schemes.

This document updates [RFC4291]. A new M-bit is defined; if set to "1", it indicates an IPv4 multicast address is embedded in the low-order 32 bits of the IPv6 multicast address. Appendix A.1 enumerates the arguments in favor of defining an address format while Appendix A.2 discusses why identifying an IPv4-embedded IPv6 multicast address is needed.

This specification can be used in conjunction with other extensions such as building unicast prefix-based multicast IPv6 address [RFC3306] or embedding the rendezvous point [RFC3956].

This document is a companion document to [RFC6052] which focuses exclusively on IPv4-embedded IPv6 unicast addresses.

Details about design choices are documented in Appendix A.

2. Terminology

This document makes use of the following terms:

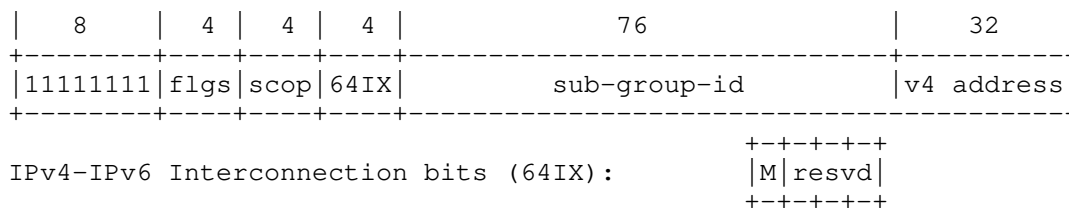
- o IPv4-embedded IPv6 multicast address: denotes a multicast IPv6 address which includes in 32 bits an IPv4 address. Two types of IPv6 addresses are defined that carry an IPv4 address in the low-order 32 bits of the address. The format to build such addresses

is defined in Section 3 for Any Source Multicast (ASM) mode and Section 4 for Source Specific Multicast (SSM) mode.

- o Multicast Prefix64 (or MPREFIX64 for short) refers to an IPv6 multicast prefix to be used to construct IPv4-embedded IPv6 multicast addresses. This prefix is used to build an IPv4-embedded IPv6 multicast address as defined in Section 8. Section 8 specifies also how to extract an IPv4 address from an IPv4-embedded IPv6 multicast address.
- o ASM_MPREFIX64: denotes a multicast Prefix64 used in ASM mode. It follows the format described in Section 3.
- o SSM_MPREFIX64: denotes a multicast Prefix64 used in SSM mode. It follows the format described in Section 4.
- o IPv4-IPv6 Interconnection Function: refers to a function which is enabled in a node interconnecting an IPv4-enabled domain with an IPv6-enabled one. It can be located in various places of the multicast network. Particularly, in terms of multicast control messages, it can be an IGMP/MLD Interworking Function or an IPv4-IPv6 PIM Interworking Function. An IPv4-IPv6 Interconnection Function is configured with one or two MPREFIX64s.

3. IPv4-Embedded IPv6 Multicast Address Format: ASM Mode

To meet the requirements listed in Appendix A.4, the IPv6 multicast address format defined in [RFC4291] is modified to enclose an IPv4 multicast address. Figure 1 shows the modified address format when ASM mode is used.



"resvd" are reserved bits.

Figure 1: IPv4-Embedded IPv6 Multicast Address Format: ASM Mode

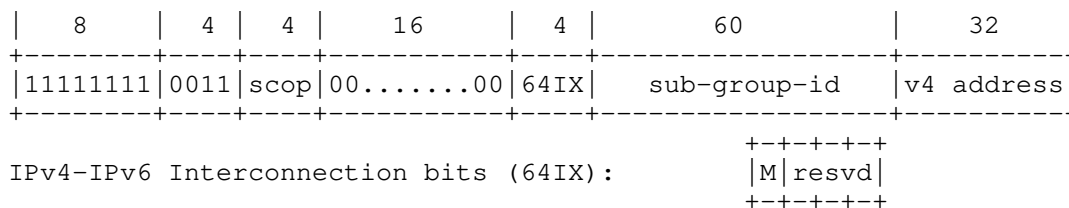
The description of the fields is as follows:

- o "flgs" and "scop" fields are defined in [RFC4291].

- o 64IX field (IPv4-IPv6 interconnection bits): The first bit is the M-bit. When "M-bit" is set to 1, it indicates that a multicast IPv4 address is embedded in the low-order 32 bits of the multicast IPv6 address. All the remaining bits are reserved and MUST be set to 0.
- o sub-group-id: This field is configurable according to local policies (e.g., enable embedded-RP) of the entity managing the IPv4-IPv6 Interconnection Function. This field MUST follow the recommendations specified in [RFC3306] if unicast-based prefix is used or the recommendations specified in [RFC3956] if embedded-RP is used. The default value is all zeros.
- o The low-order 32 bits MUST include an IPv4 multicast address when the M-bit is set to 1. The enclosed IPv4 multicast address SHOULD NOT be in 232/8 range.

4. IPv4-Embedded IPv6 Multicast Address Format: SSM Mode

As mentioned in Appendix A.4, any IPv4-embedded IPv6 address used in SSM mode MUST be part of ff3x::/32 [RFC4607]. Figure 2 describes the format of the IPv6 multicast address to be used to enclose an IPv4 multicast address.



"resvd" are reserved bits.

Figure 2: IPv4-Embedded IPv6 Multicast Address Format: SSM Mode

The description of the fields is as follows:

- o Flags MUST be set to 0011.
- o "scop" is defined in [RFC4291].
- o 64IX field (IPv4-IPv6 interconnection bits): Same meaning as Section 3.
- o sub-group-id: The default value is all zeros.
- o The low-order 32 bits MUST include an IPv4 multicast address when the M-bit is set to 1. The embedded IPv4 address SHOULD be in the 232/8 range [RFC4607].

5. Textual Representation

The embedded IPv4 address in an IPv6 multicast address is included in the last 32 bits; therefore dotted decimal notation can be used.

6. Multicast PREFIX64

For the delivery of the IPv4-IPv6 multicast interconnection services, a dedicated multicast prefix denoted as MPREFIX64 should be provisioned (e.g., using NETCONF or [I-D.ietf-softwire-multicast-prefix-option]) to any function requiring to build an IPv4-embedded IPv6 multicast address based on an IPv4 multicast address. MPREFIX64 can be of ASM or SSM type. When both modes are used, two prefixes are required to be provisioned.

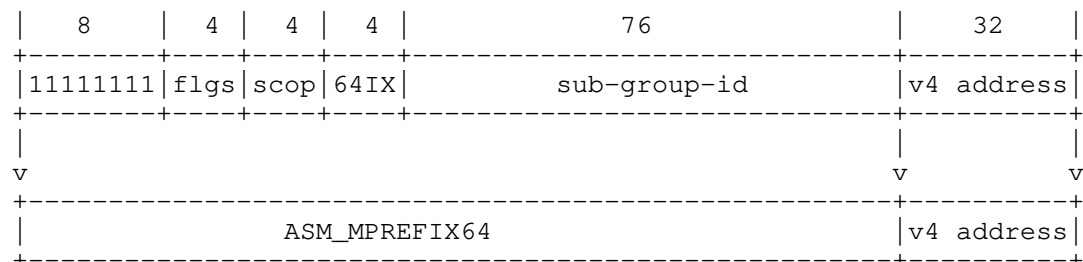
The structure of the MPREFIX64 follows the guidelines specified in Section 3 for the ASM mode and Section 4 when SSM mode is used.

The length of MPREFIX64 MUST be /96 (as shown in Figure 3).

The format of the MPREFIX64 should follow what is specified in [RFC3306] and [RFC3956] if corresponding mechanisms are used.

The format specified in Section 3 uses some reserved bits defined in [RFC3306] and [RFC3956]: the first of the reserved bits now has a meaning, while the remaining bits MUST be set to 0.

ASM Mode:



SSM Mode:

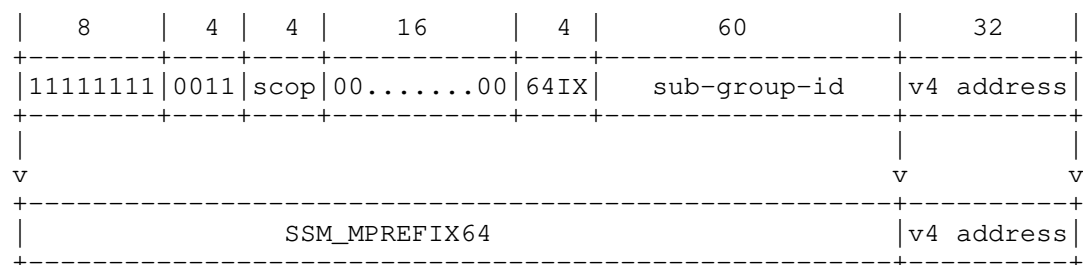


Figure 3: MPREFIX64

7. Source IPv4 Address in the IPv6 Realm

An IPv4 source is represented in the IPv6 realm with its IPv4-converted IPv6 address [RFC6052].

8. Address Translation Algorithm

IPv4-embedded IPv6 multicast addresses are composed according to the following algorithm:

- o Concatenate the MPREFIX64 and the 32 bits of the IPv4 address to obtain a 128-bit address.

The IPv4 multicast addresses are extracted from the IPv4-embedded IPv6 multicast addresses according to the following algorithm:

- o If the multicast address belongs to ff3x:0:8000/33 or ffx::8000/17, extract the last 32 bits of the IPv6 multicast address.

9. Examples

Figure 4 provides an example of ASM IPv4-Embedded IPv6 Address while Figure 5 provides an example of SSM IPv4-Embedded IPv6 Address.

MPREFIX64	IPv4 address	IPv4-embedded IPv6 address
ffxx:8000:abc::/96	224.1.2.3	ffxx:8000:abc::224.1.2.3

Figure 4: Example of ASM IPv4-embedded IPv6 address

MPREFIX64	IPv4 address	IPv4-embedded IPv6 address
ff3x:0:8000::/96	232.1.2.3	ff3x:0:8000::232.1.2.3

Figure 5: Example of SSM IPv4-embedded IPv6 address

10. IANA Considerations

Authors of this document request to reserve:

- o ff3x:0:8000/33 SSM block to embed an IPv4 multicast address in the last 32 bits.
- o ffxx:8000/17 ASM block to embed an IPv4 multicast address in the last 32 bits.

11. Security Considerations

This document defines an address format to embed an IPv4 multicast address in an IPv6 multicast address. The same security considerations as those discussed in [RFC6052] are to be taken into consideration.

12. Acknowledgements

Many thanks to R. Bonica, B. Sarikaya, P. Savola, T. Tsou and C. Bormann for their comments and review.

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.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, November 2004.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

13.2. Informative References

- [I-D.ietf-behave-nat64-learn-analysis]
Korhonen, J. and T. Savolainen, "Analysis of solution proposals for hosts to learn NAT64 prefix", draft-ietf-behave-nat64-learn-analysis-03 (work in progress), March 2012.
- [I-D.ietf-mboned-v4v6-mcast-ps]
Jacquenet, C., Boucadair, M., Lee, Y., Qin, J., Tsou, T., and Q. Sun, "IPv4-IPv6 Multicast: Problem Statement and Use Cases", draft-ietf-mboned-v4v6-mcast-ps-00 (work in progress), May 2012.
- [I-D.ietf-software-dslite-multicast]
Qin, J., Boucadair, M., Jacquenet, C., Lee, Y., and Q. Wang, "Multicast Extensions to DS-Lite Technique in Broadband Deployments", draft-ietf-software-dslite-multicast-02 (work in progress), May 2012.
- [I-D.ietf-software-mesh-multicast]
Xu, M., Cui, Y., Yang, S., Wu, J., Metz, C., and G. Shepherd, "Software Mesh Multicast", draft-ietf-software-mesh-multicast-02 (work in progress), April 2012.

- [I-D.ietf-softwire-multicast-prefix-option]
Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6
Option for IPv4-Embedded Multicast and Unicast IPv6
Prefixes", draft-ietf-softwire-multicast-prefix-option-00
(work in progress), March 2012.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", RFC 4566, July 2006.

Appendix A. Design Choices

A.1. Why an Address Format is Needed for Multicast IPv4-IPv6 Interconnection?

Arguments why an IPv6 address format is needed to embed multicast IPv4 address are quite similar to those of [RFC6052]. Concretely, the definition of a multicast address format embedding a multicast IPv4 address allows:

- o Stateless IPv4-IPv6 header translation of multicast flows;
- o Stateless IPv4-IPv6 PIM interworking function;
- o Stateless IGMP-MLD interworking function (e.g., required for an IPv4 receiver to access to IPv4 multicast content via an IPv6 network);
- o Stateless (local) synthesis of IPv6 address when IPv4 multicast address are embedded in application payload (e.g., SDP [RFC4566]);
- o Except the provisioning of the same MPREFIX64, no coordination is required between IPv4-IPv6 PIM interworking function, IGMP-MLD interworking function, IPv4-IPv6 Interconnection Function and any ALG (Application Level Gateway) in the path;
- o Minimal operational constraints on the multicast address management: IPv6 multicast addresses can be constructed using what has been deployed for IPv4 delivery mode.

A.2. Why Identifying an IPv4-Embedded IPv6 Multicast Address is Required?

Reserving an M-bit in the IPv6 multicast address (which is equivalent to reserving a dedicated multicast block for IPv4-IPv6 interconnection purposes) is a means to guide the address selection process at the receiver side; in particular it assists the receiver to select the appropriate IP multicast address while avoiding to involve an IPv4-IPv6 interconnection function in the path.

Two use cases to illustrate this behavior are provided below:

1. An ALG is required to help an IPv6 receiver to select the appropriate IP address when only the IPv4 address is advertised (e.g., using SDP); otherwise the access to the IPv4 multicast content can not be offered to the IPv6 receiver. The ALG may be located downstream the receiver. As such, the ALG does not know in advance whether the receiver is dual-stack or IPv6-only. The ALG may be tuned to insert both the original IPv4 address and corresponding IPv6 multicast address. If the M-bit is not used, a dual-stack receiver may prefer to use the IPv6 address to receive the multicast content. This address selection would require multicast flows to cross an IPv4-IPv6 interconnection function.
 2. In order to avoid involving an ALG in the path, an IPv4-only source can advertise both its IPv4 address and IPv4-embedded IPv6 multicast address. If the M-bit is not used, a dual-stack receiver may prefer to use the IPv6 address to receive the multicast content. This address selection would require multicast flows to cross an IPv4-IPv6 interconnection function.
- Reserving an M-bit in the IPv6 multicast address for IPv4-IPv6 interconnection purposes mitigates the issues discussed in [I-D.ietf-behave-nat64-learn-analysis] in a multicast context.

A.3. Location of the IPv4 Address

There is no strong argument to allow for flexible options to encode the IPv4 address inside the multicast IPv6 address. The option retained by the authors is to encode the multicast IPv4 address in the low-order 32 bits of the IPv6 address.

This choice is also motivated by the need to be compliant with [RFC3306] and [RFC3956].

A.4. Location of the M-bit

Figure 6 is a reminder of the IPv6 multicast address format as defined in [RFC4291]:

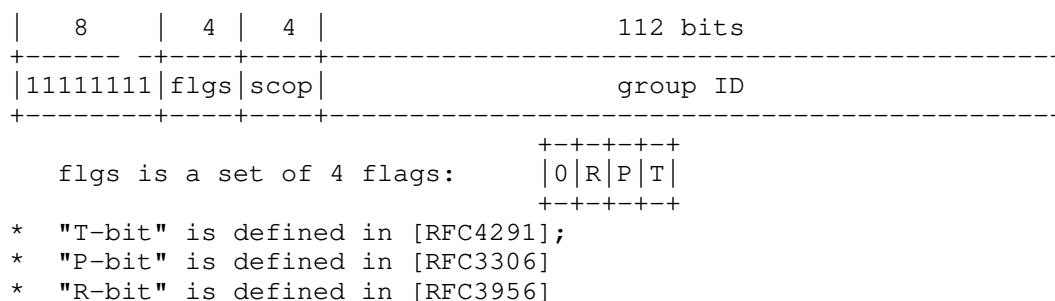


Figure 6: IPv6 Multicast address format as defined in RFC4291

It was tempting to use the remaining flag to indicate whether an IPv6 address embeds an IPv4 address or not. This choice has been abandoned by the authors for various reasons:

- o ff3x::/32 is defined as SSM. Defining a new flag would require standards and implementations to also treat ffbx::/32 as SSM.
- o Prefixes starting with ff7x are defined as embedded-RP, but not prefixes starting with fffx. Below is provided an excerpt from [RFC3956]:

"...the encoding and the protocol mode used when the two high-order bits in "flgs" are set to 11 ("fff0::/12") is intentionally unspecified until such time that the highest-order bit is defined. Without further IETF specification, implementations SHOULD NOT treat the fff0::/12 range as Embedded-RP."

as such defining a new flag would require implementations to also treat ff7x::/12 as embedded-RP prefix.

- o This is the last remaining flag and at this stage we are not sure whether there is other usage scenarios of the flag.

As a conclusion, the remaining flag is not used to indicate an IPv6 multicast address embeds an IPv4 multicast address. However the following constraints should be met:

- (1) Belong to ff3x::/32 and be compatible with unicast-based prefix [RFC3306] for SSM. Note that [RFC3306] suggests to set "plen" to 0 and "network-prefix" to 0.
- (2) Be compatible with embedded-RP [RFC3956] and unicast-based prefix [RFC3306] for ASM;
- (3) Avoid ff3x::4000:0001-ff3x::7fff:ffff which is reserved for IANA.

Meeting (1) and (2) with the same location of the M-bit is not feasible without modifying embedded-RP and unicast-based prefix specifications; this option is avoided.

As a consequence, two multicast blocks are proposed to be used when

embedding IPv4 address: one block for ASM (Section 3) and another one for the SSM (Section 4).

A.5. Encapsulation vs. Translation

IPv4-IPv6 encapsulator and translator may be embedded in the same device or even implemented with the same software module. In order to help the function whether an encapsulated IPv6 multicast packets or translated IPv6 ones are to be transferred. It was tempting to define an S-bit for that purpose but this choice has been abandoned in favor of the recommendation to use distinct MPREFIX64 for each scheme.

As such, there is no need to reserve a bit in the IPv6 multicast address to separate between the translation and the encapsulation schemes.

Authors' Addresses

Mohamed Boucadair (editor)
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange.com

Jacni Qin
Cisco
China

Email: jacni@jacni.com

Yiu L. Lee
Comcast
U.S.A

Email: yiu_lee@cable.comcast.com

Stig Venaas
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: stig@cisco.com

Xing Li
CERNET Center/Tsinghua University
Room 225, Main Building, Tsinghua University
Beijing, 100084
P.R. China

Phone: +86 10-62785983
Email: xing@cernet.edu.cn

Mingwei Xu
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing, 100084
P.R.China

Phone: +86-10-6278-5822
Email: xmw@cernet.edu.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 2012

H. Kitamura
NEC Corporation
S. Ata
Osaka City University
June 24, 2012

Corresponding Auto Names for IPv6 Addresses
<draft-kitamura-ipv6-auto-name-02.txt>

Abstract

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses. With this mechanism, all IPv6 addresses (even if they are link-local scoped addresses) can obtain their own Names, and it will be able to use Names anywhere instead of IPv6 Addresses.

IPv6 address is too long and complicated to remember, and it is very nuisance to type a literal IPv6 address manually as an argument of applications. Also, it is very difficult for human beings to tell which IPv6 address is set to which actual IPv6 node. In this sense, literal IPv6 address information can be called meaningless information for human beings.

In order to solve above problems and to provide annotated meaningful information to IPv6 addresses, mechanisms called Corresponding Auto Names for IPv6 addresses is introduced. They will become human friendly information. By applying a simple naming rule to the Auto Names (e.g., use the same Auto Name Suffix for IPv6 addresses that are set to the same interface (node)), this will contribute to help people to understand which IPv6 address / Name indicates which actual IPv6 node and provide meaningful information.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on December 2012.

Copyright Notice

Copyright (c) 2011 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.

Table of Contents

1. Introduction	3
2. Goals (What can be achieved)	4
2.1. Assumed typical IPv6 communication environment:	4
2.2. Auto Names examples	4
2.3. Auto Name Suffix for Grouped Addresses	5
2.4. Contribution in Regular Resolving (Name -> Address)	6
2.5. Contribution in Reverse Resolving (Address -> Name)	6
3. Deployed Notions and Functions that are used in Auto Names	8
3.1. Stateless Name	8
3.2. Scoped Name	8
3.3. Target IPv6 Addresses	10
4. Design of Auto Names	11
4.1. Conceptual Design on Naming Rules	11
4.1.1. <P> Value:	11
4.1.2. <I> Value:	12
4.1.3. <NGI> Value:	12
4.2. Address Type Distinction	13
4.2.1. EUI64-based Address Identification	13
4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction	13
4.3. IPv6 Address Appearance Detection and Auto Name Registration	15
4.3.1. IPv6 Address Appearance Detection mechanism	15
4.3.2. Auto Names Generation and Registration mechanism	15
4.3.3. Placement of Detector and Registrar	16
4.3.4. Detection and Registration Procedures	17
5. Security Considerations	18
6. IANA Considerations	18
Appendix A. Implementation	18
References	18
Authors' Addresses	19

1. Introduction

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses.

IPv6 address is too long and complicated to remember. For human being, values of EUI64-based or temporary addresses should be felt that they are random number series. So, it is very nuisance to type a literal IPv6 address manually as an argument of applications.

Furthermore, it is very normal and popular cases to set multiple IPv6 addresses to one node. One IPv6 node owns more than two IPv6 addresses (typically: one is link-local scoped address. the other is global scoped address) at least. Some IPv6 addresses (such as link-local scoped stateless auto-configuration addresses and temporary addresses) may become users' conscious-less address, because they are automatically set to the IPv6 node.

It is too difficult for human beings to tell which IPv6 address is set to which IPv6 node. In other words, when an IPv6 address is shown to a person, he almost can not tell that the shown IPv6 address indicates which IPv6 node. In this sense, literal IPv6 address information can be called useless or meaningless information for human beings.

Moreover, when more than two literal IPv6 addresses are shown to human beings, it is almost impossible to distinguish them whether they are same or not at a glance.

So, there are strong desires to use Name information (that is human friendly) instead of literal IPv6 Address information and to use meaningful and distinguishable information that can easily show which IPv6 address / Name indicates which actual IPv6 node.

The Corresponding Auto Names for IPv6 Addresses is introduced to solve above problems and to satisfy the above desires.

2. Goals (What can be achieved)

In this section, goals of the mechanisms of the Corresponding Auto Names for IPv6 Addresses and what can be achieved are shown by using examples.

2.1. Assumed typical IPv6 communication environment:

Two IPv6 nodes (Node A and Node B) are located on the same link. Their IPv6 Addresses are shown below.

Node A:	Literal Address

MAC Address:	00:0d:5e:b8:80:7b

LL-Address:	fe80::20d:5eff:feb8:807b%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b
	fd01:2345:6789::1234
Global Addr:	2001:db8::20d:5eff:feb8:807b
	2001:db8::1234
Node B:	Literal Address

MAC Address:	00:0c:76:d9:14:e3

LL-Address:	fe80::20c:76ff:fed9:14e3%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3
	fd01:2345:6789::5678
Global Addr:	2001:db8::20c:76ff:fed9:14e3
	2001:db8::5678

They own altogether 5 IPv6 addresses respectively;

One Link-Local scoped Address

Two Unique Local Addresses (SLLAC and manual set address)

Two Global scoped Addresses (SLLAC and manual set address)

They communicate each other.

2.2. Auto Names examples

For all addresses, respective Corresponding Auto Names are prepared and registered to a some name resolving service DB (typically the DNS is used for this) automatically by the mechanism that detects these addresses (that is explained after in this document).

Prepared Auto Names are shown below.

Node A:	Literal Address	Auto Name
MAC Address:	00:0d:5e:b8:80:7b	
LL-Address:	fe80::20d:5eff:feb8:807b%fxp0	-> L0-7bz%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b	-> U0-7bz
	fd01:2345:6789::1234	-> U1-7bz
Global Addr:	2001:DB8::20d:5eff:feb8:807b	-> G0-7bz
	2001:DB8::1234	-> G1-7bz
Node B:	Literal Address	Auto Name
MAC Address:	00:0c:76:d9:14:e3	
LL-Address:	fe80::20c:76ff:fed9:14e3%em0	-> L0-3ez%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3	-> U0-3ez
	fd01:2345:6789::5678	-> U1-3ez
Global Addr:	2001:DB8::20c:76ff:fed9:14e3	-> G0-3ez
	2001:DB8::5678	-> G1-3ez

2.3. Auto Name Suffix for Grouped Addresses

In order to make Auto Names meaningful, IPv6 addresses are grouped and Auto Name Suffix is used to show grouped addresses.

For IPv6 addresses that are set to the same interface (node), the same Auto Name Suffix that stands for the Group ID is used for their Auto Names.

As shown above:

'-7bz' is used for Auto Name Suffix (Group ID) for Node A.
'-3ez' is used for Auto Name Suffix (Group ID) for Node B.

In order to make easier to identify and remember the Auto Name Suffixes, their naming rule is based on inheriting the last octet of the node's MAC address in this example.

2.4. Contribution in Regular Resolving (Name -> Address)

In order to communicate with the specific IPv6 address of the destination node, the following procedure to type literal IPv6 address is required in the current environment. They are very stressful and nuisance procedures for human beings.

When 'ping6' or 'telnet' to the specific IPv6 address of Node B from Node A is executed, the following commands are typed.

```
>ping6 fe80::20c:76ff:fed9:14e3%fxp0
>telnet fd01:2345:6789::20c:76ff:fed9:14e3
```

Especially for link-local scoped addresses or temporary addresses, there are no way to type Names instead of literal IPv6 addresses, because they are generally not registered to name resolving services.

By introducing the Corresponding Auto Names, above typed commands are changed and replaced with the following easy and rememberable name typing procedures.

```
>ping6 L0-3ez%fxp0
>telnet U0-3ez
```

2.5. Contribution in Reverse Resolving (Address -> Name)

Communication related status information is shown to human beings in literal IPv6 address format in the current environment.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
fe80::20d:5eff:feb8:807b.8722	fe80:3::20c:76ff:fed9:14e3.23	ESTABLISH
fd01:2345:6789::1234.16258	fd01:2345:6789::5678.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
fe80::20d:5eff:feb8:807b%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
fe80::221:85ff:fea7:82ff%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
fe80::20c:76ff:fed9:14e3%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
fd01:2345:6789::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
fd01:2345:6789::5678	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
2001:DB8::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
2001:DB8::5678	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

People almost can not tell which shown literal IPv6 address indicates which IPv6 node. In this sense, shown information is meaningless and useless.

By introducing the Corresponding Auto Names, above complicated information is converted into simple and meaningful information and shown as followed.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
L0-7bz.8722	L0-e3z.23	ESTABLISH
U0-7bz.16258	U0-e3z.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
L0-7bz%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
U0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
U1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
L0-ffz%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
L0-3ez%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
U0-3ez	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
U1-3ez	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
G0-3ez	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
G1-3ez	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

Other examples where the Auto Name technique can contribute:

In log files of a server application, accesses from clients are recorded into them in literal IPv6 address format. It is almost impossible to read and understand the log files effectively without this Auto Name technique.

Also, in packet dumping applications, address information is shown in literal IPv6 address format. This Auto Name technique can significantly help for human beings to analyze and understand dumped packets.

Shown communication related status information in Auto Name format is simple and easy enough for human beings to understand. As shown above, troublesome IPv6 literal Address information can be converted into meaningful and distinguishable information by using the Corresponding Auto Names technique, and we can achieve our goals.

3. Deployed Notions and Functions that are used in Auto Names

3.1. Stateless Name

We know that we can categorize Addresses into two types. One is "stateful" address type, and the other is 'stateless' address type.

On the other, we have not been applied the same categorization to domain Names or host Names clearly. It has been assumed that existing all Names are categorized into stateful type and there is no stateless name type. Authors think that it is a time to change this preconception.

We can grasp that the introduced Corresponding Auto Name is realization of "stateless" name type, and we have deployed a notion Stateless Name clearly here.

Table 1 Stateless Name

	Stateful	Stateless
Address	DHCPv6	SLAAC
Name	existing Domain Names	Auto Names

3.2. Scoped Name

We also know that a notion called "scope" (such as link-local scope, global-scope) is introduced when we deal with addresses. Every address has its own scope.

In domain names or host names cases, the "scope" notion have NOT clearly introduced now. It is generally assumed that all names are global information and "scope" notion does not exist.

The Corresponding Auto Name is achieved by introducing Scoped Name obviously.

Scope of Auto Name for IPv6 address is the same to the scope of its IPv6 address. For example, scope of the Auto Name for the link-local IPv6 address is link-local. They are only effective within the link-local scope.

Table 1 Scoped Name

	Global	Site-Local (ULA)	Link-Local	Node-Local
Address	2001:db8::/64	fd01:2345:6789::/64	fe80::/64	
Nmae	Domain Names	Domain Names / Auto Names	Auto Names	Auto Names

At some special situation (that it is enough that Auto Name information is shared within a node), Node-Local scoped Auto Name is possible. At such a situation, we can use /etc/hosts file as a name service.

Deployment of Scoped Name:

Scoped Name notion can be easily achieved with current technologies.

As shown above, at a special case when we adopt /etc/hosts file as a name service for Auto Names, scope of Auto Names naturally becomes Node-Local.

At general cases when we adopt the DNS as a name service for Auto Names, scope of Auto Names is easily managed by the DNS query access permission control on DNS servers.

3.3. Target IPv6 Addresses

One of the goals of the Auto Name technique is to provide and set Names to all IPv6 addresses (include Link-local addresses).

All IPv6 Addresses are targets of Auto Names.

Some IPv6 Addresses have their own names (let's call them Proper Names) that are assigned manually and are registered into name resolving services (such as the DNS). It may be thought that it is not necessary to assign Auto Name to such IPv6 addresses which have Proper Names. However, we strongly recommend assigning Auto Names to them, too.

In order to provide meaningful information to IPv6 addresses, uniformed Name information for all IPv6 addresses is necessary. This is very natural behavior for 'Stateless' type technology.

Since one-to-multiple mapping is allowed in name resolving services, it will not cause problems to assign both Proper Name and Auto Name for one IPv6 address.

We may need some function that controls name display priority (which name is first Proper Name or Auto Name). This function also achieved easily by using existing current technologies.

If Auto Name and Proper Name are implemented as different name resolving services (e.g., one is /etc/hosts, the other is the DNS), name display priority is can be easily controlled by nsswitch.conf function.

If Auto Name and Proper Name are implemented as same name resolving service, name display priority is can be controlled by their registration order to the name resolving service DB.

4. Design of Auto Names

4.1. Conceptual Design on Naming Rules

Auto Names are composed of "<P><I>--<NGI>" format:

<P>: stands for Prefix of Address

1 character: (e.g., 'L', 'U', 'G')

<I>: stands for Interface ID of Address

1 character: (e.g., '0', '1')

<NGI>: stands for Node (Interface) Group ID

3 characters:
(e.g., '7bz', '3ez')

Above discussed Auto Name examples satisfy <P><I>--<NGI> format.

on Node A: L0-7bz, U0-7bz, U1-7bz, G0-7bz, G1-7bz

on Node B: L0-3ez, U0-3ez, U1-3ez, G0-3ez, G1-3ez

4.1.1. <P> Value:

<P> value stands for Prefix (Scope) (upper 64 bit) of Address as 1 character format.

Auto Names of IPv6 addresses whose prefixes are same use the same <P> value.

Typically, following characters are used for <P> value:

"L": used for Link-local scoped addresses.

"U": used for ULA

"G": used for Global scoped address

If multiple prefixes for the same scope are used, other character (such as "H", "I",,,) can be used depending on the circumstances.

"Prefix - <P> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "Prefix - <P> value" mapping table) is used to avoid collision and manage mappings of them.

4.1.2. <I> Value:

<I> value stands for Interface ID (lower 64 bit) of Address as 1 character format.

Following characters are used for <I> value:

<I> value assignment is based on three address type categorization.

"0": used for EUI64-based address
"1", "9": used for manually set addresses
(stateful addresses will be categorized here)

"a", "z": used for automatically generated and set addresses
except EUI64-based
(Temporary addresses are categorized here)

4.1.3. <NGI> Value:

<NGI> value is also called Auto Name-Suffix.

In order to make IPv6 addresses meaningful, IPv6 addresses are grouped. It is very natural to group IPv6 addresses by which node (interface) they are set. So, IPv6 addresses that are set to the same node (interface) are grouped into the same group.

<NGI> value is shown as 'XYZ' format:

'XY': (1st, 2nd chars) are inherited from
the last octet (2 characters) of the node's MAC address
'Z' : (3rd char) suffix char to avoid a collision of 'XY'
starting from "z"
if 'XY' is collided, 'Z' is changed into "y", "x" , , ,

By using the birthday paradox theorem, collision probability of 256 states (1 octet) is calculated. If 19 nodes (interfaces) exist on the same link, collision is happened with 50% probability. Collision check procedure of the last octet of MAC addresses is necessary.

"MAC address - <NGI> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "MAC address - <NGI> value" mapping table) is needed to avoid collision and manage mappings of them.

Detailed methods how to distinguish and categorize IPv6 addresses are described at the following section.

We can found one remarkable thing with the naming rules:

"L0-XYZ" is a special and very standard Auto Name. "L0-XYZ" is an Auto Name that assigned for Link-local scoped EUI64-based address. Since Almost all the IPv6 nodes have Link-local scoped EUI64-based address, with "L0-XYZ" name information we can reach or communicate the node whose last octet MAC address is known as "XY".

4.2. Address Type Distinction

If we can obtain address type information from Auto Name, convenient environment will be provided. So, there are strong desires to understand type of detected IPv6 addresses. In this section, methods how to distinguish and categorize IPv6 addresses are described.

4.2.1. EUI64-based Address Identification

Only with IPv6 address information, it is impossible to identify that the detected IPv6 address is EUI64-based address or not.

In proposed IPv6 address detection method which is described in the following section, IPv6 address and MAC address that are set to the same node (interface) is detected simultaneously.

So, with this detection method, it is very easy to identify that the detected IPv6 address is EUI64-based address or not.

4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction

It is generally difficult to distinguish whether the detected IPv6 address is manually or automatically set address.

In order to distinguish IPv6 address types, "Zero Contain Rate" technique is introduced.

For a human being, "Zero" is a special value. When a human being omits a part of information, "Zero" is used for the omitted part of information implicitly.

For a machine "Zero" is NOT a special value. "Zero" is treated as almost equal to other values.

We can reach a fact that manually set IPv6 address contains many "Zero", because it is assigned by a human being. 64bit is long for

a human being and there must be too many omitted part filled with "Zero". Especially, a human being is apt to omit and fill with "Zero" upper part of 64bit Interface ID.

In other words, we can see human relation bias from "Zero Contain Rate" of IPv6 address.

Bias Check by using Mathematical Technique:

By using mathematical probability technique, we can distinguish whether value of 64bit Interface ID is biased (manual) or non-biased (automatic).

When we see 64bit value in 8bit (1 octet) unit, it follows the binomial distribution ($n=8$ and $p=1/256$).

Under this distribution, the probability to meet "Zero" octet two times is 0.042%. It means that to meet "Zero" octet two times is too rare case if the value is non-biased.

So, we can adopt the following method:

If the value of 64bit Interface ID contains two and above "Zero" octets,
the value is bias and
the IPv6 address is identified as a manually set address.

Of cause, this method is not perfect because this is based on mathematical probability technique and heuristic human behavior, but this is very effective.
Even if wrong identification is done, no big problems are found.

4.3. IPv6 Address Appearance Detection and Auto Name Registration

In order to generate and register Auto Names automatically, two types of mechanisms are needed. One is a mechanism that detects IPv6 address appearance. The other is a mechanism that checks the detected addresses and generates Auto Names and registers them to name service.

Two functions ("Detector" and "Registrar") are introduced. "Detector" function takes in charge of the former mechanism, and "Registrar" takes in charge the latter mechanism.

4.3.1. IPv6 Address Appearance Detection mechanism

In order to detect newly appeared IPv6 address, DAD message (NS for DAD) is effectively used.

DAD message has the following good capabilities:

- issued only when node would like to set new IPv6 address
- issued for All types (link-local, global, temporary,,,))
- L2 broadcast and easy to capture (without using mirror port)
- distinguishable from other NS messages, because source address of the message is unspecified ("::") and different from others
- Captured DAD message includes all necessary information (such as, IPv6 address and MAC address)

Detector captures DAD messages and detects newly appeared IPv6 addresses. Detected information is sent to Registrar.

4.3.2. Auto Names Generation and Registration mechanism

At first, Registrar checks the Detected address information that is sent from Detector(s). By using the reverse resolving (Address -> Name), it is checked whether the Detected address information is first appearance or not. If an entry for the address does NOT exist, it is confirmed that the address is first appearance and it should be registered to the name server.

After Name for the address is prepared, duplication of the Name can be checked by using the regular resolving (Name -> Address). If an entry for the Name exist, it is confirmed that Name is duplicated (collided). Another Name is prepared and checked again until the Name is not duplicated.

Finally, Registrar registers both Regular and Reverse resolving entries for the address and prepared Auto Name are registered to the name server.

4.3.3. Placement of Detector and Registrar

Placement of Detector and Registrar is designed to make the mechanisms flexible and to make it to be applied to various environments (office networks, home networks, etc.)

Figure 1 and 2 show typical examples that indicate locations where Detector and Registrar functions are placed on the IPv6 network. Figure 1 shows a case for a single link, and Figure 2 shows a case for multiple links.

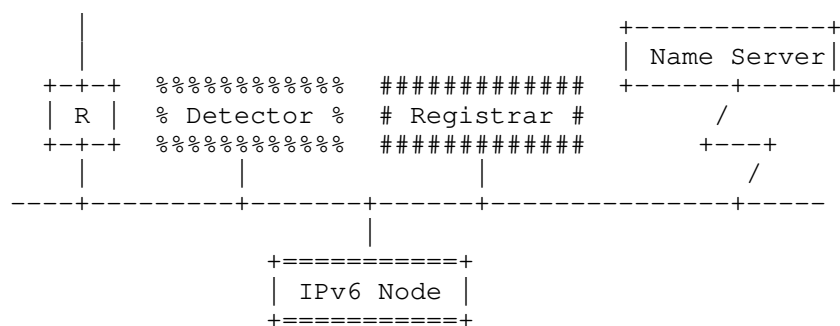


Fig. 1 Single-Link Case Example

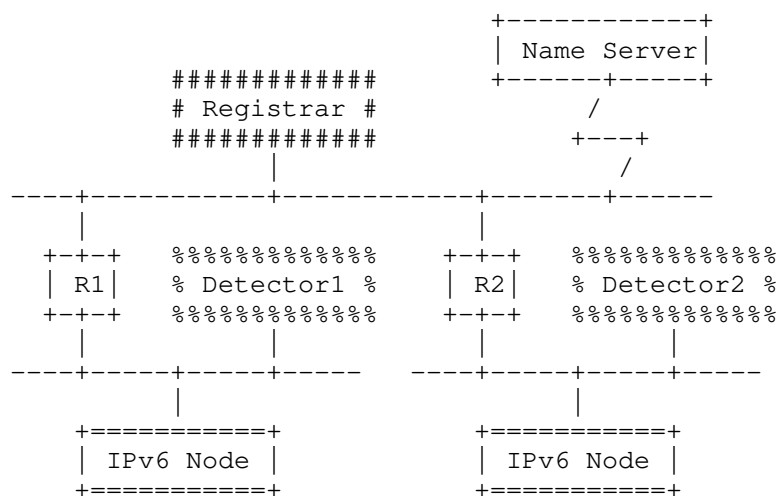
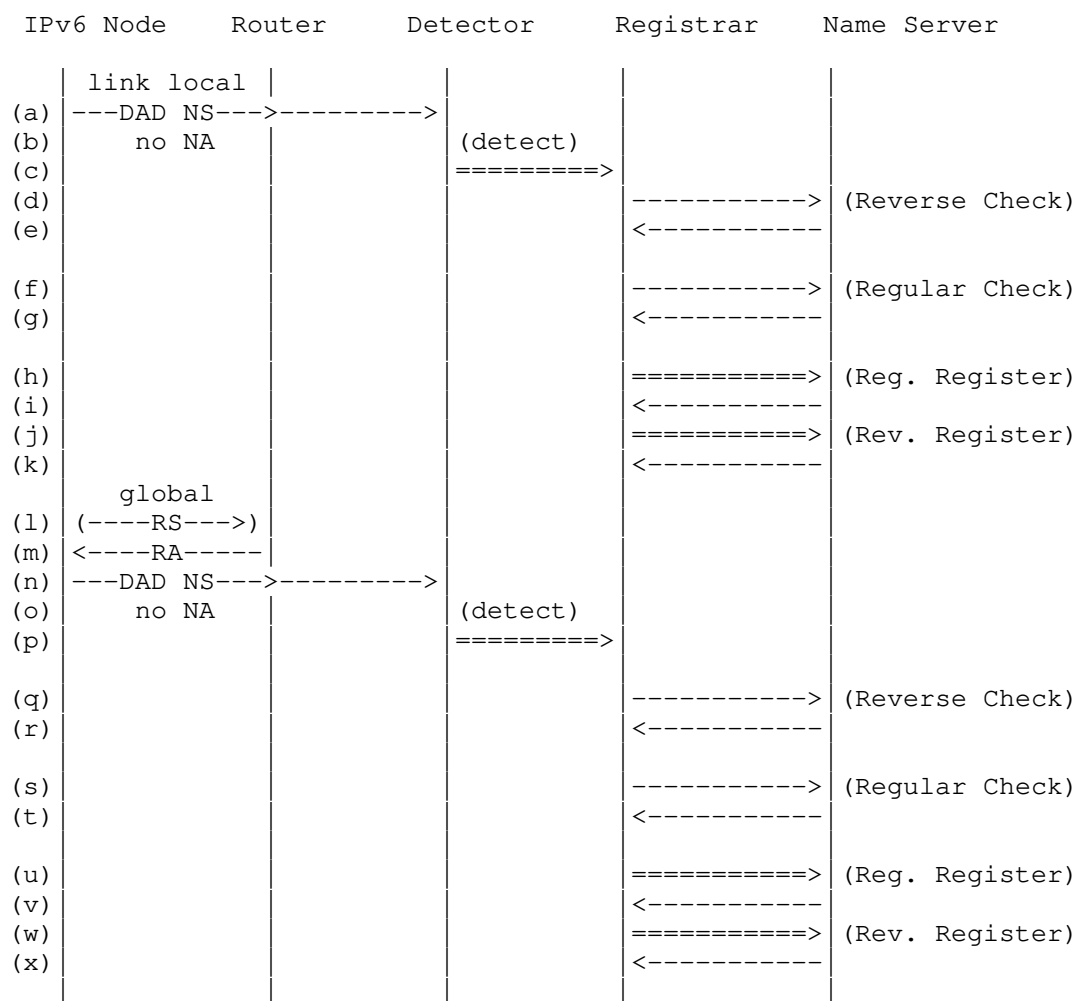


Fig. 2 Multiple-Link Case Example

4.3.4. Detection and Registration Procedures

Figure 3 shows an example of typical detection and registration procedures at IPv6 links where DAD packets are issued. DAD message packets are used for the appearance detection.



5. Security Considerations

Auto Names are generated and registered to the name service in this document. In order to register correct Auto Names information, communication between Detector and Registrar and communication between Registrar and Name Server should be protected and be secured.

In general usage, scope of Auto Names will be local (not global). Auto Names are usually local scoped names. So, we do not have to be too sensitive on the correctness of Auto Names.

6. IANA Considerations

This document does not require any resource assignments to IANA.

Appendix A. Implementation

Auto Name functions have been implemented at the following environments. It has been verified that designed functions work well.

Used functions:

Packet capture on Detector:	libpcap
Name Server:	DNS (BIND9)
Name Registration:	nsupdate (BIND9 bundled)

OS:

FreeBSD 6.2R
(Since FreeBSD OS specific functions are not used to implement, codes will run on UNIX type OS that has libpcap (such as Linux).)

References

Normative References

- [RFC4291] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006
- [RFC4861] T. Narten, E. Nordmark, W. Simpson and H. Soliman, "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
- [RFC4941] T. Narten, R. Draves and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC4941, September 2007

- [RFC1034] P. Mockapetris, "Domain names - concepts and facilities ", RFC 1034, November 1987
- [RFC1035] P. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987
- [RFC2136] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System," RFC 2136, April 1997
- [RFC4795] B. Aboba, D. Thaler, and L. Esibov, "Link-Local Multicast Name Resolution (LLMNR)," RFC4795, January 2007

Informative References

- [RFC4620] M. Crawford and B. Haberman, "IPv6 Node Information Queries," RFC4620, August 2006
- [mDNS] S. Cheshire and M. Krochmal, "Multicast DNS" <draft-cheshire-dnsext-multicastdns-14.txt> work in progress, February 2011
- [RFC3849] G. Huston, A. Lord and P. Smith, "IPv6 Address Prefix Reserved for Documentation," RFC3849, July 2004

Authors' Addresses

Hiroshi Kitamura
Knowledge Discovery Research Laboratories, NEC Corporation
(SC building 12F)1753, Shimonumabe, Nakahara-Ku, Kawasaki,
Kanagawa 211-8666, JAPAN
Graduate School of Information Systems,
University of Electro-Communications
5-1 Chofugaoka 1-Chome, Chofu-shi, Tokyo 182-8585, JAPAN
Phone: +81 44 431 7686
Fax: +81 44 431 7680
Email: kitamura@da.jp.nec.com

Shingo Ata
Graduate School of Engineering, Osaka City University
3-3-138, Sugimoto, Sumiyoshi-Ku, Osaka 558-8585, JAPAN
Phone: +81 6 6605 2191
Fax: +81 6 6605 2191
Email: ata@info.eng.osaka-cu.ac.jp

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 2013

H. Kitamura
NEC Corporation
S. Ata
Osaka City University
October 16, 2012

Corresponding Auto Names for IPv6 Addresses
<draft-kitamura-ipv6-auto-name-03.txt>

Abstract

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses. With this mechanism, all IPv6 addresses (even if they are link-local scoped addresses) can obtain their own Names, and it will be able to use Names anywhere instead of IPv6 Addresses.

IPv6 address is too long and complicated to remember, and it is very nuisance to type a literal IPv6 address manually as an argument of applications. Also, it is very difficult for human beings to tell which IPv6 address is set to which actual IPv6 node. In this sense, literal IPv6 address information can be called meaningless information for human beings.

In order to solve above problems and to provide annotated meaningful information to IPv6 addresses, mechanisms called Corresponding Auto Names for IPv6 addresses is introduced. They will become human friendly information. By applying a simple naming rule to the Auto Names (e.g., use the same Auto Name Suffix for IPv6 addresses that are set to the same interface (node)), this will contribute to help people to understand which IPv6 address / Name indicates which actual IPv6 node and provide meaningful information.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 2013.

Copyright Notice

Copyright (c) 2011 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.

Table of Contents

1. Introduction	3
2. Goals (What can be achieved)	4
2.1. Assumed typical IPv6 communication environment:	4
2.2. Auto Names examples	4
2.3. Auto Name Suffix for Grouped Addresses	5
2.4. Contribution in Regular Resolving (Name -> Address)	6
2.5. Contribution in Reverse Resolving (Address -> Name)	6
3. Deployed Notions and Functions that are used in Auto Names	8
3.1. Stateless Name	8
3.2. Scoped Name	9
3.3. Target IPv6 Addresses	10
4. Design of Auto Names	11
4.1. Conceptual Design on Naming Rules	11
4.1.1. <P> Value:	11
4.1.2. <I> Value:	12
4.1.3. <NGI> Value:	12
4.2. Address Type Distinction	13
4.2.1. EUI64-based Address Identification	13
4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction	13
5. Name Services	15
6. Security Considerations	15
7. IANA Considerations	15

Appendix A.	16
A. IPv6 Address Appearance Detection and Auto Name Registration .	16
A.1. IPv6 Address Appearance Detection mechanism	16
A.2. Auto Names Generation and Registration mechanism	16
A.3. Placement of Detector and Registrar	17
A.4. Detection and Registration Procedures	18
Appendix B. Implementation	19
Acknowledgment	19
References	19
Authors' Addresses	20

1. Introduction

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses.

IPv6 address is too long and complicated to remember. For human being, values of EUI64-based or temporary addresses should be felt that they are random number series. So, it is very nuisance to type a literal IPv6 address manually as an argument of applications.

Furthermore, it is very normal and popular cases to set multiple IPv6 addresses to one node. One IPv6 node owns more than two IPv6 addresses (typically: one is link-local scoped address. the other is global scoped address) at least. Some IPv6 addresses (such as link-local scoped stateless auto-configuration addresses and temporary addresses) may become users' conscious-less address, because they are automatically set to the IPv6 node.

It is too difficult for human beings to tell which IPv6 address is set to which IPv6 node. In other words, when an IPv6 address is shown to a person, he almost can not tell that the shown IPv6 address indicates which IPv6 node. In this sense, literal IPv6 address information can be called useless or meaningless information for human beings.

Moreover, when more than two literal IPv6 addresses are shown to human beings, it is almost impossible to distinguish them whether they are same or not at a glance.

So, there are strong desires to use Name information (that is human friendly) instead of literal IPv6 Address information and to use meaningful and distinguishable information that can easily show which IPv6 address / Name indicates which actual IPv6 node.

The Corresponding Auto Names for IPv6 Addresses is introduced to solve above problems and to satisfy the above desires.

2. Goals (What can be achieved)

In this section, goals of the mechanisms of the Corresponding Auto Names for IPv6 Addresses and what can be achieved are shown by using examples.

2.1. Assumed typical IPv6 communication environment:

Two IPv6 nodes (Node A and Node B) are located on the same link. Their IPv6 Addresses are shown below.

Node A:	Literal Address

MAC Address:	00:0d:5e:b8:80:7b

LL-Address:	fe80::20d:5eff:feb8:807b%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b
	fd01:2345:6789::1234
Global Addr:	2001:db8::20d:5eff:feb8:807b
	2001:db8::1234
Node B:	Literal Address

MAC Address:	00:0c:76:d9:14:e3

LL-Address:	fe80::20c:76ff:fed9:14e3%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3
	fd01:2345:6789::5678
Global Addr:	2001:db8::20c:76ff:fed9:14e3
	2001:db8::5678

They own altogether 5 IPv6 addresses respectively;

One Link-Local scoped Address

Two Unique Local Addresses (SLLAC and manual set address)

Two Global scoped Addresses (SLLAC and manual set address)

They communicate each other.

2.2. Auto Names examples

For all addresses, respective Corresponding Auto Names are prepared and registered to a some name resolving service DB (typically the DNS is used for this) automatically by the mechanism that detects these addresses (that is explained after in this document).

Prepared Auto Names are shown below.

Node A:	Literal Address	Auto Name
MAC Address:	00:0d:5e:b8:80:7b	
LL-Address:	fe80::20d:5eff:feb8:807b%fxp0	-> L0-7bz%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b	-> U0-7bz
	fd01:2345:6789::1234	-> U1-7bz
Global Addr:	2001:DB8::20d:5eff:feb8:807b	-> G0-7bz
	2001:DB8::1234	-> G1-7bz
Node B:	Literal Address	Auto Name
MAC Address:	00:0c:76:d9:14:e3	
LL-Address:	fe80::20c:76ff:fed9:14e3%em0	-> L0-3ez%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3	-> U0-3ez
	fd01:2345:6789::5678	-> U1-3ez
Global Addr:	2001:DB8::20c:76ff:fed9:14e3	-> G0-3ez
	2001:DB8::5678	-> G1-3ez

2.3. Auto Name Suffix for Grouped Addresses

In order to make Auto Names meaningful, IPv6 addresses are grouped and Auto Name Suffix is used to show grouped addresses.

For IPv6 addresses that are set to the same interface (node), the same Auto Name Suffix that stands for the Group ID is used for their Auto Names.

As shown above:

'-7bz' is used for Auto Name Suffix (Group ID) for Node A.
 '-3ez' is used for Auto Name Suffix (Group ID) for Node B.

In order to make easier to identify and remember the Auto Name Suffixes, their naming rule is based on inheriting the last octet of the node's MAC address in this example.

2.4. Contribution in Regular Resolving (Name -> Address)

In order to communicate with the specific IPv6 address of the destination node, the following procedure to type literal IPv6 address is required in the current environment. They are very stressful and nuisance procedures for human beings.

When 'ping6' or 'telnet' to the specific IPv6 address of Node B from Node A is executed, the following commands are typed.

```
>ping6 fe80::20c:76ff:fed9:14e3%fxp0
>telnet fd01:2345:6789::20c:76ff:fed9:14e3
```

Especially for link-local scoped addresses or temporary addresses, there are no way to type Names instead of literal IPv6 addresses, because they are generally not registered to name resolving services.

By introducing the Corresponding Auto Names, above typed commands are changed and replaced with the following easy and rememberable name typing procedures.

```
>ping6 L0-3ez%fxp0
>telnet U0-3ez
```

2.5. Contribution in Reverse Resolving (Address -> Name)

Communication related status information is shown to human beings in literal IPv6 address format in the current environment.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
fe80::20d:5eff:feb8:807b.8722	fe80:3::20c:76ff:fed9:14e3.23	ESTABLISH
fd01:2345:6789::1234.16258	fd01:2345:6789::5678.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
fe80::20d:5eff:feb8:807b%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
fe80::221:85ff:fea7:82ff%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
fe80::20c:76ff:fed9:14e3%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
fd01:2345:6789::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
fd01:2345:6789::5678	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
2001:DB8::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
2001:DB8::5678	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

People almost can not tell which shown literal IPv6 address indicates which IPv6 node. In this sense, shown information is meaningless and useless.

By introducing the Corresponding Auto Names, above complicated information is converted into simple and meaningful information and shown as followed.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
L0-7bz.8722	L0-e3z.23	ESTABLISH
U0-7bz.16258	U0-e3z.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
L0-7bz%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
U0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
U1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
L0-ffz%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
L0-3ez%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
U0-3ez	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
U1-3ez	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
G0-3ez	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
G1-3ez	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

Other examples where the Auto Name technique can contribute:

In log files of a server application, accesses from clients are recorded into them in literal IPv6 address format. It is almost impossible to read and understand the log files effectively without this Auto Name technique.

Also, in packet dumping applications, address information is shown in literal IPv6 address format. This Auto Name technique can significantly help for human beings to analyze and understand dumped packets.

Shown communication related status information in Auto Name format is simple and easy enough for human beings to understand. As shown above, troublesome IPv6 literal Address information can be converted into meaningful and distinguishable information by using the Corresponding Auto Names technique, and we can achieve our goals.

3. Deployed Notions and Functions that are used in Auto Names

3.1. Stateless Name

We know that we can categorize Addresses into two types. One is "stateful" address type, and the other is 'stateless' address type.

On the other, we have not been applied the same categorization to domain Names or host Names clearly. It has been assumed that existing all Names are categorized into stateful type and there is no stateless name type. Authors think that it is a time to change this preconception.

We can grasp that the introduced Corresponding Auto Name is realization of "stateless" name type, and we have deployed a notion Stateless Name clearly here.

Table 1 Stateless Name

	Stateful	Stateless
Address	DHCPv6	SLAAC
Nmae	existing Domain Names	Auto Names

3.2. Scoped Name

We also know that a notion called "scope" (such as link-local scope, global-scope) is introduced when we deal with addresses. Every address has its own scope.

In domain names or host names cases, the "scope" notion have NOT clearly introduced now. It is generally assumed that all names are global information and "scope" notion does not exist.

The Corresponding Auto Name is achieved by introducing Scoped Name obviously.

Scope of Auto Name for IPv6 address is the same to the scope of its IPv6 address. For example, scope of the Auto Name for the link-local IPv6 address is link-local. They are only effective within the link-local scope.

Table 1 Scoped Name

	Global	Site-Local (ULA)	Link-Local	Node-Local
Address	2001:db8::/64	fd01:2345:6789::/64	fe80::/64	
Nmae	Domain Names	Domain Names / Auto Names	Auto Names	Auto Names

At some special situation (that it is enough that Auto Name information is shared within a node), Node-Local scoped Auto Name is possible. At such a situation, we can use /etc/hosts file as a name service.

Deployment of Scoped Name:

Scoped Name notion can be easily achieved with current technologies.

As shown above, at a special case when we adopt /etc/hosts file as a name service for Auto Names, scope of Auto Names naturally becomes Node-Local.

At general cases when we adopt the DNS as a name service for Auto Names, scope of Auto Names is easily managed by the DNS query access permission control on DNS servers.

3.3. Target IPv6 Addresses

One of the goals of the Auto Name technique is to provide and set Names to all IPv6 addresses (include Link-local addresses).

All IPv6 Addresses are targets of Auto Names.

Some IPv6 Addresses have their own names (let's call them Proper Names) that are assigned manually and are registered into name resolving services (such as the DNS). It may be thought that it is not necessary to assign Auto Name to such IPv6 addresses which have Proper Names. However, we strongly recommend assigning Auto Names to them, too.

In order to provide meaningful information to IPv6 addresses, uniformed Name information for all IPv6 addresses is necessary. This is very natural behavior for 'Stateless' type technology.

Since one-to-multiple mapping is allowed in name resolving services, it will not cause problems to assign both Proper Name and Auto Name for one IPv6 address.

We may need some function that controls name display priority (which name is first Proper Name or Auto Name). This function also achieved easily by using existing current technologies.

If Auto Name and Proper Name are implemented as different name resolving services (e.g., one is /etc/hosts, the other is the DNS), name display priority is can be easily controlled by nsswitch.conf function.

If Auto Name and Proper Name are implemented as same name resolving service, name display priority is can be controlled by their registration order to the name resolving service DB.

4. Design of Auto Names

4.1. Conceptual Design on Naming Rules

Auto Names are composed of "<P><I>-<NGI>" format:

<P>: stands for Prefix of Address

1 character: (e.g., 'L', 'U', 'G')

<I>: stands for Interface ID of Address

1 character: (e.g., '0', '1')

<NGI>: stands for Node (Interface) Group ID

3 characters:
(e.g., '7bz', '3ez')

Above discussed Auto Name examples satisfy <P><I>-<NGI> format.

on Node A: L0-7bz, U0-7bz, U1-7bz, G0-7bz, G1-7bz

on Node B: L0-3ez, U0-3ez, U1-3ez, G0-3ez, G1-3ez

4.1.1. <P> Value:

<P> value stands for Prefix (Scope) (upper 64 bit) of Address as 1 character format.

Auto Names of IPv6 addresses whose prefixes are same use the same <P> value.

Typically, following characters are used for <P> value:

"L": used for Link-local scoped addresses.

"U": used for ULA

"G": used for Global scoped address

If multiple prefixes for the same scope are used, other character (such as "H", "I",,,) can be used depending on the circumstances.

"Prefix - <P> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "Prefix - <P> value" mapping table) is used to avoid collision and manage mappings of them.

4.1.2. <I> Value:

<I> value stands for Interface ID (lower 64 bit) of Address as 1 character format.

Following characters are used for <I> value:

<I> value assignment is based on three address type categorization.

"0": used for EUI64-based address
"1", "9": used for manually set addresses
(stateful addresses will be categorized here)

"a", "z": used for automatically generated and set addresses
except EUI64-based
(Temporary addresses are categorized here)

4.1.3. <NGI> Value:

<NGI> value is also called Auto Name-Suffix.

In order to make IPv6 addresses meaningful, IPv6 addresses are grouped. It is very natural to group IPv6 addresses by which node (interface) they are set. So, IPv6 addresses that are set to the same node (interface) are grouped into the same group.

<NGI> value is shown as 'XYZ' format:

'XY': (1st, 2nd chars) are inherited from
the last octet (2 characters) of the node's MAC address
'Z' : (3rd char) suffix char to avoid a collision of 'XY'
starting from "z"
if 'XY' is collided, 'Z' is changed into "y", "x" , , ,

By using the birthday paradox theorem, collision probability of 256 states (1 octet) is calculated. If 19 nodes (interfaces) exist on the same link, collision is happened with 50% probability. Collision check procedure of the last octet of MAC addresses is necessary.

"MAC address - <NGI> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "MAC address - <NGI> value" mapping table) is needed to avoid collision and manage mappings of them.

Detailed methods how to distinguish and categorize IPv6 addresses are described at the following section.

We can found one remarkable thing with the naming rules:

"L0-XYZ" is a special and very standard Auto Name. "L0-XYZ" is an Auto Name that assigned for Link-local scoped EUI64-based address. Since Almost all the IPv6 nodes have Link-local scoped EUI64-based address, with "L0-XYZ" name information we can reach or communicate the node whose last octet MAC address is known as "XY".

4.2. Address Type Distinction

If we can obtain address type information from Auto Name, convenient environment will be provided. So, there are strong desires to understand type of detected IPv6 addresses. In this section, methods how to distinguish and categorize IPv6 addresses are described.

4.2.1. EUI64-based Address Identification

Only with IPv6 address information, it is impossible to identify that the detected IPv6 address is EUI64-based address or not.

In proposed IPv6 address detection method which is described in the following section, IPv6 address and MAC address that are set to the same node (interface) is detected simultaneously.

So, with this detection method, it is very easy to identify that the detected IPv6 address is EUI64-based address or not.

4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction

It is generally difficult to distinguish whether the detected IPv6 address is manually or automatically set address.

In order to distinguish IPv6 address types, "Zero Contain Rate" technique is introduced.

For a human being, "Zero" is a special value. When a human being omits a part of information, "Zero" is used for the omitted part of information implicitly.

For a machine "Zero" is NOT a special value. "Zero" is treated as almost equal to other values.

We can reach a fact that manually set IPv6 address contains many "Zero", because it is assigned by a human being. 64bit is long for

a human being and there must be too many omitted part filled with "Zero". Especially, a human being is apt to omit and fill with "Zero" upper part of 64bit Interface ID.

In other words, we can see human relation bias from "Zero Contain Rate" of IPv6 address.

Bias Check by using Mathematical Technique:

By using mathematical probability technique, we can distinguish whether value of 64bit Interface ID is biased (manual) or non-biased (automatic).

When we see 64bit value in 8bit (1 octet) unit, it follows the binomial distribution ($n=8$ and $p=1/256$).

Under this distribution, the probability to meet "Zero" octet two times is 0.042%. It means that to meet "Zero" octet two times is too rare case if the value is non-biased.

So, we can adopt the following method:

If the value of 64bit Interface ID contains two and above "Zero" octets,
the value is bias and
the IPv6 address is identified as a manually set address.

Of cause, this method is not perfect because this is based on mathematical probability technique and heuristic human behavior, but this is very effective.
Even if wrong identification is done, no big problems are found.

5. Name Services

It is not clearly defined which Name Services is utilized to achieve Auto Names specifications. In other words, any types of Name Services can be utilized. Which Name Services is utilized is tightly dependent on which Scoped Name is adopted for the Auto Names.

If wide Scoped Name is adopted, it is very natural to utilize wide Name Service (i.e. the DNS) as the Name Services for Auto Names. If narrow scope (e.g., node-local scoped name) is adopted for Auto Names, it becomes possible to utilize node-local scoped Name Service (e.g., /etc/hosts) for Auto Name.

6. Security Considerations

Auto Names are generated and registered to the name service in this document. In order to register correct Auto Names information, communication between Detector and Registrar and communication between Registrar and Name Server should be protected and be secured.

In general usage, scope of Auto Names will be local (not global). Auto Names are usually local scoped names. So, we do not have to be too sensitive on the correctness of Auto Names.

7. IANA Considerations

This document does not require any resource assignments to IANA.

Appendix A. IPv6 Address Appearance Detection and Auto Name Registration

In order to generate and register Auto Names automatically, two types of mechanisms are needed. One is a mechanism that detects IPv6 address appearance. The other is a mechanism that checks the detected addresses and generates Auto Names and registers them to name service.

Two functions ("Detector" and "Registrar") are introduced. "Detector" function takes in charge of the former mechanism, and "Registrar" takes in charge the latter mechanism.

A.1. IPv6 Address Appearance Detection mechanism

In order to detect newly appeared IPv6 address, DAD message (NS for DAD) is effectively used.

DAD message has the following good capabilities:

- issued only when node would like to set new IPv6 address
- issued for All types (link-local, global, temporary,,,))
- L2 broadcast and easy to capture (without using mirror port)
- distinguishable from other NS messages, because source address of the message is unspecified ("::") and different from others
- Captured DAD message includes all necessary information (such as, IPv6 address and MAC address)

Detector captures DAD messages and detects newly appeared IPv6 addresses. Detected information is sent to Registrar.

A.2. Auto Names Generation and Registration mechanism

At first, Registrar checks the Detected address information that is sent from Detector(s). By using the reverse resolving (Address -> Name), it is checked whether the Detected address information is first appearance or not. If an entry for the address does NOT exist, it is confirmed that the address is first appearance and it should be registered to the name server.

After Name for the address is prepared, duplication of the Name can be checked by using the regular resolving (Name -> Address). If an entry for the Name exist, it is confirmed that Name is duplicated (collided). Another Name is prepared and checked again until the Name

is not duplicated.

Finally, Registrar registers both Regular and Reverse resolving entries for the address and prepared Auto Name are registered to the name server.

A.3. Placement of Detector and Registrar

Placement of Detector and Registrar is designed to make the mechanisms flexible and to make it to be applied to various environments (office networks, home networks, etc.)

Figure 1 and 2 show typical examples that indicate locations where Detector and Registrar functions are placed on the IPv6 network. Figure 1 shows a case for a single link, and Figure 2 shows a case for multiple links.

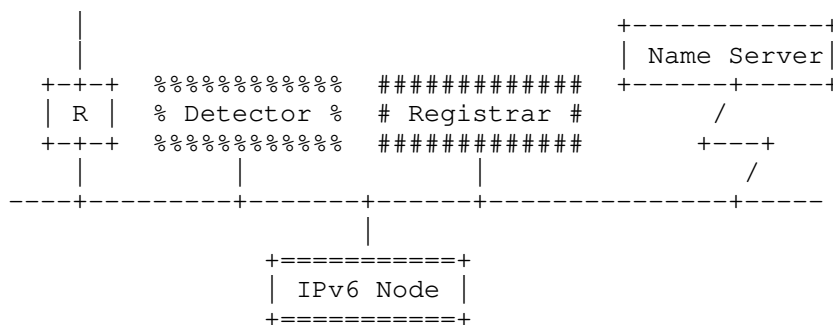


Fig. 1 Single-Link Case Example

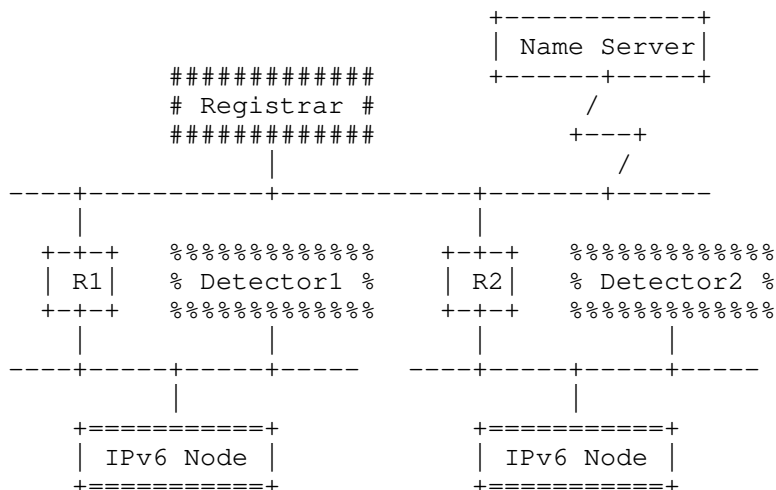
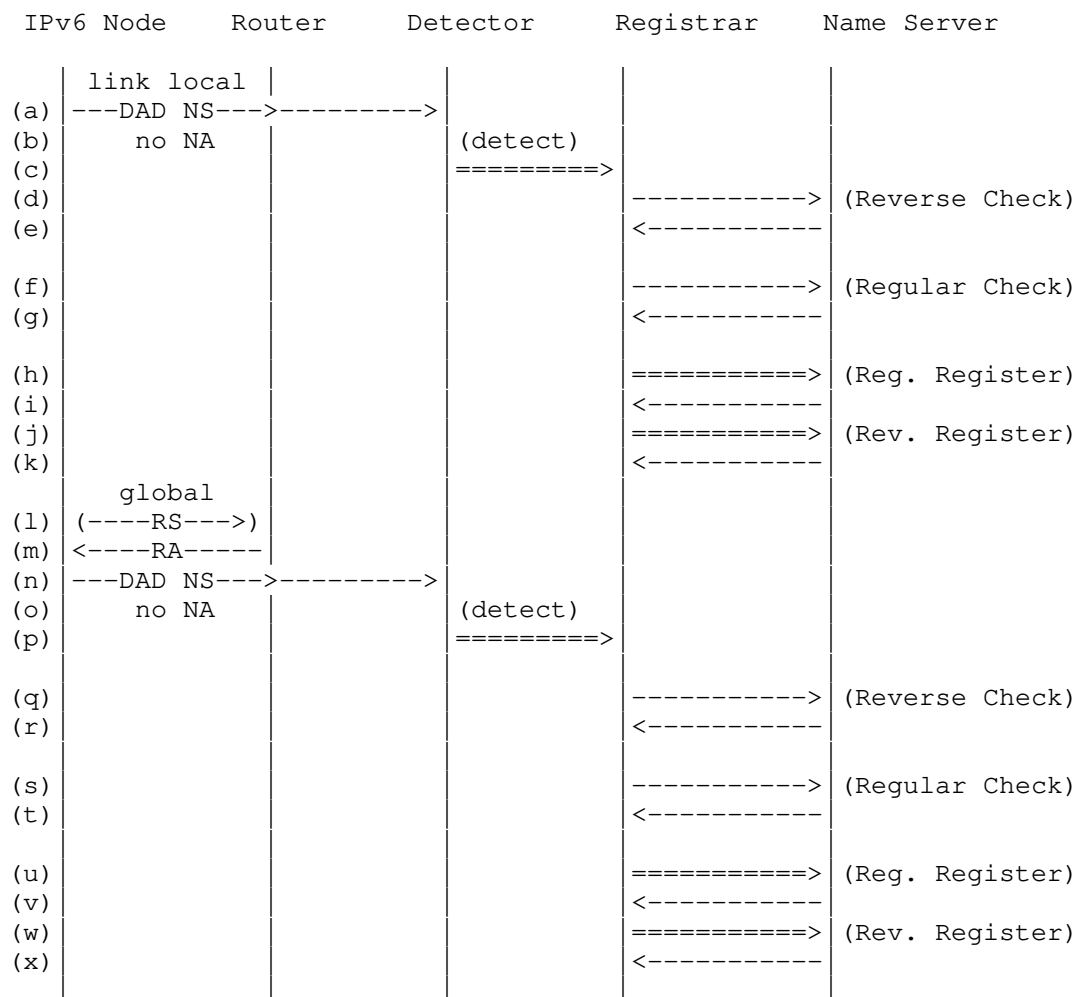


Fig. 2 Multiple-Link Case Example

A.4. Detection and Registration Procedures

Figure 3 shows an example of typical detection and registration procedures at IPv6 links where DAD packets are issued. DAD message packets are used for the appearance detection.



Appendix B. Implementation

Auto Name functions have been implemented at the following environments. It has been verified that designed functions work well.

Used functions:

Packet capture on Detector: libpcap
Name Server: DNS (BIND9)
Name Registration: nsupdate (BIND9 bundled)

OS:

FreeBSD 6.2R
(Since FreeBSD OS specific functions are not used to implement, codes will run on UNIX type OS that has libpcap (such as Linux).)

Acknowledgment

A part of this work is supported by the program: SCOPE (Strategic Information and Communications R&D Promotion Programme) operated by Ministry of Internal Affairs and Communications of JAPAN.

References

Normative References

- [RFC4291] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006
- [RFC4861] T. Narten, E. Nordmark, W. Simpson and H. Soliman, "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
- [RFC4941] T. Narten, R. Draves and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC4941, September 2007
- [RFC1034] P. Mockapetris, "Domain names - concepts and facilities ", RFC 1034, November 1987
- [RFC1035] P. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987
- [RFC2136] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System", RFC 2136, April 1997

[RFC4795] B. Aboba, D. Thaler, and L. Esibov, "Link-Local Multicast Name Resolution (LLMNR)," RFC4795, January 2007

Informative References

[RFC4620] M. Crawford and B. Haberman, "IPv6 Node Information Queries," RFC4620, August 2006

[mDNS] S. Cheshire and M. Krochmal, "Multicast DNS" <draft-cheshire-dnsext-multicastdns-14.txt> work in progress, February 2011

[RFC3849] G. Huston, A. Lord and P. Smith, "IPv6 Address Prefix Reserved for Documentation," RFC3849, July 2004

Authors' Addresses

Hiroshi Kitamura
Knowledge Discovery Research Laboratories, NEC Corporation
(SC building 12F)1753, Shimonumabe, Nakahara-Ku, Kawasaki,
Kanagawa 211-8666, JAPAN
Graduate School of Information Systems,
University of Electro-Communications
5-1 Chofugaoka 1-Chome, Chofu-shi, Tokyo 182-8585, JAPAN
Phone: +81 44 431 7686
Fax: +81 44 431 7680
Email: kitamura@da.jp.nec.com

Shingo Ata
Graduate School of Engineering, Osaka City University
3-3-138, Sugimoto, Sumiyoshi-Ku, Osaka 558-8585, JAPAN
Phone: +81 6 6605 2191
Fax: +81 6 6605 2191
Email: ata@info.eng.osaka-cu.ac.jp

6man Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

S. Krishnan
Ericsson
D. Anipko
D. Thaler
Microsoft
July 16, 2012

Packet loss resiliency for Router Solicitations
draft-krishnan-6man-resilient-rs-01

Abstract

When an interface on a host is initialized, the host transmits Router Solicitations in order to minimize the amount of time it needs to wait until the next unsolicited multicast Router Advertisement is received. In certain scenarios, these router solicitations transmitted by the host might be lost. This document specifies a mechanism for hosts to cope with the loss of the initial Router Solicitations. Furthermore, on some links, unsolicited multicast Router Advertisements are never sent and the mechanism in this document is intended to work even in such scenarios.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 17, 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
1.1. Conventions used in this document	3
2. Proposed algorithm	4
3. Open Issue	4
4. IANA Considerations	5
5. Security Considerations	5
6. Acknowledgements	5
7. References	5
7.1. Normative References	5
7.2. Informative References	5
Authors' Addresses	5

1. Introduction

As specified in [RFC4861], when an interface on a host is initialized, in order to obtain Router Advertisements quickly, a host transmits up to MAX_RTR_SOLICITATIONS (3) Router Solicitation messages, each separated by at least RTR_SOLICITATION_INTERVAL (4) seconds. In certain scenarios, these router solicitations transmitted by the host might be lost.

The generic scenario is that the interface on the host comes up before it gets access to a router. Examples include:

- a. The host is connected to a bridged residential gateway over Ethernet or WiFi. LAN connectivity is achieved at interface initialization, but the upstream WAN connectivity is not active yet. In this case, the host just gives up after the initial RS retransmits.
- b. Access networks/links that turn off periodic RAs and only send RAs in response to RSs. In this case, if the link between the AP and the host comes up before the link between the AP and the Controller/Router, the host will never be able to connect.
- c. Links that are not multicast capable. In this case, sending an RA can only be triggered by an RS (as is the case, for instance, on ISATAP [RFC5214] links).

Once the initial RSs are lost, the host gives up and assumes that there are no routers on the link as specified in Section 6.3.7 of [RFC4861]. The host will not have any form of Internet connectivity until the next unsolicited multicast Router Advertisement is received. These Router Advertisements are transmitted at most MaxRtrAdvInterval seconds apart (maximum value 1800 seconds). Thus in the worst case scenario a host would be without any connectivity for 30 minutes. In general, the delay may be unacceptable in some scenarios.

1.1. Conventions used in this document

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

2. Proposed algorithm

To achieve resiliency to packet loss, the host needs to continue retransmitting the Router Solicitations until it receives a Router Advertisement, or until it is willing to accept that no router exists. If the host continues retransmitting the RSs at RTR_SOLICITATION_INTERVAL second intervals, it may cause excessive network traffic if a large number of such hosts exists. To achieve resiliency while keeping the aggregate network traffic low, the host can use some form of exponential backoff algorithm to retransmit the RSs.

Hosts complying to this specification MUST use the exponential backoff algorithm for retransmits that is described in Section 14 of [RFC3315] in order to continuously retransmit the Router Solicitations until a Router Advertisement is received. The hosts SHOULD use the following variables as input to the retransmission algorithm:

```
IRT  4 seconds
MRT  3600 seconds
MRC  0
MRD  0
```

The initial value IRT was chosen to be in line with the current retransmission interval (RTR_SOLICITATION_INTERVAL) that is specified by [RFC4861] and the maximum retransmission time MRT was chosen to be in line with the new value of SOL_MAX_RT as specified by [SOLMAXRT]. This is to ensure that the short term behavior of the RSs is similar to what is experienced in current networks, and longer term persistent retransmission behavior trends towards being similar to that of DHCPv6 [RFC3315] [SOLMAXRT].

3. Open Issue

When an IPv6-capable host attaches to a network that does not have IPv6 enabled, it transmits 3 (MAX_RTR_SOLICITATIONS) Router Solicitations as specified in [RFC4861]. If it receives no Router Advertisements, it assumes that there are no routers present on the link and it ceases to send further RSs. With the mechanism specified in this document, the host will continue to retransmit RSs indefinitely at the rate of approximately 1 RS per hour. It is unclear how to differentiate between such a network with no IPv6 routers and a link where an IPv6 router is temporarily unreachable but could become reachable in the future.

4. IANA Considerations

This document does not require any IANA actions.

5. Security Considerations

This document does not present any additional security issues beyond those discussed in [RFC4861].

6. Acknowledgements

The author would like to thank Steve Baillargeon, and Erik Kline for their reviews and suggestions that made this document better.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [SOLMAXRT] Droms, R., "Modification to Default Value of SOL_MAX_RT", draft-droms-dhc-dhcpv6-solmaxrt-update-02 (work in progress), January 2012.

7.2. Informative References

- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

Authors' Addresses

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

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

Dmitry Anipko
Microsoft
One Microsoft Way
Redmond, WA
USA

Phone: +1 425 703 7070
Email: danipko@microsoft.com

Dave Thaler
Microsoft
One Microsoft Way
Redmond, WA
USA

Email: dthaler@microsoft.com

Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: January 14, 2013

B. Liu
Huawei Technologies Co., Ltd
W. Wang
X. Gong
University of BUPT
July 16, 2012

DHCPv6/SLAAC Address Configuration Switching for Host Renumbering
draft-liu-6renum-dhcpv6-slaac-switching-01.txt

Status of this Memo

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

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

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

This Internet-Draft will expire on January 14, 2013.

Copyright Notice

Copyright (c) 2011 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

Sometimes stateful DHCPv6 address configuration and SLAAC may be both available in one network. In ND protocol, there is a "M" (ManagedFlag) flag defined in RA message, which indicates the hosts the DHCPv6 service is available. But for some reason, the ND protocol didn't define the flag as prescriptive but only advisory. This draft proposes to use two reserved bits in RA message to let the network control the hosts that which address configuration mode should be used. This feature is useful for management, especially in a renumbering event.

Table of Contents

1. Introduction	3
2. DHCPv6/SLAAC interaction	3
2.1. Host behavior defined in standards	3
2.2. Test of desktop operating systems' behavior	4
2.2.1. Test environment	4
2.2.2. Test scenarios and results	5
2.2.3. Conclusion.....	6
3. Requirement of Address Configuration Switching in Renumbering	6
4. Proposed Standard Update	7
4.1. Adding a "DHCPv6Required" Flag	8
4.2. Adding a "ReleaseDHCPv6" Flag	8
4.3. Host Behavior of Interpreting D/R Flag	8
5. Security Considerations	9
6. IANA Considerations	9
7. References	9
7.1. Normative References	9
7.2. Informative References	9
8. Acknowledgments	9
Authors' Addresses	10

1. Introduction

In IPv6, both of the DHCPv6 [RFC3315] and Neighbor Discovery [RFC4861] protocols can provide automatic IP address configuration for the hosts. They are known as stateful address auto-configuration and SLAAC (stateless address auto-configuration) [RFC4862], and are suitable for different scenarios respectively.

Sometimes the two address configuration modes may be both available in one network. This would add more or less additional complexity for both the hosts and the network management. In ND protocol, there is a M (ManagedFlag) flag defined in RA message, which indicates the hosts the DHCPv6 service status in the network. So with using the flag, the two separated address configuration modes are somehow correlated. But for some reason, the ND protocol didn't define the flag as prescriptive but only advisory. This may vary the behavior of hosts when interpreting the M flag. (Note that, there is another O "OtherConfigFlag" flag also indicates the DHCPv6 service status, but it is not in the scope of this draft since it is not about address configuration.)

In RFC5887(Renumbering Still Needs Work), it also concerned the M flag issue, it said, "Until this ambiguous behaviour is clearly resolved by the IETF, operational problems are to be expected, since different host operating systems have taken different approaches." In this draft, we provided a brief test result in section 3 to identify "different host operating systems have taken different approaches".

This issue may cause inconvenience to the networks that need strong management (for example, the enterprise networks), because the host behavior of address configuration is somehow un-controlled by the network side so that it may violate the management policies. So in section 4, we proposed to use one of the reserved bits in RA message to let the network control the hosts that which address configuration mode should be used. We believe this feature is useful for management, especially in a renumbering event.

2. DHCPv6/SLAAC interaction

2.1. Host behavior defined in standards

In earlier SLAAC specification [RFC2462], the host behavior of interpreting M flag is as below:

"On receipt of a valid Router Advertisement, a host copies the value of the advertisement's M bit into ManagedFlag. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information. If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement."

But for some reason, the updated SLAAC specification [RFC4862] removed the relative description, it said in the RFC "considering the maturity of implementations and operational experiences. ManagedFlag and OtherConfigFlag were removed accordingly. (Note that this change does not mean the use of these flags is deprecated.)" So it feels like the IETF encourages operating system vendors to behave as they prefer to do. In the following section 2.2, we provided a test about current desktop operating systems' behavior of DHCPv6/SLAAC interaction.

2.2. Test of desktop operating systems' behavior

2.2.1. Test environment

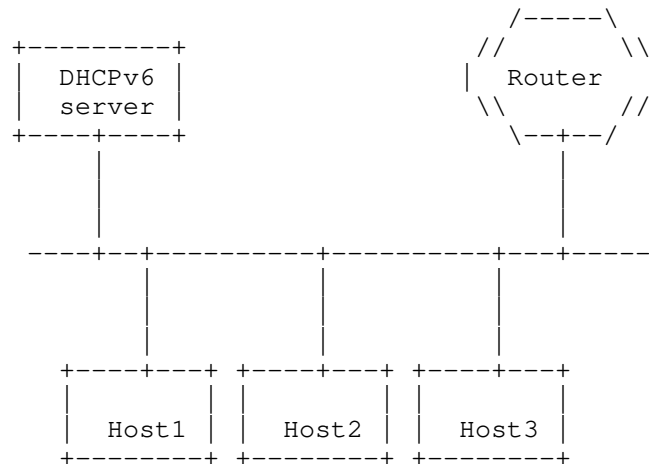


Figure 1 Test Environment

As the figure 1 shows, it is a simple LAN environment:

- The DHCPv6 server is a Linux (Ubuntu 10.04)-based PC installing `dibbler-server`.
- Host1 is a Windows 7 PC.
- Host2 is a Linux (Ubuntu 12.04, kernel 3.2.12) PC.
- Host3 is a OS X Lion 10.7.3 MacBook.

Note that, we only tested M flag behavior, O flag was not included. Because O flag is about other configuration beyond address configuration, it is out of the scope of this draft.

2.2.2. Test scenarios and results

o Scenario 0

Hosts get online, no RA received.

- Windows 7: continued sending RS messages for a while, if there is no RA replied, it then began to send DHCPv6 solicit;
- Linux-kernel_3.2.12(Ubuntu 12.04): it continued sending RS, and didn't try to send DHCPv6 solicit;
- OS X Lion 10.7.3: it continued sending RS, and didn't try to send DHCPv6 solicit (just the same with Linux);

o Scenario 1

Hosts hadn't configured addresses yet, then if RA messages with M=0 received, obviously they'll do SLAAC; if M=1, which meant SLAAC and DHCPv6 were available simultaneously in the link, the behavior is as the following:

- Windows 7: using both SLAAC and DHCPv6 to configure the addresses, regardless of whether the prefixes in SLAAC/DHCPv6 are identical or not;
- Linux-kernel_3.2.12(Ubuntu 12.04): the same action with Windows 7;
- OS X Lion 10.7.3: the same action with Windows 7;

o Scenario 2

Hosts were already SLAAC-configured only, then received RA messages with M=1:

- Windows 7: using DHCPv6 to configure another address while keep the former SLAAC-configured address;
- Linux(Ubuntu 12.04): no action. (Note that, it's different with scenarios 1);
- OS X Lion 10.7.3: no action, the same with Linux;

o Scenario 3

Hosts already configured by DHCPv6 only, then received RA messages:

- Windows 7: If M=1, it configured another address with SLAAC and kept the DHCPv6 configuration; else M=0, it released the DHCPv6 address and configured with SLAAC;
- Linux-kernel_3.2.12(Ubuntu 12.04): there's no DHCPv6-only situation for it, only in scenario 1 when M=1 it would configured with SLAAC and DHCPv6 simultaneously;
- OS X Lion 10.7.3: the same situation with Linux;

o Scenario 4

Hosts already configured with SLAAC/DHCPv6 simultaneously, then RA messages with M=0 received:

- Windows 7: it released the DHCPv6 address and configured with SLAAC;
- Linux(Ubuntu 12.04): no action;
- OS X Lion 10.7.3: no action;

2.2.3. Conclusion

Obviously, the operating systems interpreting the M flag quite differently. Windows 7 treats the flag as instruction, it even released DHCPv6 session when M=0. Linux and OS X were likely to treat the flag as advisory, when SLAAC was done, it won't care about M=1, and M=0 won't cause operation for the already configured DHCPv6 addresses.

3. Requirement of Address Configuration Switching in Renumbering

During IPv6 renumbering, the SLAAC-configured hosts can reconfigure IP addresses by receiving ND Router Advertisement (RA) messages containing new prefix information. The DHCPv6-configured hosts can

reconfigure addresses by initialing RENEW sessions when the current addresses' lease time is expired or receiving the reconfiguration messages initialed by the DHCPv6 servers.

The above mechanisms have an implicit assumption that SLAAC-configured hosts will remain SLAAC while DHCPv6-managed hosts will remain DHCPv6-managed. In [I-D.ietf-6renum-enterprise], it described several renumbering scenarios in enterprise network. For example, the network may split, merge, grow, relocate or reorganize. In these situations, it is possible that SLAAC-configured hosts may need to switch to DHCPv6-managed, or verse vice.

As discussed in section 2, the semantic of M bit is ambiguous, for example, M=0 is efficient for Windows 7 PCs to switch from DHCPv6-managed to SLAAC, but for Linux or OS X it is just invalid. So in the following section 4, we proposed to use another two flags to indicate the hosts switching between SLAAC/DHCPv6.

4. Proposed Standard Update

4.1. Semantic Space of SLAAC/DHCPv6 Interaction

We summarize the semantic instructions from network side to host side as the following. Some of them are already covered by existing implementation while some may need protocol extentions.

- Network side provides both, let the hosts select by themselves

It is exactly what M=1 meaning in RFC4861.

- Network side requires the hosts to do DHCPv6 when online

As the tests showed, when get online, all the three major Oses will initial DHCPv6 when M=1. Especially, when M=1 and RAs don't include PIO (Prefix Information Option), the host would "have to" initial DHCPv6 for address autoconfiguration. So we can consider this semantics has been covered.

- Network side requires the already SLAAC-configured hosts to do DHCPv6

As the test showed, with current ambiguous M=1 definition, Oses varied the behaviors. So this could be considered as a semantic gap.

- Network side requires the hosts to release DHCPv6 addresses

As analyzed in section 3, the network may need the hosts to switch configuration modes. With M=0, the OS behaviors are different as the test results showed. So this is another semantic gap.

4.2. Adding a "DHCPv6Required" Flag

We propose to add a flag in the standard RA message format[RFC4861], the "DHCPv6Required" D flag, which will occupy one bit in the reserved field as showed in the following figure 2.

4.3. Adding a "ReleaseDHCPv6" Flag

We propose to add one more flag in the standard RA message format, the "ReleaseDHCPv6" R flag, which will occupy one more bit in the reserved field as showed in the following figure 2.

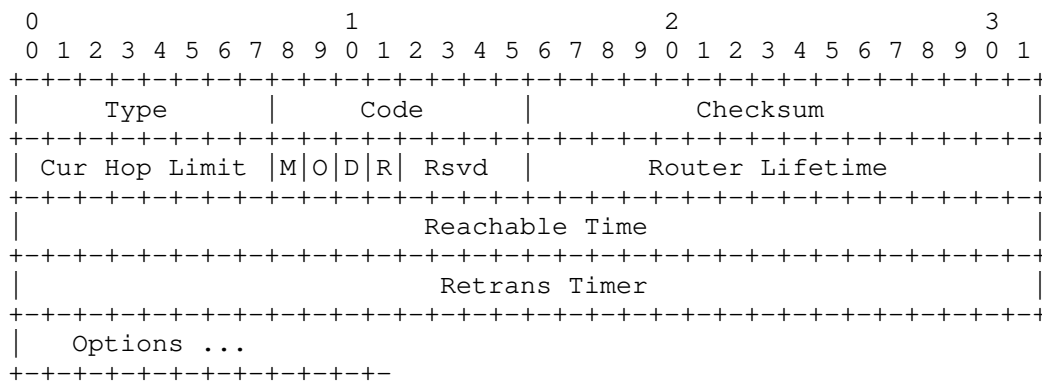


Figure 2 DHCPv6Required and ReleaseDHCPv6 flags in RA message

4.4. Host Behavior of Interpreting D/R Flag

When a host has not configured its addresses (just like scenario 0 in section 2.2) and receives RA messages with D=1, it MUST initiate a DHCPv6 stateful address autoconfiguration process.

When a host has been SLAAC-configured, and receives D=1, it MUST initiate a DHCPv6 stateful address autoconfiguration process and SHOULD deprecate SLAAC-configured addresses.

When a host has been address-configured with DHCPv6, and receives RA messages with R=1, it SHOULD release current DHCPv6 address configuration and do SLAAC.

5. Security Considerations

No more security considerations than the Neighbor Discovery protocol [RFC4861].

6. IANA Considerations

None.

7. References

7.1. Normative References

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

[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.

7.2. Informative References

[RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", RFC 5887, May 2010.

[I-D.ietf-6renum-gap-analysis]
Liu, B., and Jiang, S., "IPv6 Site Renumbering Gap Analysis", Working in Progress, March 2012

[I-D.ietf-6renum-enterprise]
Jiang, S., and B. Liu, "IPv6 Enterprise Network Renumbering Scenarios and Guidelines ", Working in Progress, March 2012.

8. Acknowledgments

The tests were done in a lab in BUPT. Thank Xudong Shi very much. He is a master student in the lab, and did a great job for the tests.

This work adopts some content from [I-D.ietf-6renum-gap-analysis].

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Bing Liu
Q14-4-A Building
Huawei Technologies Co., Ltd
Zhong-Guan-Cun Environment Protection Park, No.156 Beiqing Rd.
Hai-Dian District, Beijing
P.R. China

Email: leo.liubing@huawei.com

Wendong Wang
No.3 Teaching Building
Beijing University of Posts and Telecommunications
No.10 Xi-Tu-Cheng Rd.
Hai-Dian District, Beijing
P.R. China

Email: wdwang@bupt.edu.cn

Xiangyang Gong
No.3 Teaching Building
Beijing University of Posts and Telecommunications
No.10 Xi-Tu-Cheng Rd.
Hai-Dian District, Beijing
P.R. China

Email: xygong@bupt.edu.cn

Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: July 18, 2013

B. Liu
Huawei Technologies Co., Ltd
W. Wang
X. Gong
University of BUPT
January 15, 2013

DHCPv6/SLAAC Address Configuration Switching for Host Renumbering
draft-liu-6renum-dhcpv6-slaac-switching-02.txt

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 July 18, 2013.

Copyright Notice

Copyright (c) 2011 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

Sometimes stateful DHCPv6 address configuration and SLAAC may be both available in one network. In ND protocol, there is a "M" (ManagedFlag) flag defined in RA message, which indicates the hosts the DHCPv6 service is available. But for some reason, the ND protocol didn't define the flag as prescriptive but only advisory. This draft proposes to use two reserved bits in RA message to let the network control the hosts that which address configuration mode should be used. This feature is useful for management, especially in a renumbering event.

Table of Contents

1. Introduction	3
2. DHCPv6/SLAAC interaction	3
2.1. Host behavior defined in standards	3
2.2. Test of desktop operating systems' behavior	4
2.2.1. Test environment	4
2.2.2. Test scenarios and results	5
2.2.3. Conclusion	6
3. Requirement of Address Configuration Switching in Renumbering.	6
4. Proposed Standard Update	7
4.1. Adding a "DHCPv6Required" Flag	8
4.2. Adding a "ReleaseDHCPv6" Flag	8
4.3. Host Behavior of Interpreting D/R Flag	8
5. Security Considerations	9
6. IANA Considerations	9
7. References	9
7.1. Normative References	9
7.2. Informative References	9
8. Acknowledgments	9
Authors' Addresses	10

1. Introduction

In IPv6, both of the DHCPv6 [RFC3315] and Neighbor Discovery [RFC4861] protocols can provide automatic IP address configuration for the hosts. They are known as stateful address auto-configuration and SLAAC (stateless address auto-configuration) [RFC4862], and are suitable for different scenarios respectively.

Sometimes the two address configuration modes may be both available in one network. This would add more or less additional complexity for both the hosts and the network management. In ND protocol, there is a M (ManagedFlag) flag defined in RA message, which indicates the hosts the DHCPv6 service status in the network. So with using the flag, the two separated address configuration modes are somehow correlated. But for some reason, the ND protocol didn't define the flag as prescriptive but only advisory. This may vary the behavior of hosts when interpreting the M flag. (Note that, there is another O "OtherConfigFlag" flag also indicates the DHCPv6 service status, but it is not in the scope of this draft since it is not about address configuration.)

In RFC5887(Renumbering Still Needs Work), it also concerned the M flag issue, it said, "Until this ambiguous behaviour is clearly resolved by the IETF, operational problems are to be expected, since different host operating systems have taken different approaches." In this draft, we provided a brief test result in section 3 to identify "different host operating systems have taken different approaches".

This issue may cause inconvenience to the networks that need strong management (for example, the enterprise networks), because the host behavior of address configuration is somehow un-controlled by the network side so that it may violate the management policies. So in section 4, we proposed to use one of the reserved bits in RA message to let the network control the hosts that which address configuration mode should be used. We believe this feature is useful for management, especially in a renumbering event.

2. DHCPv6/SLAAC interaction

2.1. Host behavior defined in standards

In earlier SLAAC specification [RFC2462], the host behavior of interpreting M flag is as below:

"On receipt of a valid Router Advertisement, a host copies the value of the advertisement's M bit into ManagedFlag. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information. If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement."

But for some reason, the updated SLAAC specification [RFC4862] removed the relative description, it said in the RFC "considering the maturity of implementations and operational experiences. ManagedFlag and OtherConfigFlag were removed accordingly. (Note that this change does not mean the use of these flags is deprecated.)" So it feels like the IETF encourages operating system vendors to behave as they prefer to do. In the following section 2.2, we provided a test about current desktop operating systems' behavior of DHCPv6/SLAAC interaction.

2.2. Test of desktop operating systems' behavior

2.2.1. Test environment

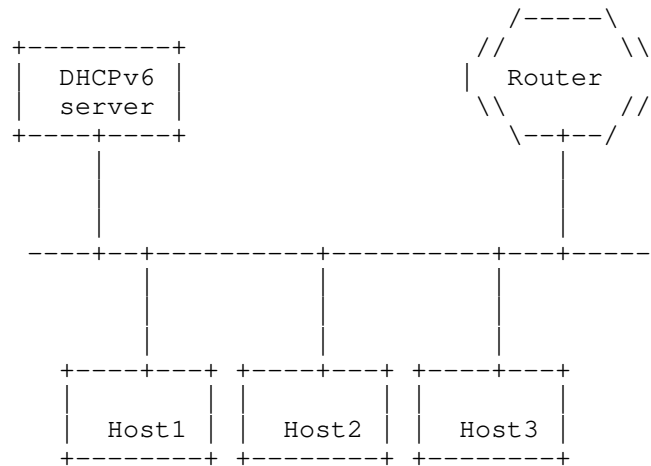


Figure 1 Test Environment

As the figure 1 shows, it is a simple LAN environment:

- The DHCPv6 server is a Linux (Ubuntu 10.04)-based PC installing `dibbler-server`.
- Host1 is a Windows 7 PC.
- Host2 is a Linux (Ubuntu 12.04, kernel 3.2.12) PC.
- Host3 is a OS X Lion 10.7.3 MacBook.

Note that, we only tested M flag behavior, O flag was not included. Because O flag is about other configuration beyond address configuration, it is out of the scope of this draft.

2.2.2. Test scenarios and results

o Scenario 0

Hosts get online, no RA received.

- Windows 7: continued sending RS messages for a while, if there is no RA replied, it then began to send DHCPv6 solicit;
- Linux-kernel_3.2.12(Ubuntu 12.04): it continued sending RS, and didn't try to send DHCPv6 solicit;
- OS X Lion 10.7.3: it continued sending RS, and didn't try to send DHCPv6 solicit (just the same with Linux);

o Scenario 1

Hosts hadn't configured addresses yet, then if RA messages with M=0 received, obviously they'll do SLAAC; if M=1, which meant SLAAC and DHCPv6 were available simultaneously in the link, the behavior is as the following:

- Windows 7: using both SLAAC and DHCPv6 to configure the addresses, regardless of whether the prefixes in SLAAC/DHCPv6 are identical or not;
- Linux-kernel_3.2.12(Ubuntu 12.04): the same action with Windows 7;
- OS X Lion 10.7.3: the same action with Windows 7;

o Scenario 2

Hosts were already SLAAC-configured only, then received RA messages with M=1:

- Windows 7: using DHCPv6 to configure another address while keep the former SLAAC-configured address;
- Linux(Ubuntu 12.04): no action.(Note that, it's different with scenarios 1);
- OS X Lion 10.7.3: no action, the same with Linux;

o Scenario 3

Hosts already configured by DHCPv6 only, then received RA messages:

- Windows 7: If M=1, it configured another address with SLAAC and kept the DHCPv6 configuration; else M=0, it released the DHCPv6 address and configured with SLAAC;
- Linux-kernel_3.2.12(Ubuntu 12.04): there's no DHCPv6-only situation for it, only in scenario 1 when M=1 it would configured with SLAAC and DHCPv6 simultaneously;
- OS X Lion 10.7.3: the same situation with Linux;

o Scenario 4

Hosts already configured with SLAAC/DHCPv6 simultaneously, then RA messages with M=0 received:

- Windows 7: it released the DHCPv6 address and configured with SLAAC;
- Linux(Ubuntu 12.04): no action;
- OS X Lion 10.7.3: no action;

2.2.3. Conclusion

Obviously, the operating systems interpreting the M flag quite differently. Windows 7 treats the flag as instruction, it even released DHCPv6 session when M=0. Linux and OS X were likely to treat the flag as advisory, when SLAAC was done, it won't care about M=1, and M=0 won't cause operation for the already configured DHCPv6 addresses.

3. Requirement of Address Configuration Switching in Renumbering

During IPv6 renumbering, the SLAAC-configured hosts can reconfigure IP addresses by receiving ND Router Advertisement (RA) messages containing new prefix information. The DHCPv6-configured hosts can

reconfigure addresses by initialing RENEW sessions when the current addresses' lease time is expired or receiving the reconfiguration messages initialed by the DHCPv6 servers.

The above mechanisms have an implicit assumption that SLAAC-configured hosts will remain SLAAC while DHCPv6-managed hosts will remain DHCPv6-managed. In [I-D.ietf-6renum-enterprise], it described several renumbering scenarios in enterprise network. For example, the network may split, merge, grow, relocate or reorganize. In these situations, it is possible that SLAAC-configured hosts may need to switch to DHCPv6-managed, or verse vice.

As discussed in section 2, the semantic of M bit is ambiguous, for example, M=0 is efficient for Windows 7 PCs to switch from DHCPv6-managed to SLAAC, but for Linux or OS X it is just invalid. So in the following section 4, we proposed to use another two flags to indicate the hosts switching between SLAAC/DHCPv6.

4. Proposed Standard Update

4.1. Semantic Space of SLAAC/DHCPv6 Interaction

We summarize the semantic instructions from network side to host side as the following. Some of them are already covered by existing implementation while some may need protocol extentions.

- Network side provides both, let the hosts select by themselves

It is exactly what M=1 meaning in RFC4861.

- Network side requires the hosts to do DHCPv6 when online

As the tests showed, when get online, all the three major Oses will initial DHCPv6 when M=1. Especially, when M=1 and RAs don't include PIO (Prefix Information Option), the host would ''have to'' initial DHCPv6 for address autoconfiguration. So we can consider this semantics has been covered.

- Network side requires the already SLAAC-configured hosts to do DHCPv6

As the test showed, with current ambiguous M=1 definition, Oses varied the behaviors. So this could be considered as a semantic gap.

- Network side requires the hosts to release DHCPv6 addresses

As analyzed in section 3, the network may need the hosts to switch configuration modes. With M=0, the OS behaviors are different as the test results showed. So this is another semantic gap.

4.2. Adding a "DHCPv6Required" Flag

We propose to add a flag in the standard RA message format[RFC4861], the "DHCPv6Required" D flag, which will occupy one bit in the reserved field as showed in the following figure 2.

4.3. Adding a "ReleaseDHCPv6" Flag

We propose to add one more flag in the standard RA message format, the "ReleaseDHCPv6" R flag, which will occupy one more bit in the reserved field as showed in the following figure 2.

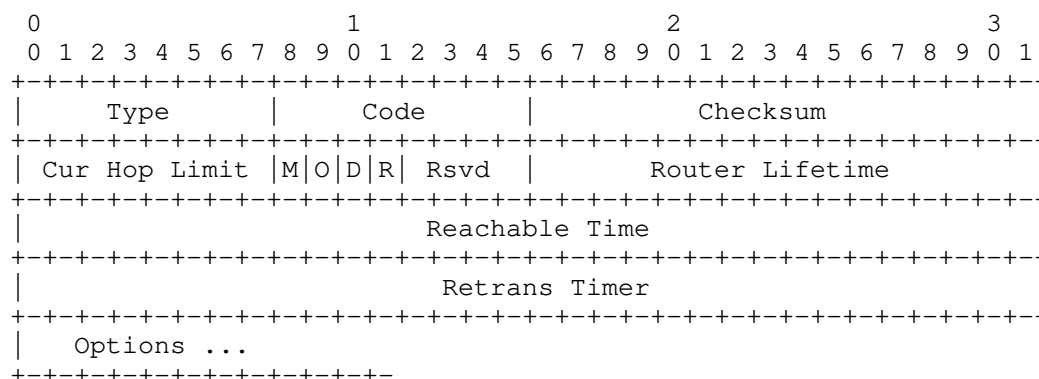


Figure 2 DHCPv6Required and ReleaseDHCPv6 flags in RA message

4.4. Host Behavior of Interpreting D/R Flag

When a host has not configured its addresses (just like scenario 0 in section 2.2) and receives RA messages with D=1, it MUST initiate a DHCPv6 stateful address autoconfiguration process.

When a host has been SLAAC-configured, and receives D=1, it MUST initiate a DHCPv6 stateful address autoconfiguration process and SHOULD deprecate SLAAC-configured addresses.

When a host has been address-configured with DHCPv6, and receives RA messages with R=1, it SHOULD release current DHCPv6 address configuration and do SLAAC.

5. Security Considerations

No more security considerations than the Neighbor Discovery protocol [RFC4861].

6. IANA Considerations

None.

7. References

7.1. Normative References

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

[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.

7.2. Informative References

[RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", RFC 5887, May 2010.

[I-D.ietf-6renum-gap-analysis]
Liu, B., and Jiang, S., "IPv6 Site Renumbering Gap Analysis", Working in Progress, March 2012

[I-D.ietf-6renum-enterprise]
Jiang, S., and B. Liu, "IPv6 Enterprise Network Renumbering Scenarios and Guidelines ", Working in Progress, March 2012.

8. Acknowledgments

The tests were done in a lab in BUPT. Thank Xudong Shi very much. He is a master student in the lab, and did a great job for the tests.

This work adopts some content from [I-D.ietf-6renum-gap-analysis].

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Bing Liu
Q14-4-A Building
Huawei Technologies Co., Ltd
Zhong-Guan-Cun Environment Protection Park, No.156 Beiqing Rd.
Hai-Dian District, Beijing
P.R. China

Email: leo.liubing@huawei.com

Wendong Wang
No.3 Teaching Building
Beijing University of Posts and Telecommunications
No.10 Xi-Tu-Cheng Rd.
Hai-Dian District, Beijing
P.R. China

Email: wdwang@bupt.edu.cn

Xiangyang Gong
No.3 Teaching Building
Beijing University of Posts and Telecommunications
No.10 Xi-Tu-Cheng Rd.
Hai-Dian District, Beijing
P.R. China

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2012

T. Savolainen
J. Nieminen
Nokia
June 18, 2012

Optimal Transmission Window Configuration Option for ICMPv6 Router
Advertisement
draft-savolainen-6man-optimal-transmission-window-00

Abstract

This specification describes an ICMPv6 Router Advertisement option for a router to configure optimal transmission window for hosts transmitting packets through the router.

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 20, 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. Introduction	3
1.1. Requirements Language	3
2. Optimal Transmission Window Option	4
3. Router Behavior	4
4. Host Behavior	5
5. Protocol Constants	6
6. IANA Considerations	7
7. Security Considerations	7
8. References	7
8.1. Normative References	7
8.2. Informative References	7
Authors' Addresses	8

1. Introduction

This document describes an ICMPv6 Router Advertisement [RFC4861] option, which the router can use in an attempt to schedule and synchronize periodical communication activities of the hosts router provides routing services for. The option describes an optimal transmission window, during which hosts should perform periodic transmissions.

In certain deployments routers are very power constrained. A class of such routers are battery powered cellular phones that are sharing the wireless cellular connection to wireless local area networks. Hosts in the local area networks may be, for example, personal computers or low-energy sensors.

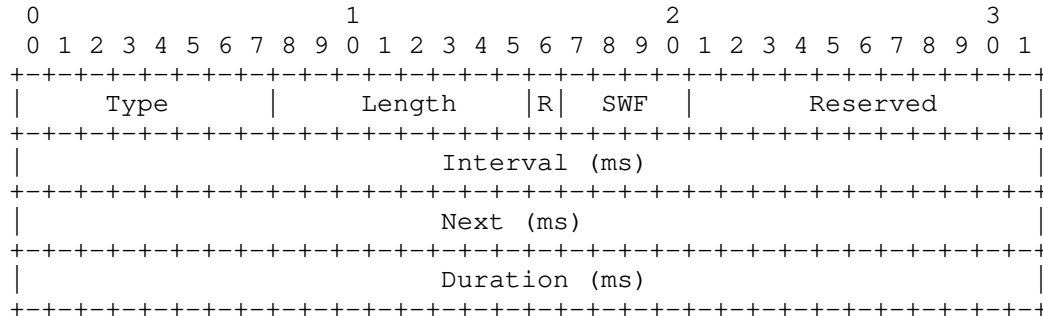
In 3GPP cellular networks the radio, once activated, stays on for some time based on network-specific timer values [Haverinen2007]. This means that, for example, a single packet originated by a host in a local area network and routed via a cellular handset can cause handset's uplink radio to be activated into a significantly power consuming state for tens of seconds.

The power consumption problem is made worse if a router provides connectivity services for multitude of hosts and, in the case of cellular handset, also provides connectivity for internal applications as well. Potentially several different entities are sending keep-alive and/or other periodic messages at random times and by so doing causing router's uplink radio to be activated unnecessarily often.

1.1. Requirements Language

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

2. Optimal Transmission Window Option



Type: TBD

Length: 2

R: If set, the optimal transmission window is open when the Router Advertisement was sent. If not set, the window may not be open.

SWF: Decimal value indicating secondary transmission window timing as fractions of Interval. Value of zero indicates lack of secondary transmission windows. Other values are used as dividers for Interval. Default value is decimal 8 (binary '1000').

Reserved: Reserved for the future, MUST be set to zero.

Interval: The time between optimal transmission windows, in milliseconds.

Next: The time to the start of the next optimal transmission window in milliseconds.

Duration: The time the optimal transmission window is open in milliseconds, for example, how long the router estimates the radio to be at least active.

3. Router Behavior

A router that attempts to synchronize periodic transmission of hosts it serves MUST include Optimal Transmission Window option in all ICMPv6 Router Advertisement messages it originates.

If the uplink radio is active at the time of sending the Router Advertisement, a router SHOULD set the R-bit on to indicate immediately suitable time for transmissions. Furthermore, in the event of uplink radio activation, a router MAY send otherwise unscheduled Router Advertisement message with R-bit set in order to indicate unscheduled power efficient transmission opportunity for hosts.

The router using this option MUST set the Interval-field to exactly match the optimal sending window, as some hosts receiving the ICMPv6 Router Advertisement can choose to go to sleep until the optimal transmission window opens. The value for the interval-field is router's implementation decision and depends on the deployment scenario. A default value of INTERVAL_DEFAULT (see Section 5) is defined for the cases where router has no better information. Interval field value of zero indicates transmission window to be always open. The SWF-field indicates presence and time of secondary transmission windows during one Interval. For example, default value of 8 indicates secondary transmission window to occur at every INTERVAL_DEFAULT/8.

With the default values for INTERVAL_DEFAULT and SWF-field hosts have secondary transmission window every 100 seconds, which is enough in case host needs to refresh UDP mappings of NAT utilizing two minute expiration time (see section 4.3 of [RFC4787]).

The Next-field MUST be always set to point to the moment of the next optimal transmission window. Even if the R-bit is set, the Next-field MUST nevertheless point to the start of the next optimal transmission window.

The Duration-field MUST indicate the length of the window during which hosts should start their periodic transmissions. The length has to be at least MIN_WINDOW_DURATION (see Section 5).

The secondary transmission window bitfield indicates possibly alternative, but still synchronized, times for hosts to transmit if the optimal sending window interval frequency is too low.

If the router implements synchronization services for router's internal applications' periodical communications, the router MUST synchronize the internal applications to communicate during the same optimal transmission window.

4. Host Behavior

A host MUST utilize the timing information received via Optimal Transmission Window option and time it's periodic transmissions accordingly, when possible. Additionally, a host MAY use Router Advertisement with this option and R-bit set as trigger for communications. The host MUST refresh it's timing states after every received Router Advertisement message.

The host MUST wait for a random period of time between the start of the optimal transmission window, or reception of a Router

Advertisement with R-bit set, and COLLISION_AVOIDANCE_DURATION (see Section 5) in order to avoid collisions caused by multitude of hosts transmitting at the same time.

Sometimes a host needs to perform time consuming operations on the link before transmitting to the Internet, such as performing Detecting Network Attachment-procedures [RFC6059] if the host has been asleep long enough. In such cases, the host SHOULD perform time consuming operations before the communications are scheduled to take place.

The host does not have to transmit during every window, but SHOULD use the one right before the application's optimal periodic communication event. If the host is running application that requires more frequent periodic messaging than what the optimal transmission window provides, the host SHOULD attempt to communicate during secondary transmission windows as configured via SWF-field.

The host MUST only use timing values as learned from the Router Advertisement message that has been used for the highest priority default router configuration. If a host supports more-specific routes [RFC4191], the host SHOULD also record optimal transmission window schedules for each more-specific route.

The host SHOULD provide an implementation specific application programming interface that applications can use to learn the optimal transmission window schedules. If the host maintains destination-specific optimal transmission window timing information, the application programming interface SHOULD allow applications to ask for the timing information specific to a destination.

The host does not have to transmit in every optimal sending window.

5. Protocol Constants

Following constants are defined for the operation of the Optimal Transmission Window Configuration option.

INTERVAL_DEFAULT: 800 seconds

MIN_WINDOW_DURATION: 1000 milliseconds

COLLISION_AVOIDANCE_DURATION: 100 milliseconds

6. IANA Considerations

This memo requests IANA to allocate a type value from the "IPv6 Neighbor Discovery Option Formats" registry for the option defined at the Section 2.

7. Security Considerations

This document specifies that a host uses timing information only from the Router Advertisements the host accepts for configuring default and more-specific routes. This helps to mitigate against attacks that try to influence transmission schedules by sending malicious Router Advertisements.

With this option it is not possible to hinder host's communications, as the option is an optimization that help nodes to synchronize transmissions with each other, while allowing transmissions at any time when necessary. Therefore, if the timing values sent in Router Advertisement do not make sense for a host, or it's applications, the values will be ignored.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

8.2. Informative References

- [Haverinen2007] Henry Haverinen, Jonne Siren, and Pasi Eronen, "Energy Consumption of Always-On Applications in WCDMA Networks", April 2007.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for
Detecting Network Attachment in IPv6", RFC 6059,
November 2010.

Authors' Addresses

Teemu Savolainen
Nokia
Hermiankatu 12 D
Helsinki FI-33720
Finland

Email: teemu.savolainen@nokia.com

Johanna Nieminen
Nokia
Itaemerenkatu 11-13
Helsinki FI-00180
Finland

Email: johanna.1.nieminen@nokia.com

