

6man Working Group
Internet-Draft
Updates: RFC 2460 (if approved)
Intended status: Standards Track
Expires: January 12, 2014

R. Bonica
Juniper Networks
W. Kumari
Google, Inc.
R. Bush
Internet Initiative Japan
H. Pfeifer
ProtocolLabs
July 11, 2013

IPv6 Fragment Header Deprecated
draft-bonica-6man-frag-deprecate-02

Abstract

This memo deprecates IPv6 fragmentation and the IPv6 fragment header. It provides reasons for deprecation and updates RFC 2460.

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 January 12, 2014.

Copyright Notice

Copyright (c) 2013 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. Case For Deprecation	3
2.1. Resource Conservation	3
2.2. Application Reliance on IPv6 Fragmentation	3
2.3. Attack Vectors	5
2.4. Operator Behavior	6
3. Applications That Rely on Fragmentation	6
3.1. DNSSEC	7
3.2. SIIT	7
3.3. OSPFv3	8
3.4. DCCP and NFS	8
3.5. Tunneling	8
4. Recommendation	8
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	11

1. Introduction

Each link on the Internet is characterized by a Maximum Transmission Unit (MTU). A link's MTU represents the maximum packet size that can be conveyed over the link, without fragmentation. IPv6 [RFC2460] requires that every link in the Internet have an MTU of 1280 octets or greater. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6.

For any given source node, the path to a particular destination is characterized by a path MTU (PMTU). At a given source, the PMTU associated with a destination is equal to the minimum MTU of all of the links in the path between the source and the destination. Because every IPv6-enabled link must support an MTU of 1280 bytes or

greater, the PMTU between any two IPv6 nodes is also 1280 bytes or greater.

[RFC2460] strongly recommends that IPv6 nodes implement Path MTU Discovery (PMTUD) [RFC1981], in order to discover and take advantage of PMTU values greater than 1280 octets. However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of PMTUD.

In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s). However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 1280 octets).

In IPv6, a packet can be fragmented only by the host that originates it. This constitutes a departure from the IPv4 [RFC0791] fragmentation strategy, in which a packet can be fragmented by its originator or by any router that it traverses en route to its destination.

This memo deprecates IPv6 fragmentation and the IPv6 fragment header. It provides reasons for deprecation and updates [RFC2460].

2. Case For Deprecation

This section presents a case for deprecating the IPv6 Fragment Header.

2.1. Resource Conservation

Packets that are fragmented at their source need to be reassembled at their destination. [Kent87] points out that the reassembly process is resource intensive. It consumes significant compute and memory resources. While the cited reference refers to IPv4 fragmentation and reassembly, many of its criticisms are equally applicable to IPv6.

By comparison, if a source node were to execute PMTUD procedures, and if applications were to avoid sending datagrams that would result in IP packets that exceed the PMTU, the task of reassembly could be avoided, altogether.

2.2. Application Reliance on IPv6 Fragmentation

Today, a limited number of applications rely upon IPv6 fragmentation.

Most popular TCP implementations include PMTUD or an extension thereof, called Packetization Layer MTU Discovery (PMTUD) [RFC4821]. Therefore, in the nominal case, applications obtaining transport services from these TCP implementations never cause IPv6 fragments to be sent. However, some TCP implementations that include PMTUD do emit segments long enough to cause IPv6 fragmentation. This happens in the following circumstance:

- o The TCP implementation establishes two (or more) sessions to the same destination
- o Because the TCP implementation has not yet emitted any long segments, the underlying IPv6 implementation estimates the PMTU for destination to be equal to the MTU of the first link in the path to the destination. This estimate is incorrect, and will be revised, below.
- o The first TCP session submits a long segment to the underlying IPv6 implementation
- o The underlying IPv6 implementation determines that if it were to encapsulate this segment in an IPv6 header, the resulting packet would not exceed its current estimate of the PMTU for the destination. So, the underlying IPv6 implementation emits a non-fragmented IPv6 packet. This packet exceeds the actual PMTU for the destination
- o A downstream router discards the long packet and returns an ICMPv6 Packet Too Big (PTB) message.
- o The first TCP session reduces its Maximum Segment Size (MSS) to an appropriate value
- o The underlying IPv6 implementation reduces its estimate of the PMTU for the destination to an appropriate value
- o The second TCP session submits a long segment to the underlying IPv6 implementation. It does so without first querying the underlying IPv6 implementation to learn its estimate of the PMTU for the destination
- o The underlying IPv6 implementation determines that if it were to encapsulate this segment in an IPv6 header, the resulting packet would exceed its current estimate of the PMTU for the destination. So, the underlying IPv6 implementation emits multiple IPv6 fragments.

The authors suggest that the behavior described above may be sub-optimal, and that TCP implementations should leverage PMTU information that the underlying IPv6 implementation could provide.

Many UDP-based [RFC0768] applications follow the recommendations of [RFC5405]. According to [RFC5405], "an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the MTU of the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement path MTU discovery itself to determine whether the path to a destination will support its desired message size without fragmentation. Applications that do not follow this recommendation to do PMTU discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending." The effective MTU for IPv6 is 1280 bytes.

However, several applications are known to rely on IPv6 fragmentation. Some of these are mentioned in Section 3.

2.3. Attack Vectors

Security researchers have found and continue to find attack vectors that rely on IP fragmentation. For example, [I-D.ietf-6man-oversized-header-chain] and [I-D.ietf-6man-nd-extension-headers] describe variants of the tiny fragment attack [RFC1858]. In this attack, a packet is crafted so that it can evade stateless firewall filters. The stateless firewall filter matches on fields drawn from the IPv6 header and an upper layer header. However, the packet is fragmented so that the upper layer header, or a significant part of that header, does not appear in the first fragment. Because a stateless firewall cannot parse payload beyond the first fragment, the packet evades detection by the firewall.

Security researcher have also studied reassembly algorithms on popular computing platforms, with the following goals:

- o to discover fragility in seldom exercised parts of the IP stack
- o to engineer flows that maximize resources consumed by the reassembly process

The Dawn and Rose Attacks [Hollis] are the products of such research.

All of the attack vectors mentioned above can be mitigated with firewalls and increasingly sophisticated reassembly algorithms.

However, the continued investment required to mitigate newly discovered vulnerabilities detracts from the cost effectiveness of IPv6 as a networking solution.

2.4. Operator Behavior

For reasons described above, and also articulated in [I-D.taylor-v6ops-fragdrop], many network operators filter all IPv6 fragments. Also, the default behavior of many currently deployed firewalls is to discard IPv6 fragments.

In one recent study [DeBoer], two researchers utilized a measurement network to measure fragment filtering. They sent packets, fragmented to the minimum MTU of 1280, to 502 IPv6 enabled and reachable probes. They found that during any given trial period, ten percent of the probes did not receive fragmented packets.

3. Applications That Rely on Fragmentation

The following is a list of applications that are currently known to rely on IPv6 fragmentation:

- o DNSSEC [RFC4035].
- o SIIT [RFC6145]
- o OSPFv3 [RFC5340]
- o NFSv4 [RFC3530]
- o DCCP [RFC4340]

Some tunneling configurations also rely upon IPv6 fragmentation. See Section 3.5 for details.

Each of these applications relies on fragmentation to a varying degree. In some cases, that reliance is essential, and cannot be broken without fundamentally changing the protocol. In other cases, that reliance is incidental, and most protocol implementations already take appropriate steps to avoid fragmentation.

Each of these applications will continue to emit IPv6 fragments, even after the IPv6 fragmentation header is deprecated. In order to achieve backwards compatibility, new IPv6 implementations will continue to support reassembly of incoming fragments. See for Section 4 details.

3.1. DNSSEC

DNSSEC can obtain transport services from either UDP or TCP. Superior performance and scaling characteristics are observed when DNSSEC runs over UDP.

When running over UDP, DNSSEC is likely to cause the generation of IPv6 fragments. By comparison, when running over TCP, DNSSEC is much less likely to cause the generation of IPv6 fragments.

When running over UDP, DNSSEC's reliance upon IPv6 fragmentation is fundamental. That reliance cannot be broken without changing the DNSSEC specification.

DNSSEC is an essential part of the Internet architecture. Therefore, this issue is for further study and must be resolved before IPv6 fragmentation can be deprecated.

3.2. SIIT

[RFC6145] requires the following:

- o "When the IPv4 sender does not set the DF bit, the translator SHOULD always include an IPv6 Fragment Header to indicate that the sender allows fragmentation. The translator MAY provide a configuration function that allows the translator not to include the Fragment Header for the non-fragmented IPv6 packets".
- o "If the DF flag is not set and the IPv4 packet will result in an IPv6 packet larger than 1280 bytes, the packet SHOULD be fragmented so the resulting IPv6 packet (with Fragment Header added to each fragment) will be less than or equal to 1280 bytes."

These behaviors cannot be changed, and for these reasons, SIIT devices will continue to emit IPv6 fragments, even after IPv6 fragmentation has been deprecated.

SIIT also emits ICMPv6 PTB messages with MTU less than 1280. In that case, the originating IPv6 node is not required to reduce the size of subsequent packets to less than 1280, but must include a Fragment header in those packets so that SIIT can obtain a suitable Identification value to use in resulting IPv4 fragments. Note that this means the payload may have to be reduced to 1232 octets (1280 minus 40 for the IPv6 header and 8 for the Fragment header), and smaller still if additional extension headers are used.

This problem could be avoided if SIIT executed an alternative procedure. For example, rather than discarding the packet and

sending an ICMPv6 PTB message with MTU less than 1280, SIIT could generate a random number for use as the Identification value and forward the packet. This issue clearly requires further study.

3.3. OSPFv3

OSPFv3 implementations may emit messages large enough to cause IPv6 fragmentation. However, in keeping with the recommendations of [RFC2460], and in order to optimize performance, most OSPFv3 implementation refrain from doing so. Many implementations simply restrict their maximum message size to some value that is safely below 1280.

3.4. DCCP and NFS

Details TBD

3.5. Tunneling

TBD

4. Recommendation

This memo deprecates IPv6 fragmentation and the IPv6 fragment header. Application and transport layer protocols SHOULD support effective PLMTUD [RFC4821], since ICMP-based PMTUD [RFC1981] is unreliable. Any application or transport layer protocol that cannot support effective PMTUD MUST NOT in any circumstances send IPv6 packets that exceed the IPv6 minimum MTU of 1280 bytes.

IPv6 stacks and forwarding nodes MUST continue to support inbound fragmented IPv6 packets as specified in [RFC2460]. However, this requirement exceeds the capability of some types of forwarding nodes such as firewalls and load balancers. Therefore implementers and operators need to be aware that on many paths through the Internet, IPv6 fragmentation will fail. Legacy applications and transport layer protocols that do not conform to the previous paragraph can expect connectivity failures as a result.

5. IANA Considerations

IANA is requested to mark the Fragment Header for IPv6 (44) as deprecated in the Protocol Numbers registry.

6. Security Considerations

Deprecation of the IPv6 Fragment Header will improve network security by eliminating attacks that rely on fragmentation.

7. Acknowledgements

The author wishes to acknowledge Tore Anderson, Mark Andrews, Brian Carpenter, Havard Eidnes, Bob Hinden, Geoff Huston, George Michaelson, Simon Perreault, Arturo Servin, Mark Smith, Fred Templin, Willem Toorop, Glen Turner and Ole Troan for their review and constructive comments.

8. References

8.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [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.
- [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.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

8.2. Informative References

- [DeBoer] De Boer, M. and J. Bosma, "Discovering Path MTU black holes on the Internet using RIPE Atlas", July 2012, <<http://www.nlnetlabs.nl/downloads/publications/pmtu-black-holes-msc-thesis.pdf>>.

- [Hollis] Hollis, K., "The Rose Attack Explained", , <http://digital.net/~gandalf/Rose_Frag_Attack_Explained.htm>.
- [I-D.ietf-6man-nd-extension-headers]
Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", draft-ietf-6man-nd-extension-headers-05 (work in progress), June 2013.
- [I-D.ietf-6man-oversized-header-chain]
Gont, F. and V. Manral, "Security and Interoperability Implications of Oversized IPv6 Header Chains", draft-ietf-6man-oversized-header-chain-02 (work in progress), November 2012.
- [I-D.ietf-6man-predictable-fragment-id]
Gont, F., "Security Implications of Predictable Fragment Identification Values", draft-ietf-6man-predictable-fragment-id-00 (work in progress), March 2013.
- [I-D.taylor-v6ops-fragdrop]
Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", draft-taylor-v6ops-fragdrop-01 (work in progress), June 2013.
- [Kent87] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology , August 1987.
- [RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, October 1995.
- [RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

Authors' Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20170
USA

Email: rbonica@juniper.net

Warren Kumari
Google, Inc.
1600 Amphitheatre Parkway
Mountainview, California 94043
USA

Email: warren@kumari.net

Randy Bush
Internet Initiative Japan
5147 Crystal Springs
Bainbridge Island Washington
USA

Email: randy@psg.com

Hagen Paul Pfeifer
Protocollabs
Munich 81379
Germany

Email: hagen.pfeifer@protocollabs.com
URI: <http://www.protocollabs.com>

6man Working Group
Internet-Draft
Updates: 3306,3956,4607,4291
(if approved)
Intended status: Standards Track
Expires: July 20, 2013

M. Boucadair
France Telecom
S. Venaas
Cisco
January 16, 2013

Updates to the IPv6 Multicast Addressing Architecture
draft-boucadair-6man-multicast-addr-arch-update-00

Abstract

This document updates the IPv6 multicast addressing architecture by defining the 17-20 reserved bits as generic flag bits. The document provides also some clarifications related to the use of these flag bits.

This document updates RFC 3956, RFC 3306, RFC 4607 and RFC 4291.

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 July 20, 2013.

Copyright Notice

Copyright (c) 2013 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. Addressing Architecture Update	3
3. Clarifications	4
3.1. Flag Bits	4
3.2. IANA Assigned SSM Block	4
4. IANA Considerations	4
5. Security Considerations	5
6. Acknowledgements	5
7. Normative References	5
Authors' Addresses	5

1. Introduction

This document updates the IPv6 multicast addressing architecture [RFC4291] by defining the 17-20 reserved bits as generic flag bits (Section 2). The document provides also some clarifications related to the use of these flag bits (Section 3.1) and also about IANA assigned SSM blocks (Section 3.2).

This document updates [RFC3956], [RFC3306], [RFC4607] and [RFC4291].

2. Addressing Architecture Update

Bits 17-20 of a multicast address are defined in [RFC3956] and [RFC3306] as reserved bits. This document defines these bits as generic flag bits so that they apply to any multicast address. Figure 1 and Figure 2 show the updated structure of the addressing architecture. The first diagram shows the update of the base IPv6 addressing architecture, and the second shows the update of so-called Embedded-RP.

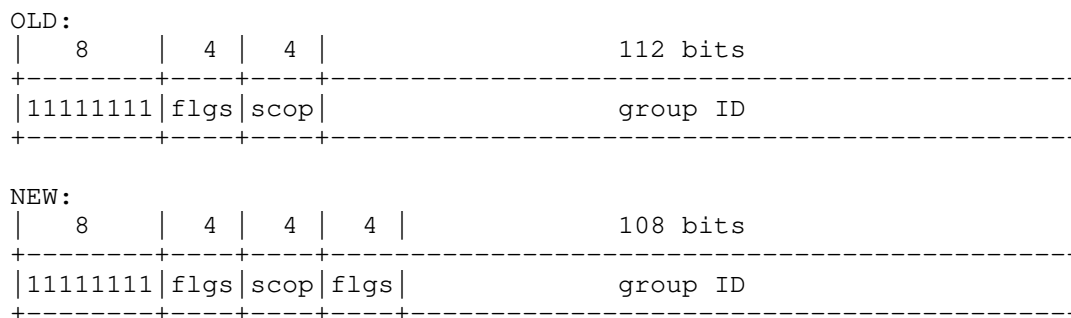


Figure 1: Updated IPv6 Multicast Addressing Architecture

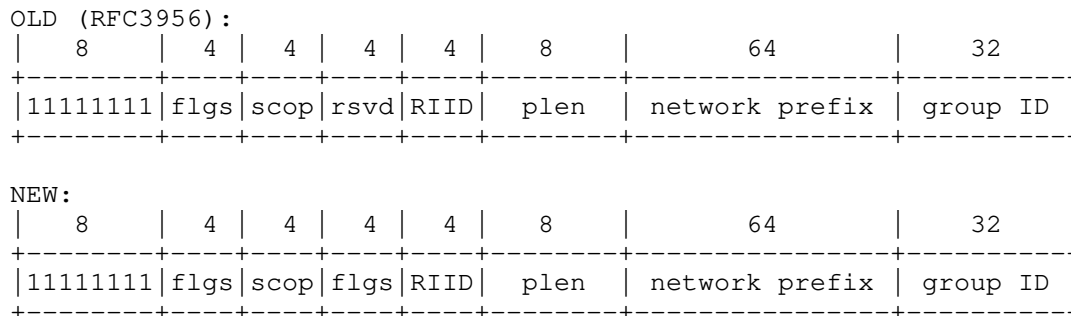


Figure 2: Embedded-RP with Updated IPv6 Multicast Address Arch.

Further specification documents may define a meaning for these flag bits. Defining the bits 17-20 as flags for all IPv6 multicast addresses allows addresses to be treated in a more uniform and generic way, and allows for these bits to be defined in the future for different purposes, irrespective of the specific type of multicast address.

3. Clarifications

3.1. Flag Bits

Some implementations and specification documents do not treat the flag bits as separate bits but tend to use their combined value as a 4-bit integer. This practice is a hurdle for assigning a meaning to the remaining flag bits. Below are listed some examples for illustration purposes:

- o the reading of [RFC4607] may lead to conclude that ff3x::/32 is the only allowed SSM IPv6 prefix block.
- o [RFC3956] states only ff70::/12 applies to Embedded-RP. Particularly, implementations should not treat the fff0::/12 range as Embedded-RP.

To avoid such confusion and to unambiguously associate a meaning with the remaining flags, the following recommendation is made

Implementations MUST treat flag bits as separate bits.

3.2. IANA Assigned SSM Block

Another issue related to SSM is the IANA assigned SSM address block. Per [RFC4607], ff3x::4000:0001 through ff3x::7fff:fff is the block for IANA assignments (<http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml>). However, IANA assignments are permanent addresses and should not have the transient bit set. Quoting from [RFC4607]:

"T = 1 indicates a non-permanently-assigned ("transient") multicast address."

4. IANA Considerations

This document may require IANA updates. However, at this point it is not clear exactly what these updates may be.

5. Security Considerations

Security considerations discussed in [RFC3956], [RFC3306], [RFC4607] and [RFC4291] MUST be taken into account.

6. Acknowledgements

Many thanks to B. Haberman for the discussions prior to the publication of this document.

7. 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.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange.com

Stig Venaas
Cisco
USA

Email: stig@cisco.com

6man WG
Internet-Draft
Updates: 4861 (if approved)
Intended status: Standards Track
Expires: August 31, 2015

S. Chakrabarti
Ericsson
E. Nordmark
Arista Networks
P. Thubert
Cisco Systems
M. Wasserman
Painless Security
February 27, 2015

IPv6 Neighbor Discovery Optimizations for Wired and Wireless Networks
draft-chakrabarti-nordmark-6man-efficient-nd-07

Abstract

IPv6 Neighbor Discovery (RFC 4861 going back to RFC 1970) was defined at a time when link-local multicast was as reliable and with the same network cost (send a packet on a yellow-coax Ethernet) as unicast and where the hosts were assumed to be always on and connected.

Since 1996 we've seen a significant change with both an introduction of wireless networks and battery operated devices, which poses significant challenges for the old assumptions. We are also seeing datacenter networks where virtual machines are not always on and connected, and scaling of multicast can be challenging.

This specification contains extensions to IPv6 Neighbor Discovery which remove most use of multicast and make sleeping hosts more efficient. The specification includes a default mixed mode where a link can have an arbitrary mix of hosts and/or routers - some implementing legacy Neighbor Discovery and some implementing the optimizations in this specification. The optimizations provide incremental benefits to hosts as soon as the first updated routers are deployed on a link.

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 31, 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	5
2. Goals and Requirements	6
2.1. Mixed-Mode Operations	7
3. Changes to ND state management	7
4. Definition Of Terms	8
5. Protocol Overview	9
5.1. Proxying to handle Mixed mode	11
6. New Neighbor Discovery Options and Messages	11
6.1. Router Advertisement flag for NEARs	11
6.2. Address Registration Option (ARO)	12
6.3. Registrar Address Option (RAO)	14
7. Conceptual Data Structures	15
8. Host Behavior	16
8.1. Host and/or Interface Initialization	16
8.2. Host Receiving Router Advertisements	16
8.3. Timing out Registrar List entries	17
8.4. Sending AROs	17
8.5. Receiving Neighbor Advertisements	18
8.6. Host Management of the TID	18
8.7. Refreshing a Registration	18
8.8. De-registering	19
8.9. Refreshing RA information	19
8.10. Sleep and Wakeup	21
8.11. Receiving Redirects	21
8.12. Movement Detection	21
9. Router Behavior	21
9.1. Router and/or Interface Initialization	22
9.2. Receiving Router Solicitations	22
9.3. Periodic Multicast RA for legacy hosts	23
9.4. Multicast RA when new information	23
9.5. Receiving ARO	23
9.6. NCE Management in NEARs	23
9.7. Sending non-zero status in ARO	24
9.8. Timing out Registered NCEs	24
9.9. Router Advertisement Consistency	25
9.10. Sending Redirects	25
9.11. Providing Information to Routing Protocols	25
9.12. Creating Legacy NCEs	25
9.13. Proxy Address Resolution and DAD for Legacy Hosts	25
10. Handling ND DoS Attack	26
11. Bootstrapping	27
12. Interaction with other protocols	28
12.1. Detecting Network Attachment (DNA)	28
12.2. DHCPv6 Interaction	28
12.3. Other use of Multicast	29
12.4. VRRP Interaction	29

13. Updated Neighbor Discovery Constants	29
14. Security Considerations	30
15. IANA Considerations	30
16. Changelog	30
17. Acknowledgements	31
18. Open Issues	32
19. References	33
19.1. Normative References	33
19.2. Informative References	33
Authors' Addresses	35

1. Introduction

IPv6 Neighbor Discovery [RFC4861] was defined at a time when local area networks had different properties than today. A common link was the yellow-coax shared wire Ethernet, where a link-layer multicast and unicast worked the same - send the packet on the wire and the interested receivers will pick it up. Thus the network cost (ignoring any processing cost on the receivers that might not completely filter out Ethernet multicast addresses that they did not want) and the reliability of sending a link-layer unicast and multicast was the same. Furthermore, the hosts at the time was always on and connected. Powering on and off the workstation/PC hosts at the time was slow and disruptive process.

Under the above assumptions it was quite efficient to maintain the shared state of the link such as the prefixes and their lifetimes using periodic multicast Router Advertisement messages. It was also efficient to use multicast Neighbor Solicitations for address resolution as a slight improvement over the broadcast use in ARP. And finally, checking for a potential duplicate IPv6 address using broadcast was efficient and believed to be likely to be robust.

Since then we've seen a tremendous change with the wide-spread deployment of WiFi and other wireless network technologies. WiFi is a case in point in that it provides the same network service abstraction as Ethernet and is often bridged with Ethernets, meaning that the Neighbor Discovery software on hosts and routers might be unaware that WiFi is being used. Yet the performance and reliability of multicast is quite different than for unicast on WiFi (see for instance [I-D.vyncke-6man-mcast-not-efficient]). Similarly 3GPP and M2M networks and devices will benefit from a standard specification for optimized Neighbor discovery. Even wired networks have evolved substantially with modern switch fabrics using explicit packet replication logic to handle multicast packets.

The assumptions about the reliability of a single multicast message for duplicate address detection has also shown to be not correct, due to a set of issues listed in [I-D.yourtchenko-6man-dad-issues].

The hosts and usage patterns has undergone radical changes as well. Hosts go to sleep when not in use to save on battery power [RFC6574] or to be more energy efficient even with mains power. The expectation is that waking up doesn't take much time and power otherwise the benefits of sleeping are greatly reduced. Initially sleeping hosts were esoteric sensor nodes, but this sleeping hosts have become mainstream in smartphones.

Some of the above trends were observed early (e.g., Ohta-san

commented on Neighbor Discovery being inefficient on WiFi a long time back) but the issues were not broadly understood. The issues were raised in the 6LowPAN context where the desire was to run IPv6 over low-power radio networks and battery operated devices. That lead to defining a set of optimizations [RFC6775] for that specific category of links. However, the trends are not limited to such specific link types.

This document applies what we have learned from 6LowPAN to all link types. That includes reusing existing support from base Neighbor Discovery (such as Redirect messages) and reusing from 6LowPAN-ND (Address Registration Option). There are additions above and beyond that to improve the robustness with redundant routers and to support mixed mode.

The optimizations are done in a way to provide incremental benefits. As soon as there is at least one router on a link which supports these optimizations, then the updated hosts on the link can sleep better, while co-existing on the same link with unmodified hosts.

2. Goals and Requirements

The goal is to remove as much Neighbor Discovery multicast traffic on the link as possible, and handle Duplicate Address Detection (DAD) without requiring the hosts to always be awake. While not an explicit goal, it turned out that the issues in [I-D.yourtchenko-6man-dad-issues] that are about robustness/correctness are also addressed as a side effect of supporting sleepy hosts.

The optimization will be highly effective for links and nodes which do not support multicast and for multicast networks without MLD snooping switches. Moreover, in the MLD-snooping networks the MLD switches will deal with less number of multicasts.

The requirements handle are:

Remove the use of multicast for DAD and Address Resolution (no multicast NS messages), and remove periodic multicast RAs. Some multicast RS and RA are needed to handle the arrival of new hosts and routers on the link since they need to bootstrap to find each other.

Remove the need for hosts to always be awake to defend their addresses by responding to any DAD probes.

Ensure that the protocol is robust against single points of failure and uses soft state which is automatically rebuilt after a state loss.

A router which does not support legacy hosts will always maintain a complete set of Neighbor Cache Entries (NCEs) for all hosts on the link. Hence there is no need for it to send Neighbor Solicitations. Thus it can avoid the problem specified in [RFC6583].

The optimized solution SHOULD be independent of the addresses allocation mechanism. In addition to supporting SLAAC [RFC4862] and DHCPv6 [RFC3315] it SHOULD also work with hosts with 'Privacy Extensions for Stateless Address Autoconfiguration in IPv6' [RFC4941] and with stable IPv6 private addresses [I-D.ietf-6man-stable-privacy-addresses] thus it handles the recommendations in [I-D.ietf-6man-default-iids].

2.1. Mixed-Mode Operations

Mixed-Mode operation is the protocol behavior when the IPv6 subnet is composed of legacy IPv6 Neighbor Discovery compliant nodes and efficiency-aware IPv6 nodes implementing this specification.

The mixed-mode model SHOULD support arbitrary combinations of legacy [RFC4861] hosts and/or routers with new hosts and/or routers on a link. The introduction of one new router SHOULD provide improved services to a new host, allowing the new host to avoid multicast and not requiring the host to be awake to defend its IPv6 addresses using DAD.

This document assumes that an implementation will have configuration knobs to determine whether it is running in legacy IPv6 ND [RFC4861] or Efficiency Aware only mode (no-legacy mode) or both (Mixed mode).

3. Changes to ND state management

These optimizations change some fundamental aspects of Neighbor Discovery. Firstly, it moves the distributed address resolution state (each node responding to a multicast NS for its own addresses) to a set of routers which maintain a list of Address Registrations for the hosts. Secondly, the information distributed in Router Advertisements changes from being periodically flooded by the routers to explicit requests from the hosts for refreshed information using unicast Router Solicitations.

4. Definition Of Terms

The keywords "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].

IPv6 ND-efficiency-aware Router (NEAR):

A router that implements the optimizations specified in this document. This router should be able to handle both legacy IPv6 nodes and nodes that sends registration request.

Efficiency-Aware Host (EAH):

The EAH is the host which implements the host functionality for optimized Neighbor Discovery mentioned in this document. At least initially implementations are likely to have a fallback to legacy Neighbor Discovery when no NEAR is on the link.

Legacy IPv6 Host:

A IPv6 host that implements [RFC4861] without the extensions in this document.

Legacy IPv6 Router:

A IPv6 router that implements [RFC4861] without the extensions in this document.

Mixed mode

A NEAR supports both legacy hosts and EAH, with a configuration knob to disable the support for legacy hosts. Some routers on the link can be legacy and some can be NEAR.

No-legacy mode

A mode configured on a NEAR to not support any legacy [RFC4861] hosts or routers. Opposite of mixed mode.

IPv6 Address Registrar

Normally the default router(s) on the link will act as IPv6 Address Registrars tracking all the EAHs. But in some cases it is more efficient to use a different set of routers as Address Registrars. The hosts are informed of the address registrars using router advertisement messages, and register with the available registrars.

EUI-64:

It is the IEEE defined 64-bit extended unique identifier formed by concatenation of 24-bit or 36-bit company id value by IEEE Registration Authority and the extension identifier within that company-id assignment. The extension identifiers are 40-bit (for 24-bit company-id) or 28-bit (for the 36-bit company-id)

respectively. The protocol supports EUI-64 for compatibility with [RFC6775].

DUID

It is a DHCP Unique ID of a device [RFC3315]. The DUID is assumed to be stable in a given IPv6 subnet. A device which does not have an EUI-64 can form and use a DUID in its address registrations.

NCE

Neighbor Cache Entry. It is a conceptual data structure introduced in section 5.1 of [RFC4861] and further elaborated in [RFC6775].

TID

The Transaction ID is a device generated sequence number used for registration. This number is used to allow a host to have concurrent registrations with different routers, while also being able to robustly replace a registration with a new one. It facilitates interoperability with protocols like RPL [RFC6550] which use a TID internally to handle host movement.

5. Protocol Overview

In a nutshell, the following basic optimizations are made from the original IPv6 Neighbor Discovery protocol [RFC4861]:

- o Adds Node Registration with the default router(s).
- o Address Resolution and DAD uses the registered addresses instead of multicast Neighbor Solicitation messages for non-link-local IPv6 addresses.
- o Does not require unsolicited periodic Router Advertisements.
- o Supports mixed-mode operation where legacy IPv6 hosts and routers and NEARs and EAHs can co-exist on the same link. This support can be configured off.

When a host powers on it behaves conforms to legacy ND [RFC4861] by multicasting up to MAX_RTR_SOLICITATIONS Router Solicitations and receives Router Advertisements. The additional information in the Router Advertisements by the NEARs is used by the EAH to build a list of IPv6 Address Registrars. As the host is forming its IPv6 addresses (using any of the many stateless and stateful IPv6 address allocation mechanism) then, instead of using a multicast DAD message, it unicasts an Neighbor Solicitation with the new Address Registration Option (ARO) to the Registrars. Assuming an IPv6

addresses are not duplicate the EAH receives a Neighbor Advertisement with the ARO option from the NEARs. The EAH refreshes the registered addresses before they expire, thereby removing the need for the EAH to be awake to defend its addresses using DAD as specified in [RFC4862] as the NEARs will handle DAD.

The routers on the link advertise the prefixes without setting the L flag. Thus only the IPv6 link-local addresses are considered on-link. Thus the hosts will send packets to a default router, and the default routers maintain all the registrations. Hence a router will know the link-layer addresses of all the registered hosts. This enables the router to forward the packet (without needing any Address Resolution with the multicast Neighbor Solicitation), and also to send a Redirect to the originating host informing the host of the link-layer address.

Instead of relying on periodic multicast RAs to refresh the lifetimes of prefixes etc., the hosts ask for refreshed information by unicasting a Router Solicitation before the information expires. Note that [I-D.nordmark-6man-rs-refresh] make that behavior more explicit by having the routers advertise a timeout.

The periodic multicast RAs may be used to provide new information such as additional prefixes, radical reduction in the preferred and/or valid lifetime for a prefix. A host does not know to ask for such information. Thus a router that wishes to quickly disseminate such change can resort to a few multicast RAs, or wait for the hosts to request a refresh using a Router Solicitation.

The routers can crash and loose all their state, including the Address Registrations. On router initialization the router will multicast a few initial RAs. The protocol has a Router Epoch mechanism which is used by the hosts to detect that the router has lost state. In that case the hosts will immediately re-register allowing the router to quickly rebuild its Address Registration state.

Normally it is sufficient for the hosts to register with all the default routers on the link. However, in some cases such as simplistic VRRP deployment the hosts should register with all the VRRP routers even though they only know of one virtual router IPv6 address. And in other cases it would be more efficient to only register with one router even though there are multiple default routers. The RAs can contain a Registrar Address Option to explicitly tell the hosts where to register.

Sleepy hosts are supported by this Neighbor Discovery procedures as they are not woken up periodically by Router Advertisement multicast

messages or Neighbor Solicitation multicast messages. Sleepy nodes may wake up in its own schedule and send unicast registration refresh messages before the registration lifetime expiration. The recommended procedure on wakeup is to unicast a Neighbor Solicitation to the default router(s), which serves as DNA check [RFC6059] that the host is on the same link, performs NUD against the router, and includes the Address Registration Option to refresh the registration.

5.1. Proxying to handle Mixed mode

When there are one or more legacy routers on the link then the recommendation is to configure those to advertise the prefixes with L=0 just as the NEARs. That results in the hosts sending all packets to a default router unless/until they receive a Redirect. However, the legacy routers do not maintain the address registrations. Thus even though all the hosts send the packets to the routers, the legacy routers might in turn need to perform Address Resolution by multicasting a Neighbor Solicitation per [RFC4861]. In addition, legacy hosts and legacy routers will perform DAD as specified in [RFC4862] that is, by sending a multicast NS and waiting for a NS or NA which indicates a conflict. A EAH will not receive and respond to such messages.

If the NEARs have been configured to operate in mixed-mode, then they will respond to multicast NS messages from legacy nodes for both DAD and Address Resolution. They will respond with the Target Link Layer Address being that of the registered host, so that subsequent communication will not go via the routers. (Implementations of "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] might proxy using their own MAC address as TLLA, but that is outside of the scope of this document.)

6. New Neighbor Discovery Options and Messages

This specification introduces a new flag in the RAs, reuses and extends the ARO option from [RFC6775] and introduces a new Registrar Address option.

6.1. Router Advertisement flag for NEARs

A new Router Advertisement flag is needed in order to distinguish a router advertisement sent by a NEAR, which will trigger an EAH to register with the NEAR. This flag is ignored by the legacy IPv6 hosts.

The current flags field in RA is reproduced here with the added 'E' bit.

```

0 1 2 3 4 5 6 7
+---+---+---+---+
|M|O|H|Prf|P|E|R|
+---+---+---+---+

```

The 'E' bit is set to 1 in a RA sent by a NEAR. In all other cases the E bit MUST be 0.

6.2. Address Registration Option (ARO)

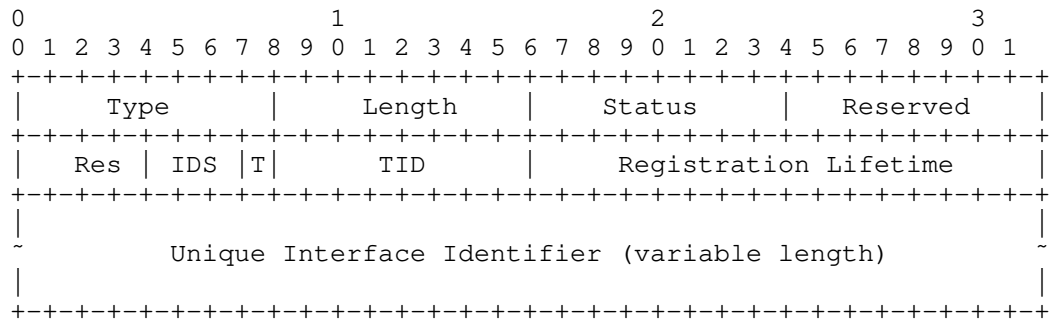
The Address Registration Option was defined in [RFC6775] for the purposes of 6LoWPAN and this document extends it in a backwards compatible way by using some of the reserved fields. The extensions are to handle different unique identifiers than EUI-64 (this document specifies how to use DHCP Unique Identifiers with the ability to use other identifier namespaces in the future) and a Transaction Id.

The Unique Interface Identifier is used by the NEARs to distinguish between a refresh of an existing registration and a different host trying to register an IPv6 address which is already registered by some other host. Thus the requirement is that the unique id is unique per link, but due to the potential for host mobility across links and subnets it should be globally unique. Both an EUI-64 and a DUID satisfies that requirement.

The TID is used by the NEARs to handle the case when due to packet loss one NEAR might have a old registration and another NEAR has a newer registration. The TID allows them to determine which is more recent. The TID also facilitates the interaction with RPL [RFC6550].

An Address Registration Option can be included in unicast Neighbor Solicitation (NS) messages sent by hosts. Thus it can be included in the unicast NS messages that a host sends as part of Neighbor Unreachability Detection to determine that it can still reach the default router(s). The ARO is used by the receiving router to reliably maintain its Neighbor Cache. The same option is included in corresponding Neighbor Advertisement (NA) messages with a Status field indicating the success or failure of the registration.

When the ARO is sent by a host then, for links which have link-layer addresses, a SLLA option MUST be included. The address that is registered is the IPv6 source address of the Neighbor Solicitation message. Typically a host would have several addresses to register, with each one being registered using a separate NS containing an ARO. (This approach facilitates applying SeND [RFC3971].)



Fields:

Type: 33 [RFC6775]

Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. The value 0 is invalid.

Status: 8-bit unsigned integer. Indicates the status of a registration in the NA response. MUST be set to 0 in NS messages. See [RFC6775].

Reserved: 8 bits. This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Res: 4 bits. This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

IDS: 3 bits. Identifier name Space. Indicates whether the Unique Interface Identifier is a DUID or or a IEEE assigned EUI-64 with room to define additional name spaces.

T bit: One bit flag. Set if the TID octet is valid.

TID: 8-bit integer. It is a transaction id maintained by the host and used by the NEARs to determine the most recent registration.

Registration Lifetime: 16-bit unsigned integer. The amount of time in a unit of 60 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option. A value of zero means to remove

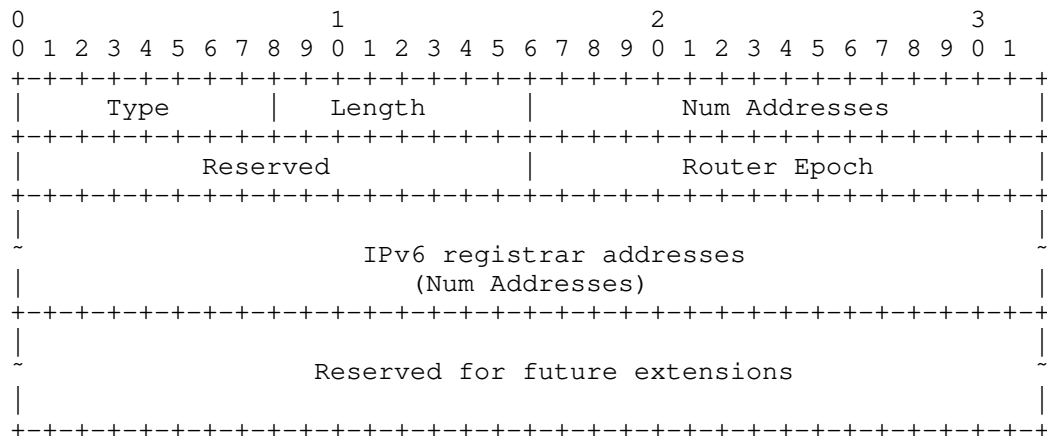
the registration.

Unique Interface Identifier: Variable length in multiples of 8 bytes. If the IDS=000, then it is an 8 byte (64 bit) unmodified EUI-64. If IDS=0011 then it is a variable length DUID. A DUID MUST be zero padded to a multiple of 8 bytes.

When a node includes ARO option in a Neighbor Solicitation it MUST, on links that have link-layer addresses, also include a SLLA option. That option is needed so that the registrar can record the link-layer address on success and send back an error if the address is a duplicate.

6.3. Registrar Address Option (RAO)

Normally the Registrars are the Default Routers. However, there are cases (like some approaches to handle VRRP, or coordinated separate routers) where there is a need to have different (and either more or less) Registrars than Default Routers. Furthermore, to robustly handle NEAR state state loss this option carries a Router Epoch which triggers the EAHs to re-register on a router epoch change. The RAO contains the information for both of those.



Fields:

Type: TBD (IANA)

Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. The value 0 is invalid.

Num Addresses 16-bit unsigned integer. Set to zero if there are no addresses i.e., when the option is used to only carry the router epoch. NumAddresses*2 + 1 must not exceed the Length.

Reserved 16-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Router epoch 16-bit integer. A router MUST pick a new epoch after a state loss, either by keeping the epoch in stable storage and incrementing it, or picking a good random number.

IPv6 registrar addresses Zero or more IPv6 addresses, typically of link-local scope.

The receiver MUST silently ignore any data after the IPv6 registrar addresses field (such data is present when the Length is greater than NumAddresses*2 + 1).

The Registrar Addresses are subject to the same lifetime as the Default Router Lifetime (thus there is no explicit lifetime field in the RAO).

7. Conceptual Data Structures

In addition to the Conceptual Data structures in [RFC4861] a EAH needs to maintain the new Registrar List for each interface. The Registrar List contains the set of IP addresses to which the host needs to send Address Registrations. Each IP address has a Router Epoch (used to determine when a router might have lost state). Also, the host MAY use this data structure to track when it needs to refresh its registrations with the registrar.

The use of explicit registrations with lifetimes plus the desire to not multicast Neighbor Solicitation messages for hosts imply that we manage the Neighbor Cache entries slightly differently than in [RFC4861]. This results in two different types of NCEs and the types specify how those entries can be removed:

Legacy: Entries that are subject to the normal rules in [RFC4861] that allow for garbage collection when low on memory. Legacy entries are created only when there is no duplicate NCE. The legacy entries are converted to the registered entries upon successful processing of ARO. Legacy type can be considered as union of

garbage-collectible and Tentative Type NCEs described in [RFC6775].

Registered: Entries that have an explicit registered lifetime and are kept until this lifetime expires or they are explicitly unregistered.

Note that the type of the NCE is orthogonal to the states specified in [RFC4861]. There can only be one type of NCE for an IP address at a time.

8. Host Behavior

A EAH follows [RFC4861] and applicable parts of [RFC6775] as specified in this section./

A EAH implementation MAY be able to fall back to [RFC4861] host behavior if there is no NEAR on the link.

8.1. Host and/or Interface Initialization

A host multicasts Router Solicitation at system startup or interface initialization as specified in [RFC4861] and its updates such as [I-D.ietf-6man-resilient-rs]. If the interface initialization is due to potential host movement or a wakeup from sleep then the host initially sends a unicast Neighbor Solicitation to the default router(s).

Unlike RFC4861 the RS MUST (on link layers which have addresses) include a SLLA option, which is used by the router to unicast the RA.

The host is not required to join the solicited-node multicast address(es) but it MUST join the all-nodes multicast address.

8.2. Host Receiving Router Advertisements

The RA is validated and processed as specified in [RFC4861] with additional behavior for RAO and the Registrar List as follows.

When a RA is received without a RAO (but with the E flag set), or the RAO contains no Registrar Addresses, then the IPv6 source address is added/updated in the Registrar List. When a RA is received with a RAO then the IPv6 Registrar Addresses in that option are added/updated in the data structure.

If those Registrar List (or entries) already exist and the Router Epoch in the RAO differs from the Router Epoch in the Registrar List

entry, or if the entry does not exist, then the host will initiate sending NS messages with ARO options to the new/updated Registration List entries. Note that if the RA contains no RAO (but the E flag is set) then for the purposes of the epoch comparison one should use a zero Router Epoch.

However, if the Default Router Lifetime in the RA is zero, then any matching Registration List entry (or entries) are instead deleted from the Registration List. It is OPTIONAL for the host to de-register when an entry is deleted from the Registration List.

The host can form its IPv6 address using any available mechanism - SLAAC, DHCPv6, temporary addresses, etc - as the registration mechanism is orthogonal and independent of the address allocation. The Address Registration procedure replaces the DAD procedure in [RFC4862].

8.3. Timing out Registrar List entries

The lifetime for the Registrar List entries are taken from the Default Router Lifetime in the RA. When an entry is removed the host MAY send AROs with a zero Registration Lifetime to the removed Registrar Addresses.

8.4. Sending AROs

When a host has formed a new IPv6 address, or when the host learns of a new NEAR and has existing IPv6 addresses, then it would register the new things (could be new addresses to all the existing Registrars, or the all the IPv6 addresses with the new Registrar. IPv6 link-local addresses are registered as well as the global addresses and ULAs.

If the EAH has a TID then it sets the T-bit and includes the TID in the ARO. When the host registers its addresses with multiple Registrars it uses the same TID. However, if the host has moved (lost its network attachment and determines it is attached to a different link using e.g., DNA [RFC6059]), then it will increment the TID value and use the new value for subsequent registrations.

The host places its Unique Interface Identifier (whether it is a DUID or EUI-64) in the ARO. This identifier is typically kept in stable storage so that the host can use the same identifier over time. It MUST use the same identifier when it re-registers its address, since otherwise all those will be returned as duplicates.

The NS which includes the ARO option MUST include a SLLA option on link layers that have link-layer addresses.

The EAH retransmits NS messages with ARO as specified in [RFC6775] until it receives a NA message from the Registrar containing an ARO. The number of such retransmissions SHOULD be configurable.

8.5. Receiving Neighbor Advertisements

The Neighbor Advertisement are validated and processed as specified in [RFC4861] for example to handle Neighbor Unreachability Detection (NUD). In addition, the host processes any received ARO as follows.

If the ARO has status code 0 (Success), then the host updates the information in the Registrar List to know when it next needs to refresh the registered address with this Registrar. No further processing is needed of the ARO.

If the ARO has status code 1 (Duplicate), then the host can not use the IPv6 address. The host follows the address allocation protocol it used to pick a new address - be that DHCPv6, temporary addresses, etc.

If the ARO has a status code 2 (Neighbor Cache Full) in response to its registration request from a Registrar, then the node SHOULD continue to register the address with other Registrars (when available).

TBD: What about other not yet defined status code values?

8.6. Host Management of the TID

It is RECOMMENDED that the EAH MAY maintain a sequence counter (TID) in stable storage according to section 7 of [RFC6550]. The TID is used to resolve conflicts between different registrations issues by the same host for the same IPv6 address. Conflicts can be due to different link-layer addresses, but it can also be due to registering with different NEARs/Registrars and those routers connect use protocols like RPL for routing, and RPL uses a TID to handle movement vs. multi-homing.

Thus the same TID should be used if the host is registering its addresses with multiple Registrars at the same time. But if the host might have moved to a different attachment point, then it should increment its TID for subsequent registrations.

8.7. Refreshing a Registration

A host SHOULD send a Registration message in order to renew its registration before its registration lifetime expires in order to continue its connectivity with the network.

This specification does not constrain the implementation. One possible implementation strategy is to attempt re-register at 1/3rd of the registration lifetime, and if no response try again at 2/3rd of the lifetime, etc. Another possible strategy is to wait until the end of the registration lifetime and then do the same relatively rapid retransmissions as for the initial registration [RFC6775]. In all cases the host SHOULD apply a random factor to its re-registration timeout to avoid self-synchronizing behavior across lots of hosts. Sleeping hosts would re-register when they are waking up to do other work.

8.8. De-registering

If anytime, the node decides that it does not need a particular default router's service anymore, then it SHOULD send a de-registration message to that NEAR/Registrar. Similarly if the host stops using a particular IPv6 address, then it SHOULD de-register that address with all the Registrars it had registered with. This applies even if the host was using the IPv6 address, then went to sleep, and then picked a different set of IPv6 addresses. In such a case it is preferred if the host de-registers before going to sleep. A mobile host SHOULD first de-register its addresses before it moves away from the subnet (if the mobile host can know that in advance of moving.)

De-registration is performed by unicasting a Neighbor Solicitation with an ARO where the Registration Lifetime is set to zero. Such an ARO should have an incremented TID. De-registration AROs are retransmitted just like other AROs as specified above.

8.9. Refreshing RA information

The EAH is responsible for asking the routers for updates to the information contained in the Router Advertisements, since the NEARs will not multicast such updates. That is done by sending unicast RSs to the router(s) which will result in unicast RAs. However, significant care is required in determining when the RSs should be transmitted.

As part of normal operation the Default Routers, Prefixes, and other RA information have lifetimes, and there are a few common cases:

1. The advertised lifetimes are constant i.e., the routers keep on advancing the time when the information will expire.
2. The routers decrement the advertised lifetimes in real time i.e., the information is set to expire at a determined time and subsequent RAs have lower and lower lifetimes.

3. The routers forcibly expire some information by advertising it with a zero lifetime for a while, and then stop advertising it.
4. A router crashes or is silently removed from the network and as a result there are no more updates. For example, that default router will expire and there is little benefit in trying to refresh it by sending lots of RSs.

The host's logic for refreshing the information needs to be careful to not send a large number of RSs, in particular if there is information that is supposed to expire at a fixed time i.e., the lifetime decrements in real time.

A host MUST NOT try to refresh information because its lifetime is near zero, since that would cause unnecessary RSs. Instead the refresh needs to be based on when the information was most recently received from the router. A lifetime of 10 minutes that was heard from the router 10 minutes ago might be normal as part of expiring some information. But a remaining lifetime of 10 minutes for a prefix that was last heard 24 hours ago with a lifetime of 24 hours means that a refresh is in order.

It is RECOMMENDED that the host track the expiry time (the wall clock time when some information will expire) and when it receives an RA with that information it SHOULD check whether the expiry time is moving forward, or appears to be frozen in time. That can tell the difference between the first two cases above, and avoid unnecessary RSs as some information naturally expires. Furthermore it is RECOMMENDED that the host track which information was received from which router, so that it can see when a router used to provide the information no longer provides it. That helps to see if the third case above might be in play. Finally, if a router has not responded to a few (e.g., MAX_RTR_SOLICITATIONS) attempts to get a refresh, then the router might be unreachable or dead, and there is little benefit in sending further RSs to that router. When the router comes back it will multicast a few RAs.

When the hosts determines that case 1 above is likely, then it should pick a reasonable time to ask for refreshes. The exact refresh behavior is not mandated by this specification, but the same implementation strategies as for refreshing address registrations in Section 8.7 can be considered.

A example simple implementation approach is to only base the refreshing on the default router lifetime (thus ignore prefix and other lifetime), and pick a refresh time which is 1/3 of the default router lifetime. If no RA is received, a subsequent refresh can be done at 2/3 of the default router lifetime. If that does not result

in a RA, then MAX_INITIAL_RTR_ADVERTISEMENTS can be sent as the router lifetime is about to expire. Note that a default router lifetime of zero MUST NOT result in sending a RS refresh based on a timeout of zero.

If the host is unable to refresh the information and as a result ends up with an empty default router list, or all the default routers are marked as UNREACHABLE by NUD, then the host MAY switch to sending initial multicast Router Solicitations as in Section 8.1.

Note that [I-D.nordmark-6man-rs-refresh] make that behavior more explicit by having the routers advertise a timeout.

8.10. Sleep and Wakeup

The protocol allows the sleepy nodes to complete its sleep schedule without waking up due to periodic Router Advertisement messages or due to Multicast Neighbor Solicitation for address resolution. The node registration lifetime SHOULD be related with its sleep interval period in order to avoid waking up in the middle of sleep for registration refresh. Depending on the application, the registration lifetime SHOULD be equal to or integral multiple of a node's sleep interval period.

When a host wakes up it can combine movement detecting (DNA), NUD, and refreshing its Address Registration by sending a unicast NS with an ARO to its existing default router(s).

8.11. Receiving Redirects

An EAH handles Redirect messages as specified in [RFC4861].

8.12. Movement Detection

When a host moves from one subnet to another its IPv6 prefix changes and the movement detection is determined according to the existing IPv6 movement detection described in [RFC6059].

9. Router Behavior

A NEAR follows [RFC4861] and applicable parts of [RFC6775] as follows in this section.

A NEAR SHOULD support legacy hosts and mixed mode as specified in this section by being able to proxy Address Resolution and DAD. The NEAR SHOULD implement a knob to be able to disable this behavior. That knob can either be set to "mixed-mode" or to "no-legacy-mode".

The RECOMMENDED default mode of operation for the NEAR is Mixed-mode.

A NEAR should be configured to advertise prefixes without the on-link (L-bit) unset. Furthermore, any legacy routers attached to the same link as a NEAR should be configured the same way. That reduces the cases in mixed mode when multicast NS messages are needed between legacy hosts and routers.

9.1. Router and/or Interface Initialization

A NEAR multicasts some initial Router Advertisements (MAX_INITIAL_RTR_ADVERTISEMENTS) at system startup or interface initialization as specified in [RFC4861] and its updates.

The NEAR MUST join the all-nodes and all-routers multicast addresses. In mixed mode it MUST also join the solicited-node multicast address(es) for its addresses and also for all the Registered NCEs.

A NEAR picks a new Router Epoch if it has lost the Registered NCEs, which is typically the case for router initialization. Either the Router Epoch can be stored in stable storage and incremented on each router initialization, or the NEAR can pick a good random number on router initialization. The effect of occasionally picking the same Router Epoch as before the state loss is that it will take longer for the router to build up its state of Registered NCEs.

9.2. Receiving Router Solicitations

Periodic RAs SHOULD be avoided. Only solicited RAs are RECOMMENDED. An RA MUST contain the Source Link-layer Address option containing Router's link-layer address (this is optional in [RFC4861]). An RA MUST carry any Prefix information option with L bit being unset, so that hosts do not multicast any NS messages as part of address resolution. A new flag (E-flag) is introduced in the RA which the hosts use to distinguish a NEAR from a legacy router.

Unlike [RFC4861] which suggests multicast Router Advertisements, this specification optimizes the exchange by always unicasting RAs in response to RSs. This is possible since the RS always includes a SLLA option, which is used by the router to unicast the RA.

If the NEAR has been configured to send an explicit set of IPv6 Registrar Addresses, or implements a Router Epoch, then the NEAR includes a RAO in all its RAs.

9.3. Periodic Multicast RA for legacy hosts

The NEAR MUST NOT send periodic RA in no-legacy mode. In mixed mode the NEAR needs to send periodic multicast RAs as specified in [RFC4861] to support legacy hosts.

9.4. Multicast RA when new information

When a router has new information to share (new prefixes, prefixes that should be immediately deprecated, etc) it MAY multicast up to MAX_INITIAL_RTR_ADVERTISEMENTS number of Router Advertisements. Note that such new information is not likely to reach sleeping hosts until those hosts refresh by sending a RS.

9.5. Receiving ARO

The NEAR follows the logic in [RFC6775] for managing the NCEs and responding to NS messages with the ARO option. The slight modification is that instead of using an EUI-64 as the key to check for duplicates, the NEAR uses the IDS value plus the variable length Unique Interface Identifier value as the key. In addition the NEAR checks the new TID field as follows.

The TID field is used together with age of a registration for arbitration between two routers to ensure freshness of the registration of a given target address. Same value of TID indicates that both states of registration are valid. In case of a mismatch between comparable TIDs, the most recent TID wins. The TIDs are compared as specified in section 7 of [RFC6550].

9.6. NCE Management in NEARs

When a host interacts with a router by sending Router Solicitations that does not match with the existing NCE entry of any type, a Legacy NCE is first created. Once a node successfully registers with a Router the result is a Registered NCE. As Routers send RAs to legacy hosts, or receive multicast NS messages from other Routers the result is Legacy NCEs.

A Router Solicitation might be received from a host that has not yet registered its address with the router or from a legacy [RFC4861] host in the Mixed-mode operation.

The router MUST NOT modify an existing Registered Neighbor Cache entry based on the SLLA option from the Router Solicitation. Thus, a router SHOULD create a tentative Legacy Neighbor Cache entry based on SLLA option when there is no match with the existing NCE. Such a legacy Neighbor Cache entry SHOULD be timed out in

TENTATIVE_LEGACY_NCE_LIFETIME seconds unless a registration converts it into a Registered NCE.

However, in 'Mixed-mode' operation, the router does not require to keep track of TENTATIVE_LEGACY_NCE_LIFETIME as it does not know if the RS request is from a legacy host or from a EAH. However, it creates the legacy type of NCE and updates it to a registered NCE if the ARO NS request arrives corresponding to the Legacy NCE. Successful processing of ARO will complete the NCE creation phase.

If ARO did not result in a duplicate address being detected, and the registration life-time is non-zero, the router creates or updates the Registered NCE for the IPv6 address. If the Neighbor Cache is full and new entries need to be created, then the router SHOULD respond with a NA with status field set to 2. For successful creation of NCE, the router SHOULD include a copy of ARO and send NA to the requester with the status field 0. A TLLA (Target Link Layer) Option is not required with this NA.

Typically for efficiency-aware routers (NEAR), the Registration Lifetime and IDS plus Unique Interface Identifier are recorded in the Neighbor Cache Entry along with the existing information described in [RFC4861]. The registered NCE are meant to be ready and reachable for communication and no address resolution is required in the link. An EAH will renew its registration to Registered NCE at the router. However the router may perform NUD towards the EAH hosts as per [RFC4861]. Should NUD fail the NEAR MUST NOT remove the Registered NCE. Instead it marks it as UNREACHABLE.

9.7. Sending non-zero status in ARO

If the Registration fails (due to it being a duplicate or the Neighbor Cache being full), then the NEAR will send an NA with ARO having a non-zero status. However, it needs to send that back to the originator of the failing ARO, and that host and link-layer address will not be present in the Neighbor Cache.

The NEAR forms a NA with ARO, and then forms the link-layer address by using the content of the SLLA option in the NS, bypassing the Neighbor Cache to send this error.

9.8. Timing out Registered NCEs

The router SHOULD NOT garbage collect Registered Neighbor Cache entries since they need to retain them until the Registration Lifetime expires. If a NEAR receives a NS message from the same host one with ARO and another without ARO then the NS message with ARO gets the precedence and the NS without ARO is ignored.

Similarly, if Neighbor Unreachability Detection on the router determines that the host is UNREACHABLE (based on the logic in [RFC4861]), the Neighbor Cache entry SHOULD NOT be deleted but be retained until the Registration Lifetime expires. If an ARO arrives for an NCE that is in UNREACHABLE state, that NCE should be marked as STALE.

The NEAR router SHOULD deny registration to a new registration request with the status code 2 when it reaches the maximum capacity for handling the neighbor cache.

9.9. Router Advertisement Consistency

The NEAR follows section 6.2.7 in [RFC4861] by receiving RAs from other routers (NEAR and legacy) on the link. In addition to the checks in that section it verifies that the prefixes do not have the L flag set, and that the Registrar Address options are consistent. Two RAOs are inconsistent if they contain the have a different Router Epoch and have some IPv6 Registration Addresses in common.

9.10. Sending Redirects

A NEAR sends redirects (with target=destination) to inform the hosts of the link-layer address of the nodes on the link.

This can be disabled on specific link types for instance, radio link technologies with hidden terminal issues.

9.11. Providing Information to Routing Protocols

If there is a routing protocols like RPL which wants visibility into the location of each IPv6 address, then this can be retrieved from the Registered NCEs on the NEAR.

9.12. Creating Legacy NCEs

In mixed-mode a NEAR will create Legacy NCEs when receiving RA, RS, and NS messages based on the source of those packets. When not in mixed-mode it needs to create Legacy NCEs for other routers by looking at those packets.

9.13. Proxy Address Resolution and DAD for Legacy Hosts

This section applies in mixed mode. It does not apply in no-legacy mode.

A NEAR in mixed mode MUST join all solicited-node for all Registered NCEs.

The NEAR SHOULD continue to support the legacy IPv6 Neighbor Solicitation requests in the mixed mode. The NEAR router SHOULD act as the ND proxy on behalf of EAH hosts for the legacy nodes' NS requests for EAH. This form of proxying is to respond with a NA that has a TLLAO taken from the Registered NCE for the target. Thus it is unlike ND Proxy as specified in [RFC4389]. (Implementations of "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] might proxy using their own MAC address as TLLA, but that is outside of the scope of this document.)

In the context of this specification, proxy means:

- o Responding to DAD probes for a registered NCE. A DAD probe from a legacy host would not contain any ARO, hence the NEAR will assume it is always a duplicate if the IPv6 target address has a Registered NCE.
- o Defending a registered address using NA messages with and ARO option and the Override bit set if the ARO option in the NS indicates either a different node (different IDS+Unique Interface Id) or a older registration (when comparing the TID).
- o Advertising a registered address using NA messages, asynchronously or as a response to a Neighbor Solicitation messages.
- o Looking up a destination on the link using Neighbor Solicitation messages in order to deliver packets arriving for the EAH.

The NEAR SHOULD also support DAD from a EAH for IPv6 address that might be in use by a legacy node. Thus when a NEAR in mixed-mode received an ARO for a new address it SHOULD perform DAD as specified in [RFC4862] by sending a multicast DAD message. Once that times out the NEAR can respond to the ARO. If a legacy host responds to the DAD probe, then the NEAR will respond to the ARO with Status=1 (Duplicate Address).

10. Handling ND DoS Attack

IETF community has discussed possible issues with /64 DoS attacks on the ND networks when an attacker host can send thousands of packets to the router with an on-link destination address or sending RS messages to initiate a Neighbor Solicitation from the neighboring router which will create a number of INCOMPLETE NCE entries for non-existent nodes in the network resulting in table overflow and denial of service of the existing communications.

The efficiency-aware behavior documented in this specification avoids

the ND DoS attacks by:

- o Having the hosts register with the default router(s).
- o Having the hosts send their packets via the default router(s).
- o Not resolving addresses for the routing solicitor by mandating SLLA option along with RS
- o Checking for duplicates in NCE before the registration
- o On-link IPv6-destinations on a particular link must be registered else these packets are not resolved and extra NCEs are not created

In order to get maximal benefits from the ND-DoS protection from Address Registrations, the hosts and routers on the link need to be upgraded to NEARs and EAHs, respectively. With some legacy hosts the routers will still need to create INCOMPLETE NCEs and send NSs, which keeps the DoS opportunity open. However, with fewer legacy hosts the lower rate limits can be applied on creation of INCOMPLETE NCEs.

11. Bootstrapping

The bootstrapping mechanism described here is applicable for the efficiency-aware hosts and routers. At the start, the host uses its link-local address to send Router Solicitation and then it sends the Address Registration Option as described in this document in order to verify the IPv6 address. Note that on wakeup from sleep and after potential movement to a different link the host initially sends a unicast Neighbor Solicitation to the default router(s).

The following step 3 and 4 SHOULD be repeated for all the IPv6 addresses that are used for communications.

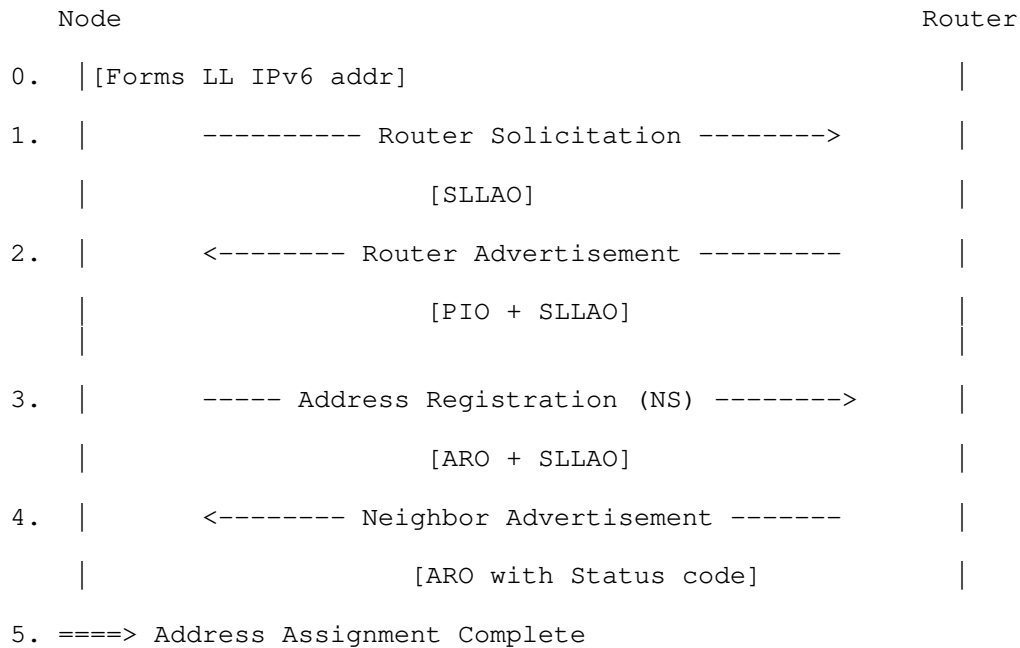


Figure 1: Neighbor Discovery Address Registration and bootstrapping

12. Interaction with other protocols

12.1. Detecting Network Attachment (DNA)

IPv6 DNA [RFC6059] uses unicast NS probes and link-layer indications to detect movement of its network attachments. That is consistent with the mechanism in this specification to unicast a NS when a host wakes up - this document recommends adding the ARO to that NS message.

Thus the ND optimization solution will work seamlessly with DNA implementations and no change is required in DNA solution because of Neighbor Discovery updates. It is a deployment and configuration consideration as to run the network in mixed mode or efficient-mode.

12.2. DHCPv6 Interaction

The protocol mechanisms in this document are orthogonal to the address assignment mechanism in use. If DHCPv6 is used for address assignment by an EAH then, if there are one or more NEARs on the subnet, the EAH will replace the DAD check specified in [RFC3315]

with Address Registration as specified in this document.

12.3. Other use of Multicast

Although the solution described in this document prevents unnecessary multicast messages in the IPv6 ND procedure, it does not affect normal IPv6 multicast packets nor the ability of nodes to join and leave the multicast group or forwarding multicast traffic or responding to multicast queries.

12.4. VRRP Interaction

A VRRP set of routers can operate with efficient-nd in two different ways:

- o Provide the illusion to hosts that they are a single router for the purposes of registrations. No RAO is needed in that case. But the pair needs some mechanism to synchronize their neighbor caches. Such a mechanism is out of scope of this document.
- o Have the hosts register with each router independently. In that case the VRRP router includes the RAO with the individual IP addresses of the routers in the pair. No synchronization of the neighbor caches are needed in that case.

13. Updated Neighbor Discovery Constants

This section discusses the updated default values of ND constants based on [RFC4861] section 10. New and changed constants are listed only for efficiency-aware-nd implementation. These values SHOULD be configurable and tunable to fit implementations and deployment.

Router Constants:

MAX_RTR_ADVERTISEMENTS (NEW)	3 transmissions
MIN_DELAY_BETWEEN_RAS (CHANGED)	1 second
TENTATIVE_LEGACY_NCE_LIFETIME (NEW)	30 seconds

Host Constants:

MAX_RTR_SOLICITATION_INTERVAL (NEW)	60 seconds
-------------------------------------	------------

Also refer to [RFC6583] , [RFC7048] and [RFC6775] for further tuning of ND constants.

14. Security Considerations

These optimizations are not known to introduce any new threats against Neighbor Discovery beyond what is already documented for IPv6 [RFC3756].

Section 11.2 of [RFC4861] applies to this document as well.

This mechanism minimizes the possibility of ND /64 DoS attacks in efficiency-aware mode. See Section 10.

The mechanisms in this document work with SeND [RFC3971] in the no-legacy mode. In the mixed mode operation when a NEAR needs to respond to a legacy host sending a NS for a EAH, then SeND would not automatically fit. Potentially proxy SeND [RFC6496] could be used, but that would require explicit awareness and setup between the proxy and the proxied EAHs which seems impractical.

The mechanisms in this specification are orthogonal to the address allocation thus works as well with SLAAC and DHCPv6 as the various privacy-enhanced address allocation specifications. In particular, using an EUI-64 for the Unique Interface Identifier in this protocol does not require or assume that the IPv6 addresses will be formed using the modified EUI-64 format.

The mechanism uses a Unique Interface Identifier for the purposes of telling apart a re-registration from the same host and a duplicate/conflicting registration from a different host. That unique ID is not globally visible. Currently the protocol supports DHCPv6 DUID and EUI-64 format for this I-D, but other formats which facilitate non-linkability (such as strong random numbers large enough to be unlikely to cause collisions) can be defined.

15. IANA Considerations

A new flag (E-bit) in RA has been introduced. IANA assignment of the E-bit flag is required upon approval of this document.

This document needs a new Neighbor Discovery option type for the RAO.

16. Changelog

Changes from draft-chakrabarti-nordmark-energy-aware-nd-06:

- o Added references to dad-issues and rs-refresh.

Changes from draft-chakrabarti-nordmark-energy-aware-nd-05:

- o Fixed typos.
- o Clarified that on interface initialization after sleep or potential movement the host unicasts a NS to the default router(s).
- o Simplified the example timer handling for refreshing RA information.
- o Added handling of DAD from EAH to legacy node that was included in -04 and lost in the -05 edits.

Changes from draft-chakrabarti-nordmark-energy-aware-nd-04:

- o Significantly simplified the description of the protocol.
- o Added clarification on problem statement
- o Clarified that privacy and temporary addresses will be supported
- o Added an IDS field in the ARO to allow a DHCP Unique ID (DUID) as an alternative to EUI-64, with room to define other (pseudo) unique identifiers.
- o Allowed router redirects for NEAR.
- o Addressed some of comments made in the 6man list.
- o Added RAO to handle VRRP and similar cases when the default router list and registrar list needs to be different.
- o Added Router Epoch to cause re-registration on NEAR state loss.
- o Specified considerations for when to refresh address registrations.
- o Specified considerations for when to refresh RA information.

17. Acknowledgements

The primary idea of this document are from 6LoWPAN Neighbor Discovery document [RFC6775] and the discussions from the 6lowpan working group members, chairs Carsten Bormann and Geoff Mulligan and through our discussions with Zach Shelby, editor of the [RFC6775].

The inspiration of such a IPv6 generic document came from Margaret Wasserman who saw a need for such a document at the IOT workshop at Prague IETF.

The authors acknowledge the ND denial of service issues and key causes mentioned in the draft-halpern-6man-nddos-mitigation document by Joel Halpern. Thanks to Joel Halpern for pinpointing the problems that are now addressed in the NCE management discussion in this document.

The authors like to thank Dave Thaler, Stuart Cheshire, Jari Arkko, Ylva Jading, Niklas J. Johnsson, Reda Nedjar, Purvi Shah, Jaume Rius Riu, Fredrik Garneij, Andrew Yourtchenko, Jouni Korhonen, Suresh Krishnan, Brian Haberman, Anders Brandt, Mark Smith, Lorenzo Colitti, David Miles, Eric Vyncke, Mark ZZZ Smith, Mikael Abrahamsson, Eric Levy-Abignoli, and Carsten Bormann for their useful comments and suggestions on this work.

18. Open Issues

The known open issues are:

- o IPv6 link-local addresses are always on-link and in this version of the document that results in multicast NS messages. The technique used in 6LowPAN-ND is too restrictive (extract the link-layer address from the IID). Should we send link-locals to routers and depend on Redirect?
- o If the Router Epoch is critical then we will see a RAO in all the RAs sent by NEARs. In such a case we don't need the E-bit in the RA.
- o Editorial: Add Comparison with 6lowpan-nd and 4861?
- o Editorial: Verify and update the description in this document to make it complete removing the need to read 6LowPAN-ND.
- o When a router has new information for the RA, currently it takes a while to disseminate that to sleeping nodes as this depends on when the hosts send a RS. We could potentially improve this is we could have an "information epoch number" in the ARO sent in the NA. But that only helps if the registrations are refreshed more frequently than the RA information.
- o Future? Currently if a router changes its information, a sleeping host would not find out when it wakes up and sends the NS with ARO. That could be improved if we fit the Router Epoch in NA/ARO.

But there is no room for 16 bits.

- o A separate but related problem is with unused NCEs due to frequent IPv6 address change e.g., hosts which pick a different set of addresses each time they wake up. This document recommends that they be de-registered by the host. That could be made simpler by introducing some Host Epoch counter in the NS/ARO.

19. References

19.1. Normative References

- [I-D.ietf-6man-resilient-rs]
Krishnan, S., Anipko, D., and D. Thaler, "Packet loss resiliency for Router Solicitations", draft-ietf-6man-resilient-rs-04 (work in progress), October 2014.
- [I-D.nordmark-6man-rs-refresh]
Nordmark, E., Yourtchenko, A., and S. Krishnan, "IPv6 Neighbor Discovery Optional Unicast RS/RA Refresh", draft-nordmark-6man-rs-refresh-01 (work in progress), October 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

19.2. Informative References

- [I-D.ietf-6man-default-iids]
Gont, F., Cooper, A., Thaler, D., and W. Will, "Recommendation on Stable IPv6 Interface Identifiers", draft-ietf-6man-default-iids-02 (work in progress), January 2015.
- [I-D.ietf-6man-stable-privacy-addresses]
Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address

Autoconfiguration (SLAAC)",
draft-ietf-6man-stable-privacy-addresses-17 (work in
progress), January 2014.

[I-D.vyncke-6man-mcast-not-efficient]

Vyncke, E., Thubert, P., Levy-Abegnoli, E., and A.
Yourtchenko, "Why Network-Layer Multicast is Not Always
Efficient At Datalink Layer",
draft-vyncke-6man-mcast-not-efficient-01 (work in
progress), February 2014.

[I-D.yourtchenko-6man-dad-issues]

Yourtchenko, A., "A survey of issues related to IPv6
Duplicate Address Detection",
draft-yourtchenko-6man-dad-issues-00 (work in progress),
October 2014.

[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.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure
Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery
Proxies (ND Proxy)", RFC 4389, April 2006.

[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.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for
Detecting Network Attachment in IPv6", RFC 6059,
November 2010.

[RFC6496] Krishnan, S., Laganier, J., Bonola, M., and A. Garcia-
Martinez, "Secure Proxy ND Support for SEcure Neighbor
Discovery (SEND)", RFC 6496, February 2012.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R.,
Levis, P., Pister, K., Struik, R., Vasseur, JP., and R.

Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

[RFC6574] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, April 2012.

[RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, March 2012.

[RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, January 2014.

Authors' Addresses

Samita Chakrabarti
Ericsson
San Jose, CA
USA

Email: samita.chakrabarti@ericsson.com

Erik Nordmark
Arista Networks
Santa Clara, CA
USA

Email: nordmark@arista.com

Pascal Thubert
Cisco Systems

Email: pthubert@cisco.com

Margaret Wasserman
Painless Security

Email: mrw@painless-security.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

A. Cooper
CDT
F. Gont
Huawei Technologies
D. Thaler
Microsoft
July 15, 2013

Privacy Considerations for IPv6 Address Generation Mechanisms
draft-cooper-6man-ipv6-address-generation-privacy-00.txt

Abstract

This document discusses privacy and security considerations for several IPv6 address generation mechanisms, both standardized and non-standardized. It evaluates how different mechanisms mitigate different threats and the trade-offs that implementors, developers, and users face in choosing different addresses or address generation mechanisms.

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 16, 2014.

Copyright Notice

Copyright (c) 2013 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. Terminology	4
3. Weaknesses in IEEE-identifier-based IIDs	4
3.1. Correlation of activities over time	4
3.2. Location tracking	5
3.3. Address scanning	6
3.4. Device-specific vulnerability exploitation	6
4. Privacy and security properties of address generation mechanisms	6
4.1. Single-address scenarios	7
4.1.1. Static, manually configured address only	8
4.1.2. Cryptographically generated address only	8
4.1.3. Temporary address only	8
4.1.4. Persistent random address only	8
4.1.5. Random-per-network address only	9
4.1.6. DHCPv6 address only	9
4.2. Multiple-address scenarios	9
4.2.1. Temporary addresses and IEEE-identifier-based address	10
4.2.2. Temporary addresses and persistent random address	11
4.2.3. Temporary addresses and random-per-network addresses	11
5. Other Privacy Issues	11
6. Miscellaneous Issues with IPv6 addressing	12
6.1. Network Operation	12
6.2. Compliance	12
6.3. Intellectual Property Rights (IPRs)	12
7. Security Considerations	12
8. IANA Considerations	13
9. Acknowledgements	13
10. Informative References	13
Authors' Addresses	15

1. Introduction

IPv6 was designed to improve upon IPv4 in many respects, and mechanisms for address assignment were one such area for improvement. In addition to static address assignment and DHCP, stateless autoconfiguration was developed as a less intensive, fate-shared means of performing address assignment. With stateless autoconfiguration, routers advertise on-link prefixes and hosts generate their own interface identifiers (IIDs) to complete their addresses. Over the years, many interface identifier generation techniques have been defined, both standardized and non-standardized:

- o Manual configuration
 - * IPv4 address
 - * Service port
 - * Wordy
 - * Low-byte
- o Stateless Address Auto-Configuration (SLAAC)
 - * IEEE 802 48-bit MAC or IEEE EUI-64 identifier [RFC1972][RFC2464]
 - * Cryptographically generated [RFC3972]
 - * Persistent random [Microsoft]
 - * Temporary (also known as "privacy addresses") [RFC4941]
 - * Random-per-network (also known as "stable privacy addresses") [I-D.ietf-6man-stable-privacy-addresses]
- o DHCPv6-based [RFC3315]
- o Specified by transition/co-existence technologies
 - * IPv4 address and port [RFC4380]

Deriving the IID from a globally unique IEEE identifier [RFC2462] was one of the earliest mechanisms developed. A number of privacy and security issues related to the interface IDs derived from IEEE identifiers were discovered after their standardization, and many of the mechanisms developed later aimed to mitigate some or all of these weaknesses. This document identifies four types of threats against IEEE-identifier-based IIDs, and discusses how other existing techniques for generating IIDs do or do not mitigate those threats.

2. Terminology

This section clarifies the terminology used throughout this document.

Stable address:

An address that does not vary over time within the same network. Note that [RFC4941] refers to these as "public" addresses, but "stable" is used here for reasons explained in Section 4.2.

Temporary address:

An address that varies over time within the same network.

Public address:

An address that has been published on some sort of directory service, such as the DNS [RFC1034].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words take their normative meanings only when they are presented in ALL UPPERCASE.

3. Weaknesses in IEEE-identifier-based IIDs

There are a number of privacy and security implications that exist for hosts that use IEEE-identifier-based IIDs. This section discusses four generic attack types: correlation of activities over time, location tracking, device-specific vulnerability exploitation, and address scanning. The first three of these rely on the attacker first gaining knowledge of the target host's IID. This can be achieved by a number of different attackers: the operator of a server to which the host connects, such as a web server or a peer-to-peer server; an entity that connects to the same network as the target (such as a conference network or any public network); or an entity that is on-path to the destinations with which the host communicates, such as a network operator.

3.1. Correlation of activities over time

As with other identifiers, an IPv6 address can be used to correlate the activities of a host for at least as long as the lifetime of the address. The correlation made possible by IEEE-identifier-based IIDs is of particular concern because MAC addresses are much more permanent than, say, DHCP leases. MAC addresses tend to last roughly the lifetime of a device's network interface, allowing correlation on the order of years, compared to days for DHCP.

As [RFC4941] explains,

"[t]he use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Anytime the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. ... The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear."

IP addresses are just one example of information that can be used to correlate activities over time. DNS names, cookies [RFC6265], browser fingerprints [Panopticlick], and application-layer usernames can all be used to link a host's activities together. Although IEEE-identifier-based IIDs are likely to last at least as long or longer than these other identifiers, IIDs generated in other ways may have shorter or longer lifetimes than these identifiers depending on how they are generated. Therefore, the extent to which a host's activities can be correlated depends on whether the host uses multiple identifiers together and the lifetimes of all of those identifiers. Frequently refreshing an IPv6 address may not mitigate correlation if an attacker has access to other longer lived identifiers for a particular host. This is an important caveat to keep in mind throughout the discussion of correlation in this document. For further discussion of correlation, see Section 5.2.1 of [I-D.iab-privacy-considerations].

3.2. Location tracking

Because the IPv6 address structure is divided between a topological portion and an interface identifier portion, an interface identifier that remains constant when a host connects to different networks (as an IEEE-identifier-based IID does) provides a way for observers to track the movements of that host. In a passive attack on a mobile host, a server that receives connections from the same host over time would be able to determine the host's movements as its prefix changes.

Active attacks are also possible. An attacker that first learns the host's interface identifier by being connected to the same network segment, running a server that the host connects to, or being on-path to the host's communications could subsequently probe other networks for the presence of the same interface identifier by sending a probe packet (ICMPv6 Echo Request, or any other probe packet). Even if the host does not respond, the first hop router will usually respond with an ICMP Address Unreachable when the host is not present, and be silent when the host is present.

3.3. Address scanning

The structure of IEEE-based identifiers used for address generation can be leveraged by an attacker to reduce the target search space [I-D.ietf-opsec-ipv6-host-scanning]. The 24-bit Organizationally Unique Identifier (OUI) of MAC addresses, together with the fixed value (0xff, 0xfe) used to form a Modified EUI-64 Interface Identifier, greatly help to reduce the search space, making it easier for an attacker to scan for individual addresses using widely-known popular OUIs.

3.4. Device-specific vulnerability exploitation

IPv6 addresses that embed IEEE identifiers leak information about the device (Network Interface Card vendor, or even Operating System and/or software type), which could be leveraged by an attacker with knowledge of device/software-specific vulnerabilities to quickly find possible targets. Attackers can exploit vulnerabilities in hosts whose IIDs they have previously obtained, or scan an address space to find potential targets.

4. Privacy and security properties of address generation mechanisms

Analysis of the extent to which a particular host is protected against the threats described in Section 3 depends on how each of a host's IIDs is generated and used. In some scenarios, a host configures a single global address and uses it for all communications. In other scenarios, a host configures multiple addresses using different mechanisms and may use any or all of them. This section compares the privacy and security properties of a variety of IID generation/use scenarios. The scenarios are grouped according to whether one or more addresses are configured. The table below provides a summary of the analysis.

Mechanism(s)	Correlation	Location tracking	Address scanning	Device exploits
--------------	-------------	-------------------	------------------	-----------------

Static manual only	For address lifetime	For address lifetime	NP	Depends on generation mechanism
CGA only	Within single network	NP	NP	NP
Temporary only	NP	NP	NP	NP
Persistent random only	For address lifetime	For address lifetime	NP	NP
Random-per-network only	Within single network	NP	NP	NP
Temporary and IEEE-based	When IEEE-based is in use, or for temp address lifetime	Possible	Possible	Possible
Temporary and persistent random	When random is in use, or for temp address lifetime	Possible	Possible	Possible
Temporary and random-per-network	Within single network, or for temp address lifetime	NP	NP	NP

Legend: NP = Not possible

Table 1: Privacy and security properties of IPv6 address generation mechanisms

4.1. Single-address scenarios

4.1.1. Static, manually configured address only

Because static, manually configured addresses are persistent, both correlation and location tracking are possible for the life of the address.

The extent to which location tracking can be successfully performed depends, to a some extent, on the uniqueness of the employed Interface ID. For example, one would expect "low byte" Interface IDs to be more widely reused than, for example, Interface IDs where the whole 64-bits follow some pattern that is unique to a specific organization. Widely reused Interface IDs will typically lead to false positives when performing location tracking.

Because they do not embed OUIs, manually configured addresses are not vulnerable to device-specific exploitation. Whether they are vulnerable to address scanning depends on the specifics of how they are generated.

4.1.2. Cryptographically generated address only

Cryptographically generated addresses (CGAs) [RFC3972] bind a hash of the host's public key to an IPv6 address in the SEcure Neighbor Discovery (SEND) [RFC3971] protocol. CGAs are uniquely generated for each subnet prefix, which means that correlation is possible within a single network only. A host that stays connected to the same network could therefore be tracked at length, whereas a mobile host's activities could only be correlated for the duration of each network connection. Location tracking is not possible with CGAs. CGAs also do not allow device-specific exploitation or address scanning attacks.

4.1.3. Temporary address only

A host that uses only a temporary address mitigates all four threats. Its activities may only be correlated for the lifetime a single address.

4.1.4. Persistent random address only

Although a mechanism to generate a static, random IID has not been standardized, it has been in wide use for many years on at least one platform (Windows). Windows uses the [RFC4941] random generation mechanism in lieu of generating an IEEE-identifier-based IID. This mitigates the device-specific exploitation and address scanning attacks, but still allows correlation and location tracking because the address is persistent across networks and time.

4.1.5. Random-per-network address only

[I-D.ietf-6man-stable-privacy-addresses] specifies a mechanism that generates a unique random IID for each network. A host that stays connected to the same network could therefore be tracked at length, whereas a mobile host's activities could only be correlated for the duration of each network connection. Location tracking is not possible with these addresses. They also do not allow device-specific exploitation or address scanning attacks.

4.1.6. DHCPv6 address only

TBD

4.2. Multiple-address scenarios

[RFC3041] (later obsoleted by [RFC4941]) sought to address some of the problems described in Section 3 by defining "temporary addresses" (commonly referred to as "privacy addresses") for outbound connections. Temporary addresses are meant to supplement the other IIDs that a device might use, not to replace them. They are randomly generated and change daily by default. The idea was for temporary addresses to be used for outgoing connections (e.g. web browsing) while maintaining the ability to use a stable address when more address stability is desired (e.g., in DNS advertisements).

[RFC3484] originally specified that stable addresses be used for outbound connections unless an application explicitly prefers temporary addresses. The default preference for stable addresses was established to avoid applications potentially failing due to the short lifetime of temporary addresses or the possibility of a reverse look-up failure or error. However, [RFC3484] allowed that "implementations for which privacy considerations outweigh these application compatibility concerns MAY reverse the sense of this rule" and instead prefer by default temporary addresses rather than stable addresses. Indeed most implementations (notably including Windows) chose to default to temporary addresses for outbound connections since privacy was considered more important (and few applications supported IPv6 at the time, so application compatibility concerns were minimal). [RFC6724] then obsoleted [RFC3484] and changed the default to match what implementations actually did.

The envisioned relationship in [RFC3484] between stability of an address and its use in "public" can be misleading when conducting privacy analysis. The stability of an address and the extent to which it is linkable to some other public identifier are independent of one another. For example, there is nothing that prevents a host from publishing a temporary address in a public place, such as the

DNS. Publishing both a stable address and a temporary address in the DNS or elsewhere where they can be linked together by a public identifier allows the host's activities when using either address to be correlated together.

Moreover, because temporary addresses were designed to supplement other addresses generated by a host, the host may still configure a more stable address even if it only ever intentionally uses temporary addresses (as source addresses) for communication to off-link destinations. An attacker can probe for the stable address even if it is never used as such a source address or advertised (e.g., in DNS or SIP) outside the link.

The scenarios in this section describe the privacy and security properties in cases where a host configures both a temporary address and an address generated via another mechanism. The analysis distinguishes between cases when both addresses are used as source addresses or are advertised off-link and cases where only the temporary address is used in those manners.

[TO DO: Add in Temporary + manual, Temporary + DHCP, Temporary + other link-layer-derived, Temporary + CGA, and perhaps re-arrange this section to avoid repetition.]

4.2.1. Temporary addresses and IEEE-identifier-based address

By using an IEEE-identifier-based IID and temporary addresses, a host can be vulnerable to the same attacks as if temporary addresses were not in use, although the viability of some of them depends on how the host uses each address. An attacker can correlate all of the host's activities for which it uses its IEEE-identifier-based IID. Once an attacker has obtained the IEEE-identifier-based IID, location tracking becomes possible on other networks even if the host only makes use of temporary addresses on those other networks; the attacker can actively probe the other networks for the presence of the IEEE-identifier-based IID. Device-specific vulnerabilities can still be exploited. Address scanning is also still possible because the IEEE-identifier-based address can be probed.

A host that configures but does not use an IEEE-identifier-based IID is vulnerable to address scanning because the address can be probed even if the IEEE-identifier-based address is otherwise never used. Once an attacker has received such a response, it can exploit device-specific vulnerabilities or probe other networks over time to track the location of the host. Correlation over time, however, is significantly mitigated, since the temporary addresses that the host routinely uses on the network refresh often.

4.2.2. Temporary addresses and persistent random address

Using a persistent, random address as a stable address for server-like connections together with temporary addresses for outbound connections presents some improvements over the previous scenario. The address scanning and device-specific exploitation attacks are no longer possible because the OUI is no longer embedded in any of the host's addresses. However, correlation of some activities across time and location tracking are both still possible because the random IID is persistent. As in Section 4.2.1, once an attacker has obtained the host's random IID, location tracking is possible on any network by probing for that IID, even if the host only uses temporary addresses on those networks.

A host that configures but does not use a persistent random address mitigates all four threats. Correlation is only possible for the lifetime of a temporary address. The persistent random address cannot be easily discovered in an address scan (although it is available to an on-link adversary), which prevents an attacker from using it for location tracking or device-specific exploitation.

4.2.3. Temporary addresses and random-per-network addresses

When used together with temporary addresses, the random-per-network mechanism [I-D.ietf-6man-stable-privacy-addresses] improves upon the previous scenario by limiting the potential for correlation to the lifetime of the random-per-network address (which may still be lengthy for hosts that are not mobile) and eliminating the possibility for location tracking (since a different IID is generated for each subnet prefix).

As in the previous scenario, a host that configures but does not use a random-per-network address mitigates all four threats.

5. Other Privacy Issues

Since IPv6 subnets have unique prefixes, they reveal some information about the location of the subnet, just as IPv4 addresses do. Hiding this information is one motivation for using NAT in IPv6 (see RFC 5902 section 2.4).

Teredo [RFC4380] specifies a means to generate an IPv6 address from the underlying IPv4 address and port, leaving many other bits set to zero. This makes it relatively easy for an attacker to scan for IPv6 addresses by guessing the Teredo client's IPv4 address and port (which for many NATs is not randomized). For this reason, popular implementations (e.g., Windows), began deviating from the standard by including 12 random bits in place of zero bits. This modification was later standardized in [RFC5991].

6. Miscellaneous Issues with IPv6 addressing

6.1. Network Operation

It is generally agreed that IPv6 addresses that vary over time in a specific network tend to increase the complexity of event logging, trouble-shooting, enforcement of access controls and quality of service, etc. As a result, some organizations disable the use of temporary addresses [RFC4941] even at the expense of reduced privacy [Broersma].

6.2. Compliance

Major IPv6 compliance testing suites required (and still require) implementations to support MAC-derived suffixes in order to be approved as compliant. Implementations that fail to support MAC-derived suffixes are therefore largely not eligible to receive the benefits of compliance certification (e.g., use of the IPv6 logo, eligibility for government contracts, etc.). This document recommends that these be relaxed to allow other forms of address generation that are more amenable to privacy.

6.3. Intellectual Property Rights (IPRs)

Some IPv6 addressing techniques might be covered by Intellectual Property rights, which might limit their implementation in different Operating Systems. [CGA-IPR] and [KAME-CGA] discuss the IPRs on CGAs.

7. Security Considerations

This whole document concerns the privacy and security properties of different IPv6 address generation mechanisms.

8. IANA Considerations

This document does not require actions by IANA.

9. Acknowledgements

The authors would like to thank Bernard Aboba and Rich Draves.

10. Informative References

[Broersma]

Broersma, R., "IPv6 Everywhere: Living with a Fully IPv6-enabled environment", Australian IPv6 Summit 2010, Melbourne, VIC Australia, October 2010, October 2010, <http://www.ipv6.org.au/10ipv6summit/talks/Ron_Broersma.pdf>.

[CGA-IPR] IETF, "Intellectual Property Rights on RFC 3972", 2005.

[I-D.iab-privacy-considerations]

Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", draft-iab-privacy-considerations-03 (work in progress), July 2012.

[I-D.ietf-6man-stable-privacy-addresses]

Gont, F., "A method for Generating Stable Privacy-Enhanced Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC)", draft-ietf-6man-stable-privacy-addresses-10 (work in progress), June 2013.

[I-D.ietf-opsec-ipv6-host-scanning]

Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning-01 (work in progress), April 2013.

[KAME-CGA]

KAME, "The KAME IPR policy and concerns of some technologies which have IPR claims", 2005.

[Microsoft]

Microsoft, "IPv6 interface identifiers", 2013.

[Panopticlick]

Electronic Frontier Foundation, "Panopticlick", 2011.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

- [RFC1972] Crawford, M., "A Method for the Transmission of IPv6 Packets over Ethernet Networks", RFC 1972, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC5991] Thaler, D., Krishnan, S., and J. Hoagland, "Teredo Security Updates", RFC 5991, September 2010.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

Authors' Addresses

Alissa Cooper
CDT
1634 Eye St. NW, Suite 1100
Washington, DC 20006
US

Phone: +1-202-637-9800
Email: acooper@cdt.org
URI: <http://www.cdt.org/>

Fernando Gont
Huawei Technologies
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

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

Dave Thaler
Microsoft
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: +1 425 703 8835
Email: dthaler@microsoft.com

Internet Engineering Task Force
Internet-Draft
Updates: RFC 4291 (if approved)
Intended status: Standards Track
Expires: January 31, 2014

R. Droms
Cisco
July 30, 2013

IPv6 Multicast Address Scopes
draft-droms-6man-multicast-scopes-02.txt

Abstract

This document updates the definitions of IPv6 multicast scopes.

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 31, 2014.

Copyright Notice

Copyright (c) 2013 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.

1. Definition of IPv6 Multicast Address Scopes

RFC 4291 [RFC4291] defines "scop is a 4-bit multicast scope value used to limit the scope of the multicast group." scop 3 is defined as "reserved" in RFC 4291. The multicast protocol specification in draft-ietf-roll-trickle-mcast [I-D.ietf-roll-trickle-mcast] desires to use multicast scop 3 for transport of multicast traffic scoped to a RPL realm (or "domain") [RFC6550]. The use of this scop value is to accommodate a multicast scope that is greater than Link-Local but is also automatically determined by the network architecture; for example, all of the hosts and routers in a multi-link subnet RPL realm.

The following table updates the definitions in RFC 4291:

0	reserved
1	Interface-Local scope
2	Link-Local scope
3	Realm-Local scope
4	Admin-Local scope
5	Site-Local scope
6	(unassigned)
7	(unassigned)
8	Organization-Local scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	Global scope
F	reserved

The following paragraph is added as the third paragraph following the list of scop values in RFC 4291:

Realm-Local scope is the largest scope that is automatically configured, i.e., automatically derived from physical connectivity or other, non-multicast-related configuration.

2. Definition of Realm-Local scopes

The definition of any Realm-Local scope for a particular network technology should be published in an RFC. For example, such a scope definition would be appropriate for publication in an "IPv6-over-foo" RFC.

Any RFCs that include the definition of a Realm-Local scope will be listed in the IANA "IPv6 Multicast Address Scopes" registry.

3. IANA Considerations

IANA is asked to establish a sub-registry titled "IPv6 Multicast Address Scopes" in the existing "Internet Protocol version 6 (IPv6) Multicast Address Allocations" registry. The "IPv6 Multicast Address Scopes" is to be populated with the scope values given in section 1, with a note associated with scope 3 listing all RFCs that define Realm-Local scoping rules that use scope 3.

4. Security Considerations

This document has no security considerations beyond those in RFC 4291 [RFC4291].

5. References

5.1. Normative References

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

5.2. Informative References

[I-D.ietf-roll-trickle-mcast]
Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-04 (work in progress), February 2013.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

Author's Address

Ralph Droms
Cisco
1414 Massachusetts Avenue
Boxborough, MA 01719
US

Phone: +1 978 936 1674
Email: rdroms@cisco.com

INTERNET-DRAFT

N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan
October 20, 2014

Intended Status: Proposed Standard
Expires: April 2015

IPv6 Performance and Diagnostic Metrics Destination Option
draft-elkins-6man-ipv6-pdm-dest-option-09

Abstract

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. This document describes an implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header. A discussion of the field limits, calculations, and usage of the PDM in measurement is described in a companion document.

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/lid-abstracts.html>

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

Copyright and License Notice

Copyright (c) 2014 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	Terminology	3
2	Performance and Diagnostic Metrics Destination Option	3
2.1	Destination Options Header	3
2.2	Performance and Diagnostic Metrics Destination Option	4
2.3	Time Base	5
2.4	Timer-value scaling	7
2.5	Header Placement	8
2.6	Implementation Considerations	9
2.6.1	Dynamic Configuration Options	9
2.6.2	Data Length Filtering	9
2.6.3	5-tuple Aging	9
3	Backward Compatibility	10
4	Security Considerations	10
5	IANA Considerations	10
6	References	10
6.2	Informative References	11
7	Acknowledgments	11
	Authors' Addresses	11

1 Introduction

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. This document describes an implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header. A discussion of the field limits, calculations, and usage of the PDM in measurement is described in a companion document: "IPPM Considerations for the IPv6 PDM Destination Option" [ELK-IPPM]. [ELK-IPPM] discusses measurement of end-user Quality of Experience (QoE) as well as the details of the scaling factors used. The current document describes the fields in the PDM header itself.

As defined in RFC2460 [RFC2460], destination options are carried by the IPv6 Destination Options extension header. Destination options include optional information that need be examined only by the IPv6 node given as the destination address in the IPv6 header, not by routers or other "middle boxes". This document specifies a new destination option, the Performance and Diagnostic Metrics (PDM) destination option.

1.1 Terminology

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

2 Performance and Diagnostic Metrics Destination Option

2.1 Destination Options Header

The IPv6 Destination Options Header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options Header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC2460 [RFC2460].

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the IPv6 Destination Options extension header (Next Header value = 60). The PDM does not require time synchronization.

2.2 Performance and Diagnostic Metrics Destination Option

The Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

TIMEBASE : Base timer unit
 SCALEDL : Scale for Delta Last Received
 SCALEDS : Scale for Delta Last Sent
 PSNTP : Packet Sequence Number This Packet
 PSNLR : Packet Sequence Number Last Received
 DELTALR : Delta Last Received
 DELTALS : Delta Last Sent

For a full explanation of the SCALEDL, SCALEDS, DELTALR, DELTALS fields, see [ELK-IPPM].

The PDM destination option is encoded in type-length-value (TLV) format as follows:

0																1																2																3																															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																																								
Option Type																Option Length																TB																ScaledDL																ScaledS															
PSN This Packet																PSN Last Received																																																															
Delta Last Received																Delta Last Sent																																																															

Option Type

TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780]

Option Length

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 16.

Time Base

2-bit unsigned integer. It will indicate the unit measurement for this device. That is, for a value of 11 in the Time Base field, a value of 1 in the DELTA fields indicates this device has incremented the time by 1 picosecond. Similarly, for a value of 01 in the Time Base field, a DELTA value of 1 indicates an increment of 1 microsecond.

The possible values of Time Base are as follows:

- 00 - milliseconds
- 01 - microseconds
- 10 - nanoseconds
- 11 - picoseconds

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use.

Initialized at a random number and monotonically incremented for packet on the 5-tuple. The 5-tuple consists of the source and destination IP addresses, the source and destination ports, and the upper layer protocol (ex. TCP, ICMP, etc).

Operating systems MUST implement a separate packet sequence number counter per 5-tuple. Operating systems MUST NOT implement a single counter for all connections.

Note: This is consistent with the current implementation of the IPID field in IPv4 for many, but not all, stacks.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned integer. This is the PSN of the packet last received on the 5-tuple.

Scale Delta Last Received (SCALEDLR)

7-bit signed integer. This is the scaling value for the Delta Last Received (DELTALR) field. The possible values are from -64 to +63.

Scale Delta Last Sent (SCALEDLS)

7-bit signed integer. This is the scaling value for the Delta Last Sent (DELTALS) field. The possible values are from -64 to +63.

Delta Last Received (DELTALR)

A 16-bit unsigned integer field.

$\text{DELTALR} = \text{Send time packet 2} - \text{Receive time packet 1}$

The value is according to the scale in SCALEDLR.

Delta Last Sent (DELTALS)

A 16-bit unsigned integer field.

Delta Last Sent = Receive time packet 2 - Send time packet 1

The value is in according to the scale in SCALEDLS.

Option Type

The two highest-order bits of the Option Type field are encoded to indicate specific processing of the option; for the PDM destination option, these two bits MUST be set to 00. This indicates the following processing requirements:

00 - skip over this option and continue processing the header.

RFC2460 [RFC2460] defines other values for the Option Type field. These MUST NOT be used in the PDM. The other values are as follows:

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

In keeping with RFC2460 [RFC2460], the third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination.

In the PDM, the value of the third-highest-order bit MUST be 0. The possible values are as follows:

0 - Option Data does not change en-route

1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

2.3 Time Base

This specification allows for the fact that different CPU TOD clocks use different binary points. For some clocks, a value of 1 could indicate 1 microsecond, whereas other clocks could use the value 1 to indicate 1 millisecond. In the former case, the binary digits to the right of that binary point measure $2^{*(-n)}$ microseconds, and in the latter case, $2^{*(-n)}$ milliseconds.

The Time Base allows us to ensure we have a common reference, at the very least, common knowledge of what the binary point is for the transmitted values.

2.4 Timer-value scaling

As discussed in [TRAM-TCPM] we propose storing not an entire time-interval value, but just the most significant bits of that value, along with a scaling factor to indicate the magnitude of the time-interval value. In our case, we will use the high-order 16 bits. The scaling value will be the number of bits in the timer register to the right of the 16th significant bit. That is, if the timer register contains this binary value:

```
1110100011010100101001010001000000000000
<-16 bits      -><-24 bits      ->
```

then, the values stored would be 1110 1000 1101 0100 in binary (E8D4 hexadecimal) for the time value and 24 for the scaling value. Note that the displayed value is the binary equivalent of 1 second expressed in picoseconds.

The following table represents a device which has a TimeBase of picosecond (or 11). For this Time Base value the smallest and simplest time value to represent is 1 picosecond; the encoded value is 1, and the scaling value is 0. Using time values in the first column in the table below, we create the following encoded values and scaling values:

Delta time	Time value in picoseconds	Encoded value	Scaling decimal
1 picosecond	1	0001	0
1 nanosecond	3e8	03e8	0
1 microsecond	f4240	f424	4
1 millisecond	3b9aca00	3b9a	16
1 second	e8d4a51000	e8d4	24
1 minute	3691d6afc000	3691	32
1 hour	cca2e51310000	cca2	36
1 day	132f4579c980000	132f	44
365 days	1b5a660ea44b80000	1b5a	52

Sample binary values (high order 16 bits taken)

```

1 psec      1                                0001
1 nsec      3e8                                0011 1110 1000
1 usec      f4240                            1111 0100 0010 0100 0000
1 msec      3b9aca00                        0011 1011 1001 1010 1100 1010 0000 0000
1 sec      e8d4a51000 1110 1000 1101 0100 1010 0101 0001 0000 0000 0000

```

The need for a signed scaling value is more apparent when the Time Base is lower. If you specify your Time Base is microseconds (or 01) then these tables looks very similar:

Delta time	Time value in microseconds	Encoded value	Scaling decimal
1 microsecond	1	0001	0
1 millisecond	3e8	03e8	0
1 second	f4240	f424	4
1 minute	3938700	e4e1	10
1 hour	d693a400	d693	16
1 day	141dd76000	a0ee	21

An issue arises when the binary point is at the microsecond level, as it is here, but the time differential to be expressed is some number of nanoseconds or picoseconds. For example 1 nanosecond is .001 microseconds, or about .000000000100000110001001001 in binary. To encode this value we would follow the same procedure:

```

<-      25 bits      ->      Scaling value: -25
0000000000100000110001001001
      <-16 bits      ->      Encoded value: 8312

```

So, the encoded value would be 8312, with a scaling value of -25 which, in the signed 7-bit SCALEDs or SCALEDR field would be represented as 1100111 in binary.

Similarly, if the Time Base is 10, indicating the clock is counting nanoseconds, the encoded value and scaling values for 1 picosecond, or .001 nanoseconds are the same, 8312 and -25, because we've changed the Time Base.

For further discussion on timing and scaling, please see "IPPM Considerations for the IPv6 PDM Destination Option" [ELK-IPPM].

2.5 Header Placement

The PDM destination option MUST be placed as follows:

- Before the upper-layer header. That is, this is the last extension header.

This follows the order defined in RFC2460 [RFC2460]

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header
- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header
- Destination Options header
- upper-layer header

For each IPv6 packet header, the PDM MUST NOT appear more than once. However, an encapsulated packet MAY contain a separate PDM associated with each encapsulated IPv6 header.

The inclusion of a PDM in a packet affects the receiving node's processing of only this single packet. No state is created or modified in the receiving node as a result of receiving a PDM in a packet.

2.6 Implementation Considerations

The PDM destination options extension header SHOULD be turned on by each stack on a host node.

2.6.1 Dynamic Configuration Options

If implemented, each operating system MUST have a default configuration parameter, e.g. `diag_header_sys_default_value=yes/no`. The operating system MAY also have a dynamic configuration option to change the configuration setting as needed.

If the PDM destination options extension header is used, then it MAY be turned on for all packets flowing through the host, applied to an upper-layer protocol (TCP, UDP, SCTP, etc), a local port, or IP address only. These are at the discretion of the implementation.

The PDM MUST NOT be changed dynamically via packet flow as this may create potential security violation or DoS attack by numerous packets turning the header on and off.

As with all other destination options extension headers, the PDM is for destination nodes only. As specified above, intermediate devices MUST neither set nor modify this field.

2.6.2 Data Length Filtering

Different results for derived metrics, such as, server delay, will be obtained if calculations are done including or excluding packets which have a data length of 0 or 1. Some protocols, for example, TCP, provide acknowledgements which have a length of 0. Keep-alive packets have a data length of 0 or 1.

Operating systems may provide the user a choice of whether to include or exclude packets with a 0 or 1 byte data length.

2.6.3 5-tuple Aging

Within the operating system, metrics must be kept on a 5-tuple basis. The 5-tuple is:

- SADDR : IP address of the sender
- SPORT : Port for sender
- DADDR : IP address of the destination
- DPORT : Port for destination
- PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP)

The question comes of when to stop keeping data or restarting the numbering for a 5-tuple. For example, in the case of TCP, at some point, the connection will terminate. Keeping data in control blocks forever, will have unfortunate consequences for the operating system.

So, the recommendation is to use a known aging parameter such as Max Segment Lifetime (MSL) as defined in Transmission Control Protocol [RFC0793]. The choice of aging parameter is left up to the implementation.

3 Backward Compatibility

The scheme proposed in this document is backward compatible with all the currently defined IPv6 extension headers. According to RFC2460 [RFC2460], if the destination node does not recognize this option, it should skip over this option and continue processing the header.

4 Security Considerations

The PDM MUST NOT be changed dynamically via packet flow as this creates a possibility for potential security violations or DoS attacks by numerous packets turning the header on and off.

5 IANA Considerations

An option type must be assigned by IANA for the Performance and Diagnostic Metrics (PDM) destination option.

6 References

6.1 Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[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.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

6.2 Informative References

[TRAM-TCPM] Trammel, B., "Encoding of Time Intervals for the TCP Timestamp Option-01", Internet Draft, July 2013. [Work in Progress]

[ELK-IPPM] Elkins, N., "IPPM Considerations for the IPv6 PDM Destination Option-02", Internet Draft, October 2014.

7 Acknowledgments

The authors would like to thank Keven Haining, Bill Jouris, Sigfrido Perdomo, and Al Morton for their comments.

Authors' Addresses

Nalini Elkins
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA 93924
United States
Phone: +1 831 659 8360
Email: nalini.elkins@insidethestack.com
<http://www.insidethestack.com>

Robert M. Hamilton
Chemical Abstracts Service
A Division of the American Chemical Society
2540 Olentangy River Road
Columbus, Ohio 43202
United States
Phone: +1 614 447 3600 x2517
Email: rhamilton@cas.org
<http://www.cas.org>

Michael S. Ackermann
Blue Cross Blue Shield of Michigan
P.O. Box 2888
Detroit, Michigan 48231
United States
Phone: +1 310 460 4080
Email: mackermann@bcbsmi.com
<http://www.bcbsmi.com>

IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: Standards Track
Expires: July 31, 2014

F. Gont
SI6 Networks / UTN-FRH
January 27, 2014

A Method for Generating Semantically Opaque Interface Identifiers with
IPv6 Stateless Address Autoconfiguration (SLAAC)
draft-ietf-6man-stable-privacy-addresses-17

Abstract

This document specifies a method for generating IPv6 Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), such that addresses configured using this method are stable within each subnet, but the Interface Identifier changes when hosts move from one network to another. This method is meant to be an alternative to generating Interface Identifiers based on hardware addresses (e.g., IEEE LAN MAC addresses), such that the benefits of stable addresses can be achieved without sacrificing the privacy of users. The method specified in this document applies to all prefixes a host may be employing, including link-local, global, and unique-local addresses.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 31, 2014.

Copyright Notice

Copyright (c) 2014 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. Terminology	5
3. Relationship to Other standards	5
4. Design goals	5
5. Algorithm specification	6
6. Resolving Duplicate Address Detection (DAD) conflicts	11
7. Specified Constants	12
8. IANA Considerations	12
9. Security Considerations	12
10. Acknowledgements	14
11. References	15
11.1. Normative References	15
11.2. Informative References	16
Appendix A. Possible sources for the Net_Iface parameter	18
A.1. Interface Index	18
A.2. Interface Name	18
A.3. Link-layer Addresses	19
A.4. Logical Network Service Identity	19
Author's Address	19

1. Introduction

[RFC4862] specifies Stateless Address Autoconfiguration (SLAAC) for IPv6 [RFC2460], which typically results in hosts configuring one or more "stable" addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) that typically embeds a hardware address (e.g., an IEEE LAN MAC address) [RFC4291]. Cryptographically Generated Addresses (CGAs) [RFC3972] are yet another method for generating Interface Identifiers, which bind a public signature key to an IPv6 address in the SEcure Neighbor Discovery (SEND) [RFC3971] protocol.

Generally, the traditional SLAAC addresses are thought to simplify network management, since they simplify Access Control Lists (ACLs) and logging. However, they have a number of drawbacks:

- o since the resulting Interface Identifiers do not vary over time, they allow correlation of node activities within the same network,

thus negatively affecting the privacy of users (see [I-D.ietf-6man-ipv6-address-generation-privacy] and [IAB-PRIVACY]).

- o since the resulting Interface Identifiers are constant across networks, the resulting IPv6 addresses can be leveraged to track and correlate the activity of a node across multiple networks (e.g. track and correlate the activities of a typical client connecting to the public Internet from different locations), thus negatively affecting the privacy of users.
- o since embedding the underlying link-layer address in the Interface Identifier will result in specific address patterns, such patterns may be leveraged by attackers to reduce the search space when performing address scanning attacks [I-D.ietf-opsec-ipv6-host-scanning]. For example, the IPv6 addresses of all nodes manufactured by the same vendor (within a given time frame) will likely contain the same IEEE Organizationally Unique Identifier (OUI) in the Interface Identifier.
- o embedding the underlying hardware address in the Interface Identifier leaks device-specific information that could be leveraged to launch device-specific attacks.
- o embedding the underlying link-layer address in the Interface Identifier means that replacement of the underlying interface hardware will result in a change of the IPv6 address(es) assigned to that interface.

[I-D.ietf-6man-ipv6-address-generation-privacy] provides additional details regarding how these vulnerabilities could be exploited, and the extent to which the method discussed in this document mitigates them.

The "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941] (henceforth referred to as "temporary addresses") were introduced to complicate the task of eavesdroppers and other information collectors (e.g., IPv6 addresses in web server logs or email headers, etc.) to correlate the activities of a node, and basically result in temporary (and random) Interface Identifiers. These temporary addresses are generated in addition to the traditional IPv6 addresses based on IEEE LAN MAC addresses, with the "temporary addresses" being employed for "outgoing communications", and the traditional SLAAC addresses being employed for "server" functions (i.e., receiving incoming connections).

It should be noted that temporary addresses can be challenging in a number of areas. For example, from a network-management point of view, they tend to increase the complexity of event logging, troubleshooting, enforcement of access controls and quality of service, etc. As a result, some organizations disable the use of temporary addresses even at the expense of reduced privacy [Broersma]. Temporary addresses may also result in increased implementation complexity, which might not be possible or desirable in some implementations (e.g., some embedded devices).

In scenarios in which temporary addresses are deliberately not used (possibly for any of the aforementioned reasons), all a host is left with is the stable addresses that have typically been generated from the underlying hardware addresses. In such scenarios, it may still be desirable to have addresses that mitigate address scanning attacks, and that at the very least do not reveal the node's identity when roaming from one network to another -- without complicating the operation of the corresponding networks.

However, even with "temporary addresses" in place, a number of issues remain to be mitigated. Namely,

- o since "temporary addresses" [RFC4941] do not eliminate the use of fixed identifiers for server-like functions, they only partially mitigate host-tracking and activity correlation across networks (see [I-D.ietf-6man-ipv6-address-generation-privacy] for some example attacks that are still possible with temporary addresses).
- o since "temporary addresses" [RFC4941] do not replace the traditional SLAAC addresses, an attacker can still leverage patterns in SLAAC addresses to greatly reduce the search space for "alive" nodes [Gont-DEEPSEC2011] [CPNI-IPv6] [I-D.ietf-opssec-ipv6-host-scanning].

Hence, there is a motivation to improve the properties of "stable" addresses regardless of whether temporary addresses are employed or not.

This document specifies a method to generate Interface Identifiers that are stable/constant for each network interface within each subnet, but that change as hosts move from one network to another, thus keeping the "stability" properties of the Interface Identifiers specified in [RFC4291], while still mitigating address-scanning attacks and preventing correlation of the activities of a node as it moves from one network to another.

2. Terminology

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

3. Relationship to Other standards

The method specified in this document is orthogonal to the use of "temporary" addresses [RFC4941], since it is meant to improve the security and privacy properties of the stable addresses that are employed along with the aforementioned "temporary" addresses. In scenarios in which "temporary addresses" are employed, implementation of the mechanism described in this document (in replacement of stable addresses based on e.g., IEEE LAN MAC addresses) will mitigate address-scanning attacks and also mitigate the remaining vectors for correlating host activities based on the node's constant (i.e. stable across networks) Interface Identifiers. On the other hand, for nodes that currently disable "temporary addresses" [RFC4941], implementation of this mechanism would mitigate the host-tracking and address scanning issues discussed in Section 1.

While the method specified in this document is meant to be used with SLAAC, this does not preclude this algorithm from being used with other address configuration mechanisms, such as DHCPv6 [RFC3315] or manual address configuration.

4. Design goals

This document specifies a method for generating Interface Identifiers to be used with IPv6 SLAAC, with the following goals:

- o The resulting Interface Identifiers remain stable for each prefix used with SLAAC within each subnet for the same network interface. That is, the algorithm generates the same Interface Identifier when configuring an address (for the same interface) belonging to the same prefix within the same subnet.
- o The resulting Interface Identifiers must change when addresses are configured for different prefixes. That is, if different autoconfiguration prefixes are used to configure addresses for the same network interface card, the resulting Interface Identifiers must be (statistically) different. This means that, given two addresses produced by the method specified in this document, it must be difficult for an attacker tell whether the addresses have been generated/used by the same node.

- o It must be difficult for an outsider to predict the Interface Identifiers that will be generated by the algorithm, even with knowledge of the Interface Identifiers generated for configuring other addresses.
- o Depending on the specific implementation approach (see Section 5 and Appendix A), the resulting Interface Identifiers may be independent of the underlying hardware (e.g. IEEE LAN MAC address). This means that e.g. replacing a Network Interface Card (NIC) or adding links dynamically to a Link Aggregation Group (LAG) will not have the (generally undesirable) effect of changing the IPv6 addresses used for that network interface.
- o The method specified in this document is meant to be an alternative to producing IPv6 addresses based hardware addresses (e.g. IEEE LAN MAC addresses, as specified in [RFC2464]). That is, this document does not formally obsolete or deprecate any of the existing algorithms to generate Interface Identifiers. It is meant to be employed for all of the stable (i.e. non-temporary) IPv6 addresses configured with SLAAC for a given interface, including global, link-local, and unique-local IPv6 addresses.

We note that this method is incrementally deployable, since it does not pose any interoperability implications when deployed on networks where other nodes do not implement or employ it. Additionally, we note that this document does not update or modify IPv6 Stateless Address Auto-Configuration (SLAAC) [RFC4862] itself, but rather only specifies an alternative algorithm to generate Interface Identifiers. Therefore, the usual address lifetime properties (as specified in the corresponding Prefix Information Options) apply when IPv6 addresses are generated as a result of employing the algorithm specified in this document with SLAAC [RFC4862]. Additionally, from the point of view of renumbering, we note that these addresses behave like the traditional IPv6 addresses (that embed a hardware address) resulting from SLAAC [RFC4862].

5. Algorithm specification

IPv6 implementations conforming to this specification MUST generate Interface Identifiers using the algorithm specified in this section in replacement of any other algorithms used for generating "stable" addresses with SLAAC (such as those specified in [RFC2464], [RFC2467], and [RFC2470]). However, implementations conforming to this specification MAY employ the algorithm specified in [RFC4941] to generate temporary addresses in addition to the addresses generated with the algorithm specified in this document. The method specified in this document MUST be employed for generating the Interface

Identifiers with SLAAC for all the stable addresses, including IPv6 global, link-local, and unique-local addresses.

Implementations conforming to this specification SHOULD provide the means for a system administrator to enable or disable the use of this algorithm for generating Interface Identifiers.

Unless otherwise noted, all of the parameters included in the expression below MUST be included when generating an Interface Identifier.

1. Compute a random (but stable) identifier with the expression:

$RID = F(\text{Prefix}, \text{Net_Iface}, \text{Network_ID}, \text{DAD_Counter}, \text{secret_key})$

Where:

RID:

Random (but stable) Identifier

F():

A pseudorandom function (PRF) that MUST NOT be computable from the outside (without knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret_key, even when given samples of the output of F() and knowledge or control of the other input parameters. F() SHOULD produce an output of at least 64 bits. F() could be implemented as a cryptographic hash of the concatenation of each of the function parameters. SHA-1 [FIPS-SHS] and SHA-256 are two possible options for F(). Note: MD5 [RFC1321] is considered unacceptable for F() [RFC6151].

Prefix:

The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message, or the link-local IPv6 unicast prefix [RFC4291].

Net_Iface:

An implementation-dependent stable identifier associated with the network interface for which the RID is being generated. An implementation MAY provide a configuration option to select the source of the identifier to be used for the Net_Iface parameter. A discussion of possible sources for this value (along with the corresponding trade-offs) can be found in Appendix A.

Network_ID:

Some network specific data that identifies the subnet to which this interface is attached. For example the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. Additionally, Simple DNA [RFC6059] describes ideas that could be leveraged to generate a Network_ID parameter. This parameter is OPTIONAL.

DAD_Counter:

A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts. It MUST be initialized to 0, and incremented by 1 for each new tentative address that is configured as a result of a DAD conflict. Implementations that record DAD_Counter in non-volatile memory for each {Prefix, Net_Iface, Network_ID} tuple MUST initialize DAD_Counter to the recorded value if such an entry exists in non-volatile memory. See Section 6 for additional details.

secret_key:

A secret key that is not known by the attacker. The secret key MUST be initialized to a pseudo-random number (see [RFC4086] for randomness requirements for security) at operating system installation time or when the IPv6 protocol stack is initialized for the first time. An implementation MAY provide the means for the system administrator to display and change the secret key.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit.

We note that [RFC4291] requires that, the Interface IDs of all unicast addresses (except those that start with the binary value 000) be 64-bit long. However, the method discussed in this document could be employed for generating Interface IDs of any arbitrary length, albeit at the expense of reduced entropy (when employing Interface IDs smaller than 64 bits).

The resulting Interface Identifier SHOULD be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID], and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, this situation SHOULD be handled in the same way as the case of duplicate addresses (see Section 6).

This document does not require the use of any specific PRF for the function F() above, since the choice of such PRF is usually a trade-

off between a number of properties (processing requirements, ease of implementation, possible intellectual property rights, etc.), and since the best possible choice for $F()$ might be different for different types of devices (e.g. embedded systems vs. regular servers) and might possibly change over time.

Including the SLAAC prefix in the PRF computation causes the Interface Identifier to vary across each prefix (link-local, global, etc.) employed by the node and, as consequently, also across networks. This mitigates the correlation of activities of multi-homed nodes (since each of the corresponding addresses will employ a different Interface ID), host-tracking (since the network prefix will change as the node moves from one network to another), and any other attacks that benefit from predictable Interface Identifiers (such as IPv6 address scanning attacks).

The `Net_Iface` is a value that identifies the network interface for which an IPv6 address is being generated. The following properties are required for the `Net_Iface` parameter:

- o it MUST be constant across system bootstrap sequences and other network events (e.g., bringing another interface up or down)
- o it MUST be different for each network interface simultaneously in use

Since the stability of the addresses generated with this method relies on the stability of all arguments of $F()$, it is key that the `Net_Iface` be constant across system bootstrap sequences and other network events. Additionally, the `Net_Iface` must uniquely identify an interface within the node, such that two interfaces connecting to the same network do not result in duplicate addresses. Different types of operating systems might benefit from different stability properties of the `Net_Iface` parameter. For example, a client-oriented operating system might want to employ `Net_Iface` identifiers that are attached to the NIC, such that a removable NIC always gets the same IPv6 address, irrespective of the system communications port to which it is attached. On the other hand, a server-oriented operating system might prefer `Net_Iface` identifiers that are attached to system slots/ports, such that replacement of a network interface card does not result in an IPv6 address change. Appendix A discusses possible sources for the `Net_Iface`, along with their pros and cons.

Including the optional `Network_ID` parameter when computing the RID value above causes the algorithm to produce a different Interface Identifier when connecting to different networks, even when configuring addresses belonging to the same prefix. This means that a host would employ a different Interface Identifier as it moves from

one network to another even for IPv6 link-local addresses or Unique Local Addresses (ULAs) [RFC4193]. In those scenarios where the `Network_ID` is unknown to the attacker, including this parameter might help mitigate attacks where a victim node connects to the same subnet as the attacker, and the attacker tries to learn the Interface Identifier used by the victim node for a remote network (see Section 9 for further details).

The `DAD_Counter` parameter provides the means to intentionally cause this algorithm to produce a different IPv6 addresses (all other parameters being the same). This could be necessary to resolve Duplicate Address Detection (DAD) conflicts, as discussed in detail in Section 6.

Note that the result of `F()` in the algorithm above is no more secure than the secret key. If an attacker is aware of the PRF that is being used by the victim (which we should expect), and the attacker can obtain enough material (i.e. addresses configured by the victim), the attacker may simply search the entire secret-key space to find matches. To protect against this, the secret key SHOULD be of at least 128 bits. Key lengths of at least 128 bits should be adequate. The secret key is initialized at system installation time to a pseudo-random number, thus allowing this mechanism to be enabled/used automatically, without user intervention. Providing a mechanism to display and change the `secret_key` would allow and administrator to cause a replaced system (with the same implementation of this document) to generate the same IPv6 addresses as the system being replaced. We note that since the privacy of the scheme specified in this document relies on the secrecy of the `secret_key` parameter, implementations should constrain access to the `secret_key` parameter to the extent practicable (e.g., require superuser privileges to access it). Furthermore, in order to prevent leakages of the `secret_key` parameter, it should not be used for any other purposes than being a parameter to the scheme specified in this document.

We note that all of the bits in the resulting Interface IDs are treated as "opaque" bits [I-D.ietf-6man-ug]. For example, the universal/local bit of Modified EUI-64 format identifiers is treated as any other bit of such identifier. In theory, this might result in IPv6 address collisions and Duplicate Address Detection (DAD) failures that would otherwise not be encountered. However, this is not deemed as a likely issue, because of the following considerations:

- o The interface IDs of all addresses (except those of addresses that start with the binary value 000) are 64-bit long. Since the method specified in this document results in random Interface IDs, the probability of DAD failures is very small.

- o Real world data indicates that MAC address reuse is far more common than assumed [HDMoore]. This means that even IPv6 addresses that employ (allegedly) unique identifiers (such as IEEE LAN MAC addresses) might result in DAD failures, and hence implementations should be prepared to gracefully handle such occurrences. Additionally, some virtualization technologies already employ hardware addresses that are randomly selected, and hence cannot be guaranteed to be unique [I-D.ietf-opsec-ipv6-host-scanning].
- o Since some popular and widely-deployed operating systems (such as Microsoft Windows) do not embed hardware addresses in the Interface IDs of their stable addresses, reliance on such unique identifiers is more reduced in the deployed world (fewer deployed systems rely on them for the avoidance of address collisions).

Finally, that since different implementation are likely to use different values for the `secret_key` parameter, and may also employ different PRFs for `F()` and different sources for the `Net_Iface` parameter, the addresses generated by this scheme should not expected to be stable across different operating system installations. For example, a host that is dual-boot or that is reinstalled may result in different IPv6 addresses for each operating system and/or installation.

6. Resolving Duplicate Address Detection (DAD) conflicts

If as a result of performing Duplicate Address Detection (DAD) [RFC4862] a host finds that the tentative address generated with the algorithm specified in Section 5 is a duplicate address, it SHOULD resolve the address conflict by trying a new tentative address as follows:

- o `DAD_Counter` is incremented by 1.
- o A new Interface Identifier is generated with the algorithm specified in Section 5, using the incremented `DAD_Counter` value.

Hosts SHOULD introduce a random delay between 0 and `IDGEN_DELAY` seconds (see Section 7) before trying a new tentative address, to avoid lock-step behavior of multiple hosts.

This procedure may be repeated a number of times until the address conflict is resolved. Hosts SHOULD try at least `IDGEN_RETRIES` (see Section 7) tentative addresses if DAD fails for successive generated addresses, in the hopes of resolving the address conflict. We also note that hosts MUST limit the number of tentative addresses that are

tried (rather than indefinitely try a new tentative address until the conflict is resolved).

In those unlikely scenarios in which duplicate addresses are detected and in which the order in which the conflicting nodes configure their addresses may vary (e.g., because they may be bootstrapped in different order), the algorithm specified in this section for resolving DAD conflicts could lead to addresses that are not stable within the same subnet. In order to mitigate this potential problem, nodes MAY record the DAD_Counter value employed for a specific {Prefix, Net_Iface, Network_ID} tuple in non-volatile memory, such that the same DAD_Counter value is employed when configuring an address for the same Prefix and subnet at any other point in time. We note that the use of non-volatile memory is OPTIONAL, and hosts that do not implement this feature are still compliant to this protocol specification.

In the event that a DAD conflict cannot be solved (possibly after trying a number of different addresses), address configuration would fail. In those scenarios, nodes MUST NOT automatically fall back to employing other algorithms for generating Interface Identifiers.

7. Specified Constants

This document specifies the following constant:

IDGEN_RETRIES:
defaults to 3.

IDGEN_DELAY:
defaults to 1 second.

8. 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.

9. Security Considerations

This document specifies an algorithm for generating Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), as an alternative to e.g. Interface Identifiers that embed hardware addresses (such as those specified in [RFC2464], [RFC2467], and [RFC2470]). When compared to such identifiers, the identifiers specified in this document have a number of advantages:

- o They prevent trivial host-tracking based on the IPv6 address, since when a host moves from one network to another the network prefix used for autoconfiguration and/or the Network ID (e.g., IEEE 802.11 SSID) will typically change, and hence the resulting Interface Identifier will also change (see [I-D.ietf-6man-ipv6-address-generation-privacy]).
- o They mitigate address-scanning techniques which leverage predictable Interface Identifiers (e.g., known Organizationally Unique Identifiers) [I-D.ietf-opsec-ipv6-host-scanning].
- o They may result in IPv6 addresses that are independent of the underlying hardware (i.e. the resulting IPv6 addresses do not change if a network interface card is replaced) if an appropriate source for Net_Iface (Section 5) is employed.
- o They prevent the information leakage produced by embedding hardware addresses in the Interface Identifier (which could be exploited to launch device-specific attacks).
- o Since the method specified in this document will result in different Interface Identifiers for each configured address, knowledge/leakage of the Interface Identifier employed for one stable address will not negatively affect the security/privacy of other stable addresses configured for other prefixes (whether at the same time or at some other point in time).

We note that while some probing techniques (such as the use of ICMPv6 Echo Request and ICMPv6 Echo Response packets) could be mitigated by a personal firewall at the target host, for other probing vectors, such as listening to ICMPv6 "Destination Unreachable, Address Unreachable" (Type 1, Code 3) error messages referring to the target addresses [I-D.ietf-opsec-ipv6-host-scanning], there is nothing a host can do (e.g., a personal firewall at the target host would not be able to mitigate this probing technique). Hence, the method specified in this document is still of value for nodes that employ personal firewalls.

In scenarios in which an attacker can connect to the same subnet as a victim node, the attacker might be able to learn the Interface Identifier employed by the victim node for an arbitrary prefix, by simply sending a forged Router Advertisement [RFC4861] for that prefix, and subsequently learning the corresponding address configured by the victim node (either listening to the Duplicate Address Detection packets, or to any other traffic that employs the newly configured address). We note that a number of factors might limit the ability of an attacker to successfully perform such an attack:

- o First-Hop security mechanisms such as RA-Guard [RFC6105] [I-D.ietf-v6ops-ra-guard-implementation] could prevent the forged Router Advertisement from reaching the victim node
- o If the victim implementation includes the (optional) Network_ID parameter for computing F() (see Section 5), and the Network_ID employed by the victim for a remote network is unknown to the attacker, the Interface Identifier learned by the attacker would differ from the one used by the victim when connecting to the legitimate network.

In any case, we note that at the point in which this kind of attack becomes a concern, a host should consider employing Secure Neighbor Discovery (SEND) [RFC3971] to prevent an attacker from illegitimately claiming authority for a network prefix.

We note that this algorithm is meant to be an alternative to Interface Identifiers such as those specified in [RFC2464], but is not meant as an alternative to temporary Interface Identifiers (such as those specified in [RFC4941]). Clearly, temporary addresses may help to mitigate the correlation of activities of a node within the same network, and may also reduce the attack exposure window (since temporary addresses are short-lived when compared to the addresses generated with the method specified in this document). We note that implementation of this algorithm would still benefit those hosts employing "temporary addresses", since it would mitigate host-tracking vectors still present when such addresses are used (see [I-D.ietf-6man-ipv6-address-generation-privacy]), and would also mitigate address-scanning techniques that leverage patterns in IPv6 addresses that embed IEEE LAN MAC addresses. Finally, we note that the method described in this document addresses some of the privacy concerns arising from the use of IPv6 addresses that embed IEEE LAN MAC addresses, without the use of temporary addresses, thus possibly offering an interesting trade-off for those scenarios in which the use of temporary addresses is not feasible.

10. Acknowledgements

The algorithm specified in this document has been inspired by Steven Bellovin's work ([RFC1948]) in the area of TCP sequence numbers.

The author would like to thank (in alphabetical order) Mikael Abrahamsson, Ran Atkinson, Karl Auer, Steven Bellovin, Matthias Bethke, Ben Campbell, Brian Carpenter, Tassos Chatzithomaoglou, Tim Chown, Alissa Cooper, Dominik Elsbroek, Stephen Farrell, Eric Gray, Brian Haberman, Bob Hinden, Christian Huitema, Ray Hunter, Jouni Korhonen, Suresh Krishnan, Eliot Lear, Jong-Hyouk Lee, Andrew McGregor, Thomas Narten, Simon Perreault, Tom Petch, Michael

Richardson, Vincent Roca, Mark Smith, Hannes Frederic Sowa, Martin Stiernerling, Dave Thaler, Ole Troan, Lloyd Wood, James Woodyatt, and He Xuan, for providing valuable comments on earlier versions of this document.

Hannes Frederic Sowa produced a reference implementation of this specification for the Linux kernel.

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).

11. References

11.1. 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.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 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.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, February 2009.
- [I-D.ietf-6man-ug]
Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", draft-ietf-6man-ug-06 (work in progress), December 2013.

11.2. Informative References

- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [RFC1948] Bellare, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC2467] Crawford, M., "Transmission of IPv6 Packets over FDDI Networks", RFC 2467, December 1998.
- [RFC2470] Crawford, M., Narten, T., and S. Thomas, "Transmission of IPv6 Packets over Token Ring Networks", RFC 2470, December 1998.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.
- [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, November 2010.

- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, February 2011.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.
- [I-D.ietf-opsec-ipv6-host-scanning]
Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning-02 (work in progress), July 2013.
- [I-D.ietf-v6ops-ra-guard-implementation]
Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", draft-ietf-v6ops-ra-guard-implementation-07 (work in progress), November 2012.
- [I-D.ietf-6man-ipv6-address-generation-privacy]
Cooper, A., Gont, F., and D. Thaler, "Privacy Considerations for IPv6 Address Generation Mechanisms", draft-ietf-6man-ipv6-address-generation-privacy-00 (work in progress), October 2013.
- [HDMoore] HD Moore, , "The Wild West", Louisville, Kentucky, U.S.A, September 2012, <<https://speakerdeck.com/hdm/derbycon-2012-the-wild-west>>.
- [IANA-RESERVED-IIID]
Reserved IPv6 Interface Identifiers, ,
"http://www.iana.org/assignments/ipv6-interface-ids/ipv6-interface-ids.xml", .
- [Gont-DEEPSEC2011]
Gont, , "Results of a Security Assessment of the Internet Protocol version 6 (IPv6)", DEEPSEC 2011 Conference, Vienna, Austria, November 2011,
<<http://www.sixnetworks.com/presentations/deepsec2011/fgont-deepsec2011-ipv6-security.pdf>>.
- [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/10ipv6summit/talks/Ron_Broersma.pdf>.

[IAB-PRIVACY]

IAB, , "Privacy and IPv6 Addresses", July 2011,
<[http://www.iab.org/wp-content/IAB-uploads/2011/07/
IPv6-addresses-privacy-review.txt](http://www.iab.org/wp-content/IAB-uploads/2011/07/IPv6-addresses-privacy-review.txt)>.

[CPNI-IPv6]

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

[FIPS-SHS]

FIPS, , "Secure Hash Standard (SHS)", Federal Information
Processing Standards Publication 180-4, March 2012,
<[http://csrc.nist.gov/publications/fips/fips180-4/
fips-180-4.pdf](http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf)>.

Appendix A. Possible sources for the Net_Iface parameter

The following subsections describe a number of possible sources for the Net_Iface parameter employed by the F() function in Section 5. The choice of a specific source for this value represents a number of trade-offs, which may vary from one implementation to another.

A.1. Interface Index

The Interface Index [RFC3493] [RFC3542] of an interface uniquely identifies an interface within a node. However, these identifiers might or might not have the stability properties required for the Net_Iface value employed by this method. For example, the Interface Index might change upon removal or installation of a network interface (typically one with a smaller value for the Interface Index, when such a naming scheme is used), or when network interfaces happen to be initialized in a different order. We note that some implementations are known to provide configuration knobs to set the Interface Index for a given interface. Such configuration knobs could be employed to prevent the Interface Index from changing (e.g. as a result of the removal of a network interface).

A.2. Interface Name

The Interface Name (e.g., "eth0", "em0", etc) tends to be more stable than the underlying Interface Index, since such stability is required/desired when interface names are employed in network configuration (firewall rules, etc.). The stability properties of Interface Names depend on implementation details, such as what is the namespace used for Interface Names. For example, "generic" interface names such as "eth0" or "wlan0" will generally be invariant with respect to network interface card replacements. On the other hand, vendor-dependent

interface names such as "rtk0" or the like will generally change when a network interface card is replaced with one from a different vendor.

We note that Interface Names might still change when network interfaces are added or removed once the system has been bootstrapped (for example, consider Universal Serial Bus-based network interface cards which might be added or removed once the system has been bootstrapped).

A.3. Link-layer Addresses

Link-layer addresses typically provide for unique identifiers for network interfaces; although, for obvious reasons, they generally change when a network interface card is replaced. In scenarios where neither Interface Indexes nor Interface Names have the stability properties specified in Section 5 for `Net_Iface`, an implementation might want to employ the link-layer address of the interface for the `Net_Iface` parameter, albeit at the expense of making the corresponding IPv6 addresses dependent on the underlying network interface card (i.e., the corresponding IPv6 address would typically change upon replacement of the underlying network interface card).

A.4. Logical Network Service Identity

Host operating systems with a conception of logical network service identity, distinct from network interface identity or index, may keep a Universally Unique Identifier (UUID) [RFC4122] or similar identifier with the stability properties appropriate for use as the `Net_Iface` parameter.

Author's Address

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>

Distributed Mobility Management (DMM)
Internet-Draft
Updates: 4861, 4862 (if approved)
Intended status: Standards Track
Expires: January 11, 2014

J. Korhonen
Renesas Mobile
B. Patil
S. Gundavelli
Cisco
P. Seite
Orange
D. Liu
China Mobile
July 10, 2013

IPv6 Prefix Properties
draft-korhonen-6man-prefix-properties-02.txt

Abstract

This specification defines an extension to the IPv6 Neighbor Discovery protocol and the stateless address autoconfiguration procedure. A new Prefix Information Option with meta data is defined that describe the properties and other prefix class meta data associated with the prefix. The stateless address autoconfiguration procedure and end hosts can make use of the additional properties and class information when selecting prefixes for a particular uses and use cases. This specification updates RFC4861 and also updates RFC4862.

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 January 11, 2014.

Copyright Notice

Copyright (c) 2013 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Background and Motivation	5
3. Option Formats	6
3.1. Prefix Information Option with Meta Data	6
3.2. Meta Data Suboptions	7
4. Host Considerations	9
4.1. Stateless Address Autoconfiguration Enhancements	9
4.2. Internal Data Structures	9
4.3. Default Address Selection	9
5. Router Considerations	10
6. Security Considerations	10
7. IANA Considerations	10
8. Acknowledgements	11
9. References	11
9.1. Normative References	11
9.2. Informative References	11
Authors' Addresses	12

1. Introduction

This specification defines a new neighbor discovery protocol message option, the Prefix Information Option with Meta Data (PIOMT), that indicate, for example, the mobility management properties associated to the prefix, and a class value that conveys metadata associated to the prefix with a local administrative domain wide importance. Furthermore, the specification discusses corresponding source address selection hint flags to the IPv6 Socket API for Source Address Selection [RFC5014].

For example, the IPv6 Socket API for Source Address Selection [RFC5014] already covers Mobile IPv6 [RFC6275] and allows selecting between a home address (HoA) and a care-of address (CoA). A mobile node (MN) with a client based mobility IP stack is supposed to know which prefixes are CoA(s) and/or HoA(s). However, this is not the case with network based mobility management where the MN is expected to be agnostic of the mobility support. There has been attempts in past to define similar functionality for the mobility protocols purposes [I-D.damic-6man-pmip6-ind]. Outside the mobility protocols, there are other potential use cases where associating properties to the advertised prefixes could be useful as discussed in [I-D.bhandari-dhc-class-based-prefix].

The extensions to [RFC4861] are minimal in a sense that they do not define new functionality, for example, to any existing mobility protocol but instead add an explicit indication of network based mobility knowledge into the IPv6 stateless address autoconfiguration (SLAAC) [RFC4862]. The new functionality is achieved by defining a new, backward compatible, option to IPv6 router advertisements that convey the required prefix related meta data information the SLAAC procedure may take use of.

This would allow for network based mobility solutions, such as Proxy Mobile IPv6 [RFC5213] or GTP [TS.29274] to explicitly indicate that their prefixes have mobility, and therefore, the MN IP stack can make an educated selection between prefixes that have mobility and those that do not. There is also a potential need to extend both [RFC3493] and [RFC5014] in order to provide required hooks into socket APIs.

The underlying assumption is that a MN has multiple prefixes to choose from. Typically this means either the MN has multiple interfaces or an interface has been configured with multiple prefixes. This specification does not make a distinction between these alternatives and does not either make any assumptions how the possible transfer of a prefix is done between interfaces in the case a network based mobility solution is used.

2. Background and Motivation

This section discusses the motivations behind adding metadata and other address selection decision making affecting information into IPv6 prefixes. The additional information is conveyed from the network to a end host during the IPv6 address configuration phase. The motivation example taken from and discussed below is from the mobile networks.

IP mobility and its centralized topological anchoring of IP addresses has known issues. For instance, non-optimal routing is a classical example. Another concerns include excessive tunneling, increased signaling due the maintenance of mobility related bindings, aggregation of traffic to centralized mobility anchor gateways and unnecessary IP mobility related state management for IP traffic that does not as such benefit from mobility. In general, it is observed that most applications do not need IP level mobility, and work just fine with "temporary" IP addresses that come and go. However, IP mobility still has its virtues making the applications unaware of mobility, and certain wireless mobile networking architecture make extensive use of network based IP mobility.

In order to overcome some of the above issues, use of local resources and topologically local addressing could be enhanced. In many cases this would lead to use of multiple addresses of which some provide mobility and some do not. However, an end host has to have means to distinguish between addresses that provide mobility, and those that are short lived and usable only within a limited topological area.

[I-D.seite-dmm-dma] and [I-D.draft-liu-dmm-dynamic-anchor-discussion] discuss the dynamic anchor solution for distributed mobility management. The idea is to use the local allocated prefix for any newly initiated 'IP session' and use the previously allocated prefix for the ongoing sessions. This specification can be used to implement the prefix selection for dynamic anchoring. For example, both the locally allocated and the remotely allocated/anchored prefixes can be identified by the prefix property option as described in Section 3.2.

This specification also shares similar motivations for classifying the prefix as described in [I-D.bhandari-dhc-class-based-prefix]. Some service providers may wish to allocate specific prefixes for some services or type of traffic. In this situation, the end host must be able to classify prefixes according to type of service.

This specification provides tools for extending the IPv6 address management and source address selection so that end hosts (and their applications) can select a proper address for their needs. This

specification complements [I-D.bhandari-dhc-class-based-prefix] by providing the SLAAC version of the additional prefix related meta data information delivery compared to the DHCPv6 stateful approach.

3. Option Formats

3.1. Prefix Information Option with Meta Data

This specification defines a new neighbor discovery protocol message option, the Prefix Information Option with Meta Data (PIOMT), to be used in router advertisement messages. The PIOMT is treated exactly the same as [RFC4861] Prefix Information Option (PIO) except with an addition of new meta data suboptions.

The PIOMT can coexist with RFC4861 PIO. The prefixes advertised in both PIOMT and PIO can even be the same. It is up to the receiving end host to select the appropriate prefix(es) for configuring its IPv6 addresses. In a case the PIO and the PIOMT share the same prefix, then all the other parameter (like flags and lifetimes) MUST be the same.

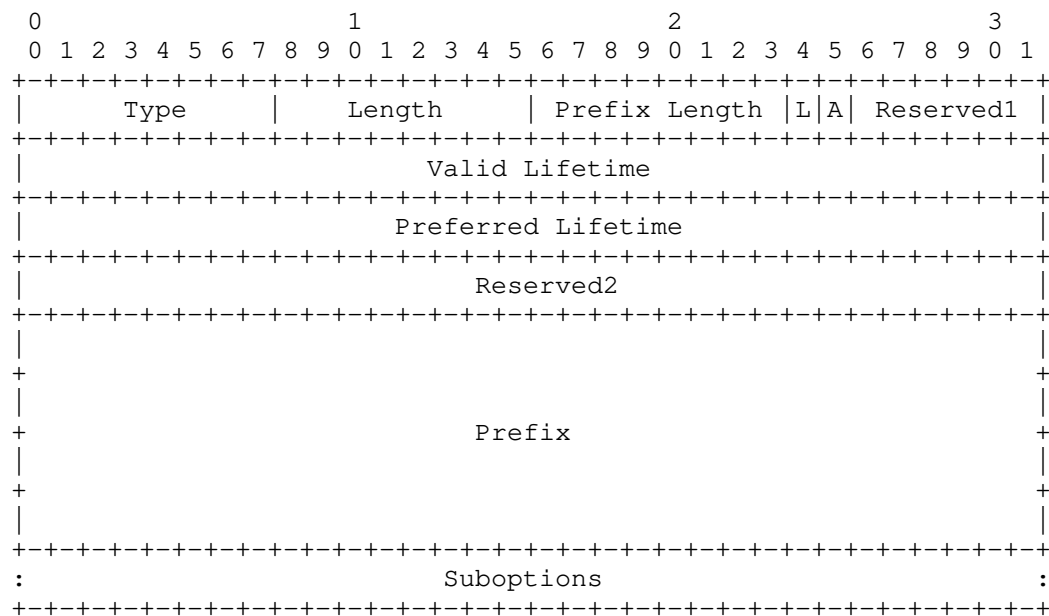


Figure 1: Prefix Information Option with additional meta data

Type

Set to TBD1.

Length

4 if no suboptions are present. Greater than 4 if one or more suboptions are present.

Suboptions

Zero or more suboptions that describe properties and other meta data attached to the advertised prefix. See Section 3.2 for description of the meta data suboption format and suboptions already defined in this specification. The existence of suboptions can be determined from the length field. If the length is greater than 4, then at least one suboption MUST be present.

Rest of the fields are handled exactly as described in Section 4.6.2. of RFC4861 [RFC4861].

3.2. Meta Data Suboptions

The generic suboption format for the PIO with meta data (PIOMT) is shown in Figure 2. The suboption follows the alignment and length rules familiar from [RFC4861]. On a particular note, the flag 'C' describes whether the suboption is mandatory to understand by the receiver or not. If 'C' is set to zero (0), the receiver can silently discard an unknown suboption and skip to the next suboption. If 'C' is set to one (1), then an unknown suboption causes the receiver to silently discard the entire PIOMT and no further suboptions need to be parsed. There can be multiple instances of the same suboption type in one PIOMT option.

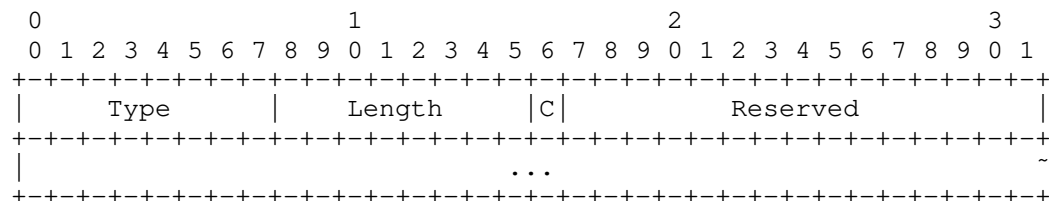


Figure 2: Generic meta data suboption format

Figure 3 shows the Prefix Properties suboption. The prefix properties values are defined in Section 6.1. of [I-D.bhandari-dhc-class-based-prefix]. When an end host receives a

router advertisement message with a PIOMT and the prefix properties suboption, it can use the suboption information as an additional hint for selecting the prefix for a desired purpose and use case. The prefix properties have global meaning i.e., they have the same treatment and handling cross administrative domains. The value for the 'C' flag SHOULD be one (1). This also implies that if the prefix properties bit vector has a flag bit set, which the receiving end host does not understand and the 'C' flag is also set, then the whole PIOMT option MUST be discarded.

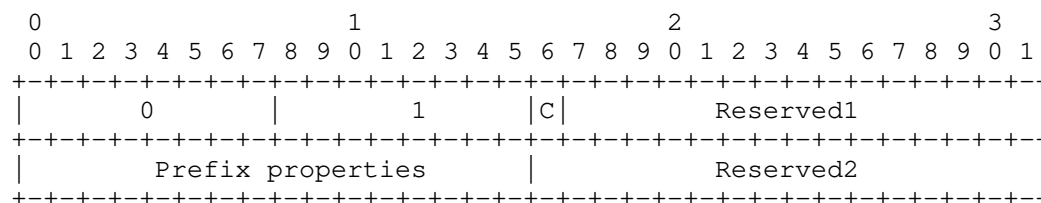


Figure 3: Prefix Properties suboption

Figure 4 shows the Prefix Class suboption. The prefix class values and usage follow what has been defined in Section 2.3. of [I-D.bhandari-dhc-class-based-prefix]. When an end host receives a router advertisement message with a PIOMT and the prefix class suboption, it can use the suboption information as an additional hint for selecting the prefix for a desired purpose and use case. The prefix class has only local administrative meaning i.e., they are local to the access network and may overlap both semantically and registry wise across different administrative domains. How the boundaries of an administrative domain are determined is outside the scope of this specification. The value for the 'C' flag SHOULD be zero (0).

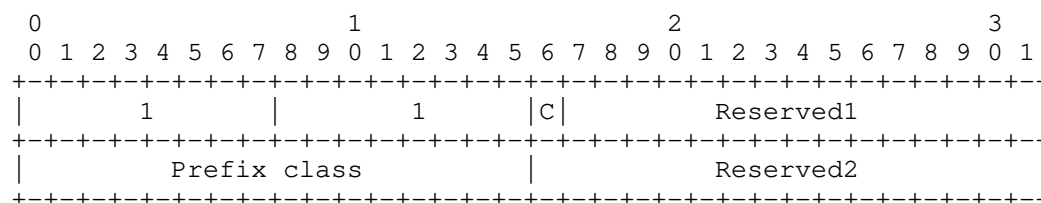


Figure 4: Prefix Class suboption

Future specifications MAY define new suboptions. One potential example could be a suboption to identify the provisioning domain where the configuration information originates.

4. Host Considerations

4.1. Stateless Address Autoconfiguration Enhancements

This specification extends to the [RFC4862] Stateless Address Autoconfiguration (SLAAC). As described in Section 3.1, a new [RFC4861] PIO like option PIOMT can be used to either complement or entirely replace the PIO in a router advertisement. An end host that understands the PIOMT option MUST always prefer a prefix found in the PIOMT over the same prefix found in the PIO option.

4.2. Internal Data Structures

The host internal data structures need to be extended with the 'prefix property' and the 'prefix class' information associated to the learned prefix and configured addresses. How this is accomplished is host implementation specific. It is also a host implementation issue how an application can learn or query both properties or class of an address or a prefix. One possibility is to provide such information through the socket API extensions (see discussion in [I-D.liu-dmm-mobility-api]). Other possibilities include the use of e.g., `ioctl()` or NetLink [RFC3549] extensions.

4.3. Default Address Selection

The 'prefix property' is only used as a hint. It does not affect the existing [RFC6724] automatically. A specific rule to host's policy table has to be inserted by an application or some daemon process. Alternatively, an application can express its address mobility property preferences through the socket API extensions (see discussion in [I-D.liu-dmm-mobility-api]), which means the socket library or middleware has to modify [RFC6724] policy table or algorithm.

The 'prefix properties' flags MAY define the prefix preference for an IP stack that understands the extensions defined in this specification. The IP stack SHOULD use the properties preferences to supersede [RFC6724] Source Address Selection Rule 8 when selecting a default source address among multiple choices and an application has not explicitly indicate what kind of source address it prefers.

The 'prefix class' defines an application 'class' the advertised prefix is intended to be used for. The class has only local administrative domain significance. The 'prefix class' can be used, for example, to identify prefixes that are meant to be used reach a voice over IP (VoIP) service or a video streaming application within the local administrative network. A specific application in the end host MAY use this additional class information when enumerating

through multiple available addresses and then select a specific address to be used for its purposes.

5. Router Considerations

A network administrator MAY configure routers complying to this specification that also emit router advertisements to include the PIOMT option into every router advertisement that would also contain the [RFC4861] PIO option. Since the PIOMT emitting router has no prior knowledge whether the end hosts on the link support the PIOMT option, it is strongly RECOMMENDED that both [RFC4861] PIO and the PIOMT are always included in the router advertisement, even if the advertised prefixes were the same. Whether a router sends router advertisements containing the PIOMT options is a local administrative decision.

A router can also make use of the 'C' flag handling in the PIOMT suboptions when introducing new functionality into the network. Since it is possible to include multiple suboptions of the same type into the PIOMT option, the router can easily make a difference between e.g., prefix properties that must be understood by the receiver and those that can safely be ignored.

6. Security Considerations

Existing Prefix Information Option related security considerations apply as described in [RFC4861] and [RFC4191]. A malicious node on the shared link could include such 'mobility property' flags in a Prefix Information Option causing the host to learn wrong information regarding the prefix and thus make misguided selection of prefixes on the link. Similarly a malicious middleman on the link could modify 'mobility property' flags in a Prefix Information Option causing misguided selection of prefixes. In order to avoid on-link attacks, SeND [RFC3971] can be used to reject Router Advertisements from potentially malicious nodes and guarantee integrity protection of the Router Advertisements.

7. IANA Considerations

Section 3.1 defines a new IPv6 Neighbor Discovery protocol option type TBD1 for the Prefix Information Option with Meta Data. The type value is defined in the existing 'IPv6 Neighbor Discovery Option Formats' IANA registry.

Section 3.2 defines a new IANA registry for the Prefix Information

Option with Meta Data suboptions. The registry allocation policy is Standards Action [RFC5226]. The initial allocations for the prefix properties and prefix class suboptions are listed in Section 3.2.

8. Acknowledgements

The authors thank Ole Troan for his feedback and suggestions on this document (the Classed PIO).

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

9.2. Informative References

- [I-D.bhandari-dhc-class-based-prefix]
Systems, C., Halwasia, G., Gundavelli, S., Deng, H., Thiebaut, L., and J. Korhonen, "DHCPv6 class based prefix", draft-bhandari-dhc-class-based-prefix-04 (work in progress), February 2013.
- [I-D.damic-6man-pmip6-ind]
Damic, D., "Proxy Mobile IPv6 indication and discovery", draft-damic-6man-pmip6-ind-00 (work in progress), March 2009.
- [I-D.draft-liu-dmm-dynamic-anchor-discussion]
Liu, D., "DMM Dynamic Anchor Discussion",

draft-liu-dmm-dynamic-anchor-discussion-00 (work in progress), March 2012.

[I-D.liu-dmm-mobility-api]

Liu, D. and H. Deng, "Mobility API Extension for Distributed Mobility Management", draft-liu-dmm-mobility-api-01 (work in progress), July 2013.

[I-D.seite-dmm-dma]

Seite, P., Bertin, P., and J. Lee, "Distributed Mobility Anchoring", draft-seite-dmm-dma-06 (work in progress), January 2013.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, July 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, September 2007.

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[TS.29274]

3GPP, "3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C)", 3GPP TS 29.060 8.11.0, December 2010.

Authors' Addresses

Jouni Korhonen
Renesas Mobile
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Basavaraj Patil
Cisco

Email: bpatil1@gmail.com

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sgundave@cisco.com

Pierrick Seite
Orange
4, rue du Clos Courtel, BP 91226
Cesson-Sevigne 35512
France

Email: pierrick.seite@orange.com

Dapeng Liu
China Mobile
32 Xuanwumen West Street
Beijing, Xicheng District 100053
China

Email: liudapeng@chinamobile.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

M. Le Pape
S. Bhandari
Cisco Systems
I. Farrer
Deutsche Telekom AG
July 15, 2013

IPv6 Prefix Meta-data and Usage
draft-lepape-6man-prefix-metadata-00

Abstract

This document presents a method for applications to influence the IPv6 source selection algorithm used by the IP stack in a host. To do so, IPv6 prefixes are associated with meta-data when configured by the network. This meta-data allows the network to describe the purpose and properties of the prefix enabling applications to make intelligent decision when selecting a prefix.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 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. Motivation	3
1.1.1. Home networks	3
1.1.2. Mobile networks	4
2. Overview	5
3. Considerations	7
3.1. Prefix meta-data propagation	7
3.2. Configuring Applications	7
3.3. Application to network stack communication	8
3.4. Default Address Selection	8
3.5. Scope of Prefix Color	8
3.5.1. Local scoping	9
3.5.2. Local scoping with fuzzy matching	9
3.5.3. Global scoping	9
3.6. Compatibility with Existing Implementations	9
4. IANA Considerations	10
5. Security Considerations	10
6. Acknowledgements	10
7. Change History (to be removed prior to publication as an RFC)	10
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Appendix A. Prototype notes	12
A.1. Homenet prototype implementation notes	12
A.1.1. Video provider service	12
A.1.2. Prefix Color delegation	12
A.1.3. Configuring Applications	13
A.1.4. Android DHCPv6	14
A.1.5. Application to network stack communication	14
A.1.6. Android kernel	15
A.1.7. Limitations of the current prototype	16

Authors' Addresses	16
--------------------	----

1. Introduction

IPv6 provides not only a larger address space than IPv4, but also allows host interfaces to have more than one IPv6 address of the same or different scope(s). When multiple prefixes are assigned to one or more network interfaces each of the prefixes can have a specific property and purpose associated with it. For example: In a mobile network, a mobile device can be assigned a prefix from its home network and another from the visiting network that it is currently attached to. Another example is a public WLAN hotspot configured with two prefixes offering Internet access. One is free, but low-quality, whilst the other is charged and offers service level guarantees.

A prefix may have well defined properties that are universal and have additional meta-data associated with it in order to communicate the prefixes local significance. When multiple prefixes are provisioned to the host, this additional information allows the host and applications to make more intelligent decisions about the best IPv6 address to select when sourcing connections.

This document introduces the motivations and considerations for having additional meta-data associated with a prefix and also proposes a format for the meta-data itself.

The underlying assumption is that a endpoint or an application has multiple prefixes to choose from. Typically this means either the endpoint has multiple interfaces or an interface has been configured with multiple prefixes. This specification does not make a distinction between these alternatives.

1.1. Motivation

In this section, the motivation for attaching meta-data to IPv6 prefixes is described in the context of both mobile and home networks. The meta-data helps to distinguish an IPv6 prefix and aids with the selection of the prefix by different applications.

1.1.1. Home networks

In a fixed network environment, the homenet CPE may also function as both a DHCPv6 client (requesting IA_PD(s)) and a DHCPv6 server allocating prefixes from delegated prefix(es) to downstream home network hosts. Some service providers may wish to delegate multiple globally unique prefixes to the CPE for use by different services classes and traffic types.

Motivations for this include:

- o Using source prefix to identify the service class / traffic type that is being transported. The source prefix may then reliably be used as a parameter for differentiated services or other purposes. E.g. [I-D.jiang-v6ops-semantic-prefix]
- o Using the specific source prefix as a host identifier for other services.
- o In multi-homed environments, a single homenet LAN may have multiple globally unique prefixes provided by the different service providers. In this scenario, correct source address selection is fundamental to successfully establishing connections. E.g. [I-D.troan-homenet-sadr]

Any host which is configured with multiple prefixes must perform a source address selection process when initiating a connection. Any client that has multiple globally unique prefixes only has source and destination longest-prefix matching policy [RFC6724] in order to make this selection. For cases such as those listed above, longest-prefix matching can not assist the client in selecting the correct source address to use. Addition information is needed to assist the client in making the correct source address selection.

1.1.2. Mobile networks

In mobile network architecture, a mobile node can be associated with multiple IPv6 prefixes belonging to different domains. E.g. home address prefix, care of address prefix (as specified in [RFC3775]). The delegated prefixes may be topologically local and some may be remote prefixes anchored on a global anchor, but available to the local anchor by means of tunnel setup in the network between the local and global anchor. Some prefixes may be local with low latency characteristics suitable for voice call break-out, some may have global mobility support.

So, the prefixes have different properties and it is necessary for the application using the prefix to learn about this property in order to use it intelligently. An example is determining if the prefix is a home address or care of address or other network characteristics that can be offered.

2. Overview

The mechanism that is described in this document describes two different types of meta data which can be used in different ways:

Prefix Properties Provides a method for an application to "hint" required source address properties to the kernel. These properties are universal and expressed as a set of flags.

Prefix Color Provides an arbitrary color value to prefixes (of local significance) enabling an application to request a source prefix with a specific color.

These two meta data types are described in more detail below.

Prefix Properties functions as follows:

- o The client receives multiple prefixes, with relevant Prefix Property meta-data attached to each prefix
- o Prefix property aware applications running on the client have a policy defining that they prefer prefixes that have specific properties.
- o On initiating a connection, the Prefix Property aware application passes the required prefix properties to the kernel along with the connect request
- o The kernel checks the requested properties against the available prefixes. If a match is found, the matching prefix is passed back to the application
- o The application uses the returned prefix when making the call to the socket API to create the connection
- o If no prefix matching the requested properties is available, then the kernel uses [RFC6724] for source address selection as normal

Prefix property offers well defined universally understood information about the prefix. Example properties include whether a prefix can provide Internet reachability, if the prefix offers application specific Internet service level, if the prefix usage is free/charged, if the prefix offers security guarantees etc. This is maintained as a global registry.

Prefix Coloring functions as follows:

- o The client receives multiple prefixes, with relevant meta-data attached
- o Color aware applications running on the client are provisioned with policy telling them which prefix color to request
- o On initiating a connection, the meta data aware application passes the required prefix color to the kernel along with the connect request
- o The kernel takes this color and selects the prefix matching the requested color and passes this back to the application
- o The application uses the returned prefix when making the call to the socket API to create the connection

Prefix colour conveys information of the prefix that is of relevance to the network where the prefix is provisioned and application using it. Example usage of prefix color include color that is provisioned to offer better video application experience. The prefix color is defined as a 16 bit numerical value.

Figure 1 illustrates a typical network with different components that can add, understand and use the meta-data attached to a prefix.

- o Mobile or ISP Network - Provisioned with prefixes that offer specific network characteristic. e.g. prefixes that do not have internet reach but can offer quality of service required for better video application experience. Includes address delegation server that associate prefixes with this information, selects and offers this information during prefix delegation
- o Home/Mobile gateway - Learns or determines characteristic of the prefix and propagates it along with prefix delegation. e.g. Determines if the prefix is locally anchored or learns the prefix meta-data from the ISP prefix delegation server and includes this information in prefix delegation to endpoints
- o Endpoint network stack - Learns the additional information associated with the prefix and offers interface to applications for listing and selecting the available prefixes
- o Prefix selection policy - Either embedded in the application/endpoint or learnt from a server that helps choose the prefix with specific characteristic for the application based on predetermined service agreement between the application/endpoint/application service provider and network service provider

- o Applications - That can utilize the prefix with specific characteristic for enhanced application user experience e.g. On demand video application, by choosing the prefix with appropriate prefix selection policy while connecting and delivering the application over the network

This prefix meta-data could be further extended to have more attributes such as the administrative domain of the prefix.

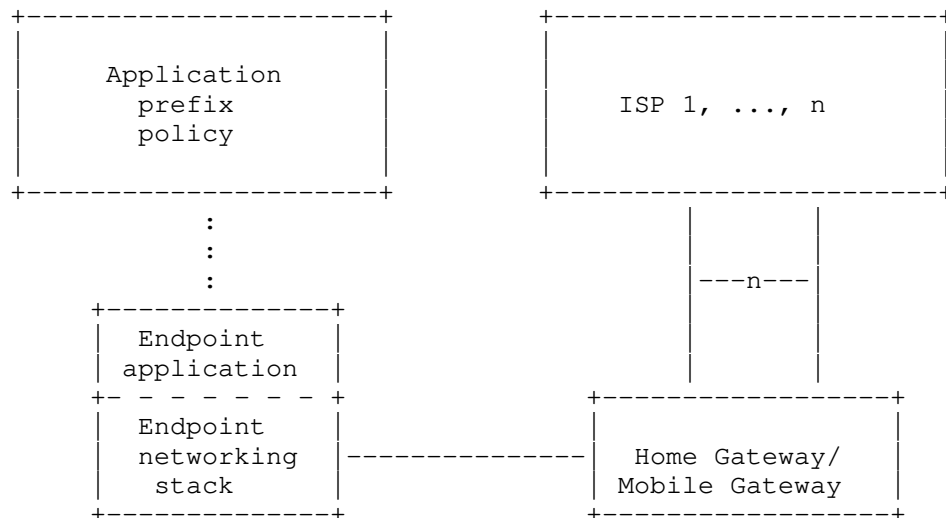


Figure 1

3. Considerations

3.1. Prefix meta-data propagation

The prefix meta-data can be delivered using DHCPv6 prefix delegation and address allocation as elaborated in [I-D.bhandari-dhc-class-based-prefix] or via IPv6 Neighbour discovery (ND) as defined in [I-D.korhonen-6man-prefix-properties].

3.2. Configuring Applications

Applications supporting multiple prefixes obtain the prefixes from the host kernel along with their meta-data.

The policy can then be contained either locally (e.g. If the application is intended only for use within a specific network, linked to a particular ISP comes prepackages with prefix color to

use), or be contained on a remote policy server. The mechanism used to exchange the meta-data information and selection between application/host with a remote server is beyond the scope of this document.

3.3. Application to network stack communication

Once an application has determined the appropriate property and color for its use it has to communicate with the network stack to select the prefix. The host internal data structures need to be extended with the 'prefix property' and the 'prefix color' information associated to the learnt prefix and configured addresses. How this is accomplished is host implementation specific. It is also a host implementation issue how an application can learn or query both properties and color of an address or a prefix. One possibility is to provide such information through the socket API extensions. Other possibilities include the use of e.g., `ioctl()` or NetLink [RFC3549] extensions or by using the IPv6 address scope [RFC4007].

Discussion point: Should prefix property and color be mutually exclusive? This would avoid complexities which takes precedence when one prefix matches color and another matches property. Possibly a prefix may be advertised with both, but the application can only request property or color.

3.4. Default Address Selection

[RFC6724] provides a mechanism for selecting which source address to use, in the absence of an application or upper layer protocol's explicit choice of a legal destination or source address.

The use of prefix meta-data allows an application to express property preferences through socket API extensions, meaning that when used for creating a socket, [RFC6724] source address selection is not required.

If a higher layer protocol or application does not include a prefix property preference when making a create socket request, then source address selection according to [RFC6724] is followed as normal.

3.5. Scope of Prefix Color

Since a home can be connected to multiple ISPs, it is possible that it receives multiple prefixes with the same color from different ISPs. Since the application coloring policy is not received with the color, multiple ISPs may use different coloring policy for a single color. For example: One ISP could use color 50 for video whilst a second ISP is using color 50 for audio.

This section presents some alternatives to handle this problem.

3.5.1. Local scoping

A locally scoped color is a value which is selected by the network and application providers with no central registry. In a multi homed network, this may result in two providers selecting the same color for different behaviors. A color translation could be performed to ensure unique color at the device that connects to multiple providers.

3.5.2. Local scoping with fuzzy matching

To avoid having to maintain multiple colors for each prefix for translating the color, a specific algorithm can be used to determine the new color from the old one on conflict.

For example, when a collision is detected, the new color value may be incremented. Further, colors could be defined to be equally spaced (e.g., 10s or 100s).

Many other encodings are possible as well, as long as obtaining the original color communicated by the ISP may be recovered in the event the application policy server requires this.

3.5.3. Global scoping

A globally scoped color avoids the need for responding to collisions. This can be achieved by disambiguating the color by attaching the domain that provisions the color to the prefix meta-data or by assigning colors from a global registry that comes with the overhead of maintaining such a registry.

3.6. Compatibility with Existing Implementations

The prefix meta-data mechanism that is described in this document provides a way of improving source address selection over the longest-prefix matching method used by [RFC6724].

However, all IPv6 capable hosts deployed at the time of writing do not have the capability of understanding and processing prefix meta-data. This means that any new mechanism must be backwards compatible with existing implementations. Also, clients which understand prefix meta-data need to support applications which do not have meta-data awareness.

There are a number of possible approaches that could be taken here. The following list is included as ideas for further development:

- o In DHCPv6 only clients which request prefixes with meta-data (e.g. signalled through OPTION_ORO in the IA_NA or IA_PD request) will receive them.
- o In case of prefix delegated using IPv6 Neighbour discovery (ND) both forms of prefix i.e with and without meta-data can be offered.
- o If an application makes a socket API request and does not include meta-data as part of the request, follow [RFC6724] source address selection, but remove any prefixes that have meta-data from the list of candidate addresses. It follows that there should be a GU prefix advertised that does not have any meta-data associated that would act as the default choice for non prefix meta-data aware clients and applications.

4. IANA Considerations

Should the global scoping for prefix color be chosen, a new registry should be created by IANA to store colors.

5. Security Considerations

TBD

6. Acknowledgements

The authors would like to acknowledge review and guidance received from

7. Change History (to be removed prior to publication as an RFC)

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.bhandari-dhc-class-based-prefix]
Systems, C., Halwasia, G., Gundavelli, S., Deng, H., Thiebaut, L., and J. Korhonen, "DHCPv6 class based prefix", draft-bhandari-dhc-class-based-prefix-04 (work in progress), February 2013.

[I-D.ietf-dhc-dhcpv4-over-ipv6]

Cui, Y., Wu, P., Wu, J., and T. Lemon, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-06 (work in progress), March 2013.

[I-D.jiang-v6ops-semantic-prefix]

Jiang, S., Sun, Q., Farrer, I., and Y. Bo, "A Framework for Semantic IPv6 Prefix", draft-jiang-v6ops-semantic-prefix-03 (work in progress), May 2013.

[I-D.korhonen-6man-prefix-properties]

Korhonen, J., Patil, B., Gundavelli, S., Seite, P., and D. Liu, "IPv6 Prefix Properties", draft-korhonen-6man-prefix-properties-02 (work in progress), July 2013.

[I-D.troan-homenet-sadr]

Troan, O. and L. Colitti, "IPv6 Multihoming with Source Address Dependent Routing (SADR)", draft-troan-homenet-sadr-00 (work in progress), February 2013.

[RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, July 2003.

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

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

Appendix A. Prototype notes

A.1. Homenet prototype implementation notes

This section provides the implementation details of a prototype video application on Android for a Galaxy Nexus device developed for the home network.

A.1.1. Video provider service

A possible use of this prefix coloring is a video service, which requires the network to guarantee a minimal throughput for streaming video. A prefix could be colored by the ISP to indicate that traffic sourced from that prefix will have a certain service level. Using prefix coloring avoids having to set up a separate network for this usage, or implement QoS traffic identification, classification and marking.

An agreement could then be established between the video service provider and the ISP, telling the video provider to use the specific color when streaming video. In the following example, the color 50 was used.

A.1.2. Prefix Color delegation

The CPE routers request prefixes using prefix delegation [RFC3633] with the `OPTION_PREFIX_CLASS` option [I-D.bhandari-dhc-class-based-prefix]. This informs the upstream provider that the CPE supports colored prefixes. If an ISP does not support this option, it will be ignored, and the CPE will only get colorless prefixes. Otherwise, the ISP returns multiple prefixes each with their associated color. A color of '0' is identical to an uncolored prefix, for application compatibility, as explained in Appendix A.1.5. If the CPE does not support colored prefixes, the ISP could decide to delegate a normally colored prefix as an colorless one, though this means hosts will use this prefix according to the default source address selection algorithm, and will not associate any meaning to it.

Once the CPE receives those prefixes, it distributes them, along with their color, using OSPF and the homenet protocols.

[I-D.troan-homenet-sadr] defines "Source Address Dependent Routing" (SADR) which ensures that packets are routed based on their destination as well as source address. SADR is necessary to ensure that a multihomed network using provider aggregatable addresses will send the packet out the right path to avoid violating the provider's ingress filtering. To ensure that those prefixes keep their meaning, Source Address Dependent Routing [I-D.troan-homenet-sadr] is implemented and used.

Colored addresses are advertised to hosts through DHCPv6, to associate the color to the address. Colorless addresses may be distributed through DHCPv6 or through Router Advertisements. Hosts supporting colored prefixes include the `OPTION_PREFIX_CLASS`, and receive colored addresses. For legacy hosts, who do not include this option, there are two possibilities :

- o Those hosts can receive all available prefixes, including colored ones, as uncolored. This allows a legacy host in a fully colored homenet to still have access to IPv6. However, those hosts may use prefixes for the wrong purposes.
- o Those hosts can receive only colorless prefixes. This ensures that a prefix will not be used for the wrong purpose. However, hosts in a fully colored environment will not get access to IPv6. This can however be what the ISP originally intended, for example if the ISP does not provide access to the IPv6 Internet, but uses IPv6 for wall gardened services, which their specific devices know how to use.

A.1.3. Configuring Applications

Applications supporting multiple prefixes obtain the prefixes from the host kernel, along with their color.

The policy can be contained either in a local database (e.g. If the application is intended only for use within a specific network, linked to a particular ISP), or be contained on a distant server.

For applications that do not contain a local database, an HTTP POST request is sent to a predefined server using a colorless prefix. This server, through means that are out of the scope of this document, selects the most appropriate color for the URIs used by the application. It then returns an XML document containing the mapping between the URIs and the colors. URIs in this document MAY use wild cards.

When the application is started, it sends the available prefixes and their color to the video provider server which answers with a wild card URI videos.example.com associated to color 50. In this example application it receives:

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings>
  <mapping>
    <URI>*://audio.example.com/*</URI>
    <color>40</color>
  </mapping>
  <mapping>
    <URI>rtsp://video.example.com/*</URI>
    <color>50</color>
  </mapping>
</mappings>
```

The server is expected to know the application, and thus to list all URIs that could be of use to the application. The application will not ask the server if it has to contact an address not in the list and will use the colorless prefix. This avoids an additional delay when trying to contact an unlisted URI.

Example: While the application is browsing the video list, it is using www.example.com, and thus the colorless prefix. However as soon as a video is chosen, it starts streaming from videos.example.com, and asks to connect to host videos.example.com with color 50, indicating that it wishes to use the colored prefix.

A.1.4. Android DHCPv6

Considering that this prototype is being implemented on Android, the first step is to get a running DHCPv6 client on Android, with support for the OPTION_PREFIX_CLASS option.

The odhcp6c client, which already supports OPTION_PREFIX_CLASS, has been ported to Android, and is set to run in parallel to the dhcpcd client used for DHCP. The success of any of the two clients results in the success of the WiFi connection, so as to support IPv6 only networks.

This client configures the IPv6 addresses using calls to IP address, which is modified to support the addition of a class option to set the prefix color.

A.1.5. Application to network stack communication

Once an application has received the appropriate color for its use, in this prototype it specifies the prefix it wishes to use by using the IPv6 address scope [RFC4007]. When resolving this address, the standard library then adds this information in the address information it returns, using the scope field, allowing the kernel to appropriately select the source IP when connecting. For this reason, a color of 0 is identical to an colorless prefix.

In the example, when downloading from `video.example.com`, the application would request a connection to `video.example.com%50`.

This allows the user to override the application's default simply by specifying a color in the scope of the URI it is trying to access, and requires little to no change in applications to support it. Applications that allow scope ids do not need to be modified in order to allow the user to use multiple prefixes (though it is then up to the user to select its color). A web browser that allows scope id would allow the user to add a color to the URI, without requiring any modifications.

A.1.6. Android kernel

To reduce the amount of modifications needed by the applications to support this prefix coloring, we need to avoid having to bind to the address in the colored prefix before initiating the connection. The kernel is expected to choose the correct source address when a colored destination is used.

This implies storing the color in the kernel, along with the address, which is done using a new attribute `IFA_color` to the netlink message `RTM_NEWADDR`, used by ip address. Setting a colored prefix using `ioctl`s is not supported.

Since colors are put in the scope id part of the destination address, we continue to use the scope element of the `sockaddr_in6` structure to store the color when sending connect messages to the kernel. The scope is only used when considering local addresses, so we interpret the presence of a scope on a non link-local address to be a color. Colors can not be assigned to link-local addresses, but since they are on the same link, source address shouldn't impact how the network treats packets. When selecting the source address, we then discard all addresses which do not have the correct color.

A.1.7. Limitations of the current prototype

It does not implement any duplicate color detection. Colors are considered to be unique within the home, and to correspond to the original color provided by the ISP. This is compatible with Global scoping. No changes would be required to the host in order to support Local scoping with fuzzy matching, but OSPF would need to detect collisions, and the server would need to recalculate the original color before making a decision. In this implementation, hosts that do not support colors do not receive colored prefixes.

Authors' Addresses

Maico Le Pape
Cisco Systems
Paris
FR

Email: maico@maicolepape.org

Shwetha Bhandari
Cisco Systems
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Ian Farrer
Deutsche Telekom AG
GTN-FM4, Landgrabenweg 151
Bonn 53227
Germany

Email: ian.farrer@telekom.de

Network Working Group
INTERNET-DRAFT

H. Rafiee
Huawei Technologies Duesseldorf GmbH
C. Meinel

Intended status: Informational
Expires: April 1, 2015

Hasso Plattner Institute
October 1, 2014

Router Advertisement based privacy extension in IPv6 autoconfiguration
<draft-rafiee-6man-ra-privacy-09.txt>

Abstract

The purpose of this document is to address the privacy issues that exist within the Privacy Extension specification and offer an alternative privacy aware solution to correct these shortcomings and also offer an alternative random number generator.

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 Expires: October 1, 2014.

Copyright Notice

Copyright (c) 2013 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. Algorithm Overview	3
2.1. Duplicate Address Detection (DAD) Process	4
3. Advantages of ra-privacy over CGA and DHCPv6	5
4. Interface ID (IID) generation based on the MAC address	5
5. Lifetime of an Interface ID (IID)	5
6. Security Considerations	6
7. IANA Considerations	6
8. Acknowledgements	6
9. References	6
9.1. Normative	6
9.2. Informative	7
Authors' Addresses	8

The purpose of this document is to address the privacy issues that exist within the Privacy Extension specification and offer an alternative privacy aware solution for a random number generator to correct these shortcomings.

1. Introduction

Today, privacy is an important concern to everyone in their everyday life. In Internet, Privacy is not bound to any particular layer in the Open Systems Interconnection (OSI) model, but some layers do have a greater impact on user's privacy. This document tries to address the deficiency that exists in the privacy extension specification [RFC4941] as it relates to the network layer. In other words, how an IP address can affect user's privacy. One of the solutions for IPv6 autoconfiguration (RFC 4682) is the use of the Privacy Extension [RFC4941]. The Privacy Extension document explains two different approaches that can be used for IID generation. In the first approach, the use of stable storage enables a node to find which IIDs are currently in use and which are in reserve. In the second approach, where stable storage is not available, it suggests the use of some randomizing approaches and also comments about other available randomizing mechanisms such as Cryptographically Generated Addresses (CGA) [RFC3972] or Dynamic Host Configuration Protocol (DHCPv6). The Privacy Extension document also refers to the use of a named approach as an approach to be used in a mechanism for generating greater randomization. This document aims to introduce a new alternative randomized algorithm to RFC 4941 that can be used in any systems to generate IID not based on the hardware Identifier. This document addresses the following problems:

- If the node is changing the physical link, it may keep the IID already used on the link it is leaving, to form an IPv6 address on the new link using SLAAC. Therefore, it negatively impacts user's privacy and enable tracking a node over the networks.
- This document also clarifies the use of other algorithm and compare it with CGA and DHCPv6 when there is no stable storage available. So that the node may be able to make use of a greatly randomized IID because, according to section 3.2.2 of RFC 4941, there is nothing to force the use of RFC 4086.

2. Algorithm Overview

This section explains how to make use of a new algorithm that will provide for a higher randomization of the IID without the need for any extra memory or stable storage. This algorithm can be used when there is no stable storage available in the node or the node does not want to store any value in memory. This approach is easy to implement while at the same time provide a good randomization level.

1. Generate a 16 byte random number called modifier. To generate this modifier, implementations should use a random seed to aid in the randomization of this number.

2. Obtain the router prefix from the Router Advertisement message

3. Obtain the nodes' current time in microseconds and call it timestamp. This field consists of a 64-bit, unsigned integer containing the number of milliseconds since January 1, 1970, 00:00 UTC, by using a fixed point format. (Please refer to section 11 for more detail)

4. Concatenate the modifier with the timestamp and the router prefix.

$R1 = (\text{modifier}(16 \text{ bytes}) || \text{timestamp}(8 \text{ bytes}) || \text{router prefix})$

Note: Implementations MIGHT skip step 3 and use this timestamp as a seed to PRND function to generate modifier. So, R1 is as follow

$R1 = (\text{modifier}(16 \text{ bytes}) || \text{router prefix})$

5. Execute SHA2 (256) against the result from step 4.

$\text{digest} = \text{SHA256}(R1)$

The use of SHA2 (256) is recommended because the chances of finding a collision are less than when using SHA1 and the generation time is acceptable (in microseconds using a standard CPU). In the future, if a faster and collision free algorithm becomes available, then Implementations need to support it.

6. Take the x bits (leftmost bits or rightmost bits, does not matter and either way is correct) from the resulting output of step 5 (SHA2 digest). x depends on the number of bits that is needed for an IID. Bits u and g do not have any special meaning as in [ugbits].

Note: in case this algorithm is used for only the generation of random number, the next steps should be skipped.

7. Concatenate the IID with the local subnet prefix in order to set the local IP address. If the lifetime of the old local address has not expired, then the node MIGHT skip this step. Otherwise it will receive a new router prefix.

8. Concatenate the IID with the router subnet prefix (Global subnet prefix), obtained from the RA message, and set it as a tentative privacy IP address. This IP address will become permanent after Duplicate Address Detection (DAD) processing.

2.1. Duplicate Address Detection (DAD) Process

After the DAD process has completed, if the node finds collisions in the network, then the modifier will be incremented and the DAD process will be repeated. If, after 3 tries, it receives the same result, it will consider this an attack and will start using that IP address.

3. Advantages of ra-privacy over CGA and DHCPv6

This algorithm has some advantages over the use of CGA and DHCPv6.

- CGA generates a highly randomized IID but CGA algorithm is compute intensive. This is the primary reason that CGA algorithm has been only implemented as an experiment but rarely enabled or used in practice.
- DHCPv6 server might generates the IIDs sequentially and assign these IIDs to the nodes. This sequential assignment is according to a range of IP addresses and might allow an attacker to scan the nodes in this network and harm their privacy. This is also true, if the node releases an IP address and set new IP address. It is again from the same range.
- An administrator needs to configure and maintain DHCPv6 server. This is not needed if a node uses Neighbor Discovery Protocol (NDP).

4. Interface ID (IID) generation based on the MAC address

When a node uses the mechanism explained in this document to generate an IID, it must not use any other IID generation approaches that are based on MAC addresses (RFC 4862) for either temporary or non temporary IID generation. The node might use the algorithm explained in [RFC7217] for the generation of a public address that does not make use of EUI-64 or the MAC address for public (global) addresses.

The choice of whether or not to list a node's address in the DNS, undisguised, depends on many factors, including the set of applications to be run on the host. Not listing a node's address in the public DNS may afford the node greater privacy, but, at the same time, it may also impair its ability to support certain applications.

5. Lifetime of an Interface ID (IID)

The node might make use of the same algorithm and the same lifetime as is explained in RFC 4941, or the node might choose a lifetime based on some other algorithms or policies [LifeTime]. If it uses the lifetime explained in RFC 4941, then it should generate a new IID whenever it is in different network, regardless of the option in the Router Advertisement message to extend this lifetime. This is because, based on Privacy Extension specification, the node does not

change its IID if the option in the router advertisement in new network extend the lifetime of IID.

6. Security Considerations

As is explained in the Privacy Extension document. the same approaches are used to maintain security, such as using Secure Neighbor Discovery (SeND) (RFC 3971) or using a monitoring system which would inform the administrator of the status of the network and of any suspended activities in the network.

7. IANA Considerations

There is no IANA consideration

C++ source code for the comparison of ra-privacy and privacy extension can be found in the following address

<http://sourceforge.net/u/hrafiee/raprivacy/ci/master/tree/>

8. Acknowledgements

The author would like to thank all those people who directly helped in improving this draft especially Andrew Yourtchenko

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4291] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture," RFC 4291, February 2006.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)," RFC 3972, March 2005.
- [RFC4941] Narten, T., Draves, R., Krishnan, S., "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker,

"Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC7217] Gont, F., " A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC7217, April 2014

9.2. Informative References

[ugbits] Carpenter B., "Significance of IPv6 Interface Identifiers", RFC 7136, February 2014

[LifeTime] Rafiee, H., Meinel, C., Nordmark, E.,
"Interface ID lifetime Algorithms",
<https://tools.ietf.org/html/draft-rafi-ee-v6ops-iid-lifetime>,
2013

Authors' Addresses

Hosnieh Rafiee
HUAWEI TECHNOLOGIES Duesseldorf GmbH
Riesstrasse 25, 80992,
Munich, Germany
Phone: +49 (0)162 204 74 58
Email: hosnieh.rafiiee@huawei.com

Christoph Meinel
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
Email: meinel@hpi.uni-potsdam.de

Network Group
INTERNET-DRAFT
Intended status: Experimental

H. Rafiee
Huawei Technologies Duesseldorf GmbH
C. Meinel

Hasso Plattner Institute

Expires: March 19, 2015

September 19, 2014

A Simple Secure Addressing Scheme for IPv6 AutoConfiguration (SSAS)
<draft-rafiee-6man-ssas-11.txt>

Abstract

Since performance and security are, both, two important criteria for a mechanism to be widely used by different nodes with various resources, the purpose of this document is to propose a mechanism for local security and to prevent IP spoofing. This mechanism also consider user's privacy.

Status of this Memo

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

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

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

This Internet-Draft will expire on March 19, 2015.

Copyright Notice

Copyright (c) 2013 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. Conventions used in this document	4
3. Algorithms Overview	4
3.1. Interface ID (IID) Generation	4
3.1.1. Signature Generation	5
3.1.2. Generation of NDP/SeND Messages	6
3.1.2.1. SSAS signature data field	6
3.1.3. SSAS verification process	8
3.2. Resource Public key Infrastructure (RPKI)	9
4. SSAS Applications	9
4.1. A solution for all nodes	9
4.2. Authentication in Network layer	9
4.3. Authentication in Application Layer	10
4.4. Other Applications	10
5. Security Considerations	10
6. IANA Considerations	11
7. Privacy Consideration	11
8. Appendix A	11
8.1. Comparison of CGA and SSAS generation time	11
9. Appendix B	12
9.1. Network-based protection vs. Node-based protection	12
10. Acknowledgements	13
11. References	13
11.1. Normative	13
11.2. Informative	14
Authors' Addresses	16

1. Introduction

In IPv6 networks, nodes can use two different mechanisms to configure their IP addresses -- Neighbor Discovery Protocol (NDP) [RFC4861, RFC4862] and Dynamic Host Configuration Protocol (DHCPv6) [RFC3315]. Unfortunately none of these mechanisms are natively secure. So, they open the nodes with so many local security problems. There are several attacks possible in local network [RFC3756]. One example is IP spoofing that enable an attacker to forge the identity of a victim node, the other example is preventing the node from configuring its IP address.

The reasons that local security is important are as follows [localSecurity]:

- Not all the nodes on the local link are trusted: viruses or other malware can infect a legitimate node in the local link and turn it to an attacker.

- Attacker might be inside the network: The networks of big enterprises might be harmed by one of the staff that was recently fired.

There is currently a mechanism available to secure the NDP, i.e., Secure Neighbor Discovery (SeND) [RFC3971]. SeND does this protection by adding 4 options to NDP packets. Among these options, Cryptographically Generated Addresses (CGA) [RFC3972] is a very important option that provides the node with the proof of IP address ownership by finding a binding between the node's public key and its IP address. Unfortunately CGA has some problems that are listed as follows:

- CGA sec value problem: This problem is explained in [cgaattack] and addressed in [cgabis].

- CGA increases complexity and decreases performance: CGA uses sec value (the value between 0 to 7) and claims to complicate the brute force attacks. (However it is not true based on [cgaattack]) If CGA sec value higher than 0 is in use, then this will reduce the performance because CGA algorithm needs to repeat some steps and it needs the high attention of the CPU and makes the CPU busy. So, CGA sec value higher than 0, consumes more energy than other nodes that do not use CGA. Today, the demands on multi-functioning smaller devices are increasing but unfortunately the battery technology is not as advanced as expected. So, the use of CGA algorithm that needs to use higher level of energy is not ideal for these types of nodes and the use of CGA sec value zero does not protect the node as expected. (Please refer to appendix A for more information)

- CGA might cause privacy issue: Since the generation of CGA higher sec values might take time. The nodes might not be willing to change its IP address and keep this address as long as the subnet prefix is

valid. If the node is a fixed node in the network, then it will be vulnerable to node tracking. The node might also not change the CGA address when it visits a new network or it might not generate any new key pairs. In other word, it might use the same CGA parameters (excluding prefix) as used in the old network and thus it will be vulnerable to node tracking.

- Packet size

CGA uses RSA as a default key pair generation algorithm. This is why, if SeND with CGA option is in use to secure NDP messages, the minimum packet size needs to carry this public key for CGA nodes is 460 bytes. Packet size also reduces the performance and causes delays in the network.

Since privacy and security are, both, very important issues in everyday life, the purpose of this document is to offer an alternative and simple addressing mechanism to generate an interface ID (IID) which provides the node with both security and privacy while does not sacrifice the performance, and tries to decrease the packet size as much as possible.

2. 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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document the use of || indicates the concatenation of the values on either side of the sign.

3. Algorithms Overview

As explained earlier, one of the problems with using the current IID generation approach is the intensive computer processing that is needed for the IID algorithm generation. Another concern is for the lack of security (if CGA is not in use). This is what this document intends to address.

3.1. Interface ID (IID) Generation

To generate the IID a node will need to execute the following steps.

1. Generate key pairs (public/private keys) using one of the latest version of ECC algorithm [RFC6090] or other fastest short key size algorithms. . The implementations SHOULD be updated with any new

version of ECC algorithm when ECC current version is no longer secure. ECC is the default algorithm, but any algorithm capable of generating a small key size in a short amount of time is viable. The node then uses this new value for the generation of the IP address and signature. Comparing the use of ECC to that of RSA shows that an ECC with a 192 bit key is equivalent to a RSA with a 7680 bit key (according to US National Security Agency) In this case the packet size would be decreased by a factor 11 times smaller than that when using RSA.

Note 1: The node MUST not generate the weak key. For ECC, the node MUST not use ECC key size lower than 192 bits. If any nodes used a weak key size, then the other nodes MUST discard receiving the message from that node. If in future, key size 192 bits is considered as a weak key size, the default key size value MUST be changed to the next strong key size.

2. Execute a hash function on the public key. The default hash function is SHA256. If in future, this hash function is no longer secure, the node MUST use the next strong hash function.

3. Take the first 64bits of the digest and call it IID. In case collision count is higher than 1, then depends on the number, takes second 64 bits or third 64 bits of this hash value.

It is not RECOMMENDED to use this algorithm in case IID is less than 64 bits [variableprefix]. A node MUST obtain the prefix length information form router advertisement messages.

4. Concatenate the IID with the local subnet prefix to set the link local IP address.

5. Concatenate the IID with the router subnet prefix (Global subnet prefix), obtained from the Router Advertisement (RA) message, and set it as a tentative public IP address. This IP address will become permanent after Duplicate Address Detection (DAD) processing. (For more information about DAD refer to section 3.1.2.)

Note 1: In this document bits u and g does not have any particular meaning and is used as a part of public key. This assumption is by the clarification of using these bits in [RFC7136].

3.1.1. Signature Generation

SSAS is not dependent to SeND but it can be used as a new option of SeND. When SSAS is used as an option of SeND, SSAS signature can be placed as a RSA signature in SeND. If SSAS is used alone, this section MUST be included in SSAS data structure. This proves that SSAS is compatible to use with SeND.

The SSAS signature is added to NDP messages in order to protect them from IP spoofing and spoofing types of attack. SSAS will provide

proof of IP address ownership. To generate the SSAS signature, the node needs to execute the following steps:

1. Concatenate the timestamp with the MAC address, collision count, algorithm type and the global (public) IP address. (see figure 1)

timestamp 8 bytes	Mac address 6 bytes	Collision Count 3 bits	Algorithm type 1 byte
Global IP address 16 bytes		Other Options variable	

Figure 1 SSAS Signature format

2. Sign the resulting value from step 1, using the ECC private key (or any other short key size algorithm), and call the resulting output the SSAS signature.

If NDP messages contain other data that must be protected, such as important routing information, then this data SHOULD also be included in the signature. The signature is designed for the inclusion of any data needing protection. If there is no data that needs protection, then the signature will only contain the timestamp, MAC address, Collision count and Global IP address (Router subnet prefix plus IID).

3.1.2. Generation of NDP/SeND Messages

After a node generates its IP address, it should then process Duplicate Address Detection in order to avoid address collisions in the network. In order to do this the node needs to generate a Neighbor Solicitation (NS) message. The SSAS signature is added to the ICMPv6 options of NS messages. The SSAS signature data field is an extended version of the standard format of the RSA signature option of SeND [RFC3971]. The timestamp option is the same as that used with SeND. In the SSAS signature, the data field contains the following items: type, length, reserved, Other Len, algorithm type, collision count, subnet prefix, other option and padding.

3.1.2.1. SSAS signature data field

Type 1 byte	Length 1 byte	Reserved 2 bytes	Other len 1 byte
Algorithm type 1 byte	Collision count 3 bits	Subnet prefix 8bytes	Other Options
Hash Function	Response No.	SSAS Signature	

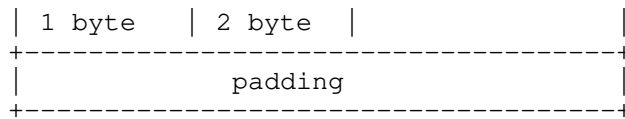


Figure 2 NDP Message Format with SSAS Signature Data Field

- Type: This option is set to 15. This is the sequential number used in SeND to indicate a SSAS data field.

- Length: The length of the Signature Data field, including the Type, Length, Reserved, Algorithm type, Signature and padding, must be a multiple of eight.

- Reserved: A 2 byte field reserved for future use. The value must be initialized to zero by the sender and should be ignored by the receiver.

- Other Len: The length of other options in multiples of eight. The length of this field is 1 byte.

- Algorithm type: The algorithm used to generate key pairs and sign the message. The length of this field is 1 byte. For ECC, this value is 0. Future algorithms will start at one and increase from there.

- Collision count: When a collision occurs during the DAD, the node will increment this value and store it in a file to be included in the sent packets for as long as the current IP address is valid. This value indicates to the node where it needs to start its check from, i.e., the first or second or third 64 bytes from the start of the hash value (digest) array of the public key.

- Subnet Prefix: This is the router subnet prefix.

- Hash Function: A hash function used to generate IID. The length of this field is 1 byte. For SHA256, this value is 0. Future algorithms will start at one and increase from there.

- Response No: This is similar to nonce but by the use of different mechanism. This value is not random and it is a copy of timestamp. In sender's message, this value MUST be set to zero and in response message (sent from a receiver node), this value MUST be set to the timestamp of the sender's message. The length of this field is 2 bytes. The sender node should cache this value in order to compare it with all responses sent by other nodes. This informs the sender node that the message is the response to his message and protects the node against replay attack.

- Other Options. This variable-length field contains important data that needs to be protected in the packet. The padding is used to insure that the field is a multiple of eight in length.

- Padding. A variable-length field containing padding to insure that the entire signature field is a multiple of eight in length. It thus contains the number of blanks needed to make the entire signature

field end on a multiple of eight.

All NDP messages (except RS messages) SHOULD contain the SSAS signature data field which allows receivers to verify senders. If a node receives a solicited NA message in response to its NS message showing that another node claims to own this address, then, after a successful verification process, this node increments the collision count by one and this value is used as explained in the "Collision count" item above. It will start from that section of the public key for the generation of a new IP address. The node repeats this 3 times and after 3 times generates a new public/private keys. Since the likelihood of two nodes having the same value is $1/(2^{63})$. This is really a small value while we also considered the order of magnitude relative to roughly 2 power 64 against sloppy implementations.

3.1.3. SSAS verification process

A node's verification process should start when it receives NDP messages. Following are the steps used in the verification process:

1. Obtain Response No from the sender's packet. Compare this value with its own timestamp that used in its previous message. If it is the same go to the next step, otherwise discard the message. (If SSAS is a part of SeND, this step should be skipped.)
2. Obtain prefix information from its own cache or from a router advertisement to make sure about the prefix sizes and number of bits used for IID.
3. Obtain the timestamp from the NDP message and call this value t1.
4. Obtain the timestamp from the node's system, convert it to UTC, and call this value t2.
5. If $(t2 - x) \leq t1 \leq (t2 + x)$ go to step 6. Otherwise, the message SHOULD be discarded without further processing. The value of x is dependent on network delays and network policy. The default value would be the value of Round Trip Time (RTT). The implementations SHOULD allow to set different values.
6. Obtain the public key from its own neighboring cache. If no matches are found in the node cache and if there is a centralized RPKI model available in the local network, then the node MIGHT obtain this public key from that node. Otherwise go to the next step.
7. Compare this to its own public key. If it is not the same, go to the next step. Otherwise, the message should be discarded without further processing. (This step should be skipped when the node uses the RPKI to obtain the other nodes' public key.)
8. Obtain the hash algorithm from the packet. By default it is SHA256.

9. Execute hash function on the public key. Takes 64bits, depends on collision count, from the hash function. Compare this value with the node's IID source IP. If it is the same, go to the next step. Otherwise, discard the message without further processing.

10. Concatenate the timestamp with the MAC address, algorithm type, collision count, sender's Global IP address (subnet prefix and IID), and other options (if any) and call this entity the plain message.

11. Obtain the SSAS signature from the SSAS signature data field. Obtain the Algorithm type from the message.

12. Verify the Signature using the public key and then enter the plain message and the SSAS signature as an input to the verification function. If the verification process is successful, process the message. Otherwise, the message should be discarded without further processing.

After a successful verification, the node SHOULD store the public key and MAC address of the sender node in its neighboring cache. By default, the cache is valid for two days but the implementation SHOULD consider a way to let the end users change this default value.

3.2. Resource Public key Infrastructure (RPKI)

To Authorize the Routers in the network and increase the security of the nodes in this network, it is recommended to use an RPKI explained in RFC 6494 and 6495. It is explained in more detail in [SSASAnalysis] and local security deployment [localSecurity].

4. SSAS Applications

4.1. A solution for all nodes

SSAS is capable to be used in standard nodes (standard computers) and nodes where limited computational resources are available. One example is the use of SSAS in sensor networks. Sensor networks are a prime example of nodes with limited resources (such as battery, CPU, and etc); see RFC 4919 [RFC4919] for use in IPv6 networks. Because currently, as explained in section 4. RFC 6775, the generation of the IID is based on EUI-64 which makes these nodes vulnerable to privacy and security attacks. One of these types of attack can occur during the Duplicate Address Detection (DAD) process.

4.2. Authentication in Network layer

Another example for the use of SSAS would be in mobile networks during the generation of IP addresses, as explained in section 4.4

RFC 6275 [RFC6275]. The current problem with the addressing mechanism in a mobile node is that no privacy is observed when a node moves to another network while usually keeping its Home Address. If there were a fast and secure mechanism available, then it would be possible to set this Home Address and change it and re-register it to the Home network. Another possible use for SSAS in mobile nodes could be as a security mechanism during the configuration of Care of Address (CoA); see section 3. RFC 5213 [RFC5213]. In that RFC, home proxy plays the role of a home agent for mobile nodes and mobile nodes set their CoA by the use of either stateful or stateless autoconfiguration. Currently they MUST use IPsec in order to secure this process. Section 4 of that RFC discusses the possibility of using another algorithm in order to secure mobile nodes.

4.3. Authentication in Application Layer

SSAS can be used as a means of authentication for the nodes in application layer. It is really important that the nodes know who they are talking to. This is because a user uses an application to connect to another node on the internet. This application either uses a domain name of the destination node (that later translates to the IP addresses) or directly uses the IP address of this node. This is where the attacker can play a role and spoof this IP address and play a MITM attack or other types of attacks. If the node uses this approach, the attacker does not have a possibility to spoof the IP address of the communicating node. So, this approach can mitigate IP spoofing during the authentication of two nodes in application layer.

4.4. Other Applications

With the wide usage of IP addresses in different types of devices and by the use of autoconfiguration mechanisms to configure these IP addresses, the need for the use of a security algorithm is increased. One type of application would be for use in vehicular networks or in the car-to-car networks. There is currently some work in progress that makes use of Neighbor Discovery. SSAS could also be a solution for enabling fast protection against ND attacks.

5. Security Considerations

There are two security considerations:

Since SSAS cannot prevent the layer 2 attacks but can mitigate it after the first verification, therefore one would need to use a monitoring device to prevent MAC spoofing. The other possibility is to have a dynamic MAC address. This means the SSAS node can use the 48 rightmost bits of the its public key as a MAC address. In this case there is a binding between the IP address, MAC address and public key. Since the verification process would have failed, it cannot be spoofed. However, this approach might be problematic from an operational view and might need to have some consideration before

being used.

Another security consideration is how to attack SSAS. One might ask oneself that what are the odds of an attacker being able to generate a public key having two four sequential bytes (from two different halves of public key) that are the same as 64 bits of that in Interface ID? If he could, he could then generate the signature using his own private key and thus break SSAS. Mathematically it has been shown that the likelihood of matching 64 bits in the public key against 64bits in the IID is $1/(2^{64})$. in [SSASAnalysis] the analysis of SSAS is explained and compared to CGA. Since the nodes in the network need to keep the public key and the MAC address of other nodes in the cache, the attacker only has a few seconds to perform this attack and then the attacker needs to perform this attack against the whole public key. For CGA, this value is less. in [cgaattack], the attack in CGA was explained. So, in general, SSAS is faster and in a good security level. In other word, SSAS tried to address the security and performance problem exists in CGA and offer a fastest algorithm.

6. IANA Considerations

There is no IANA consideration

7. Privacy Consideration

When an attacker is inside a local link, he is enable to identify a node. although, this target node changes its IP address. The reason is because the target node does not change its MAC address. However, if the public key needs to be used for verification in other mechanisms and not in local link, then it is RECOMMENDED that the public/private keys to be valid for a short period of time. The default value would be a week. The implementations need to consider the automatic key generation to avoid administrative requirements for this process.

8. Appendix A

8.1. Comparison of CGA and SSAS generation time

The following information was retrieved from [cgatime]. It shows the time required to generate CGA in different sec value. This is why, in practice, only sec value 0 and 1 can be used.

sec value 1 => ~ 1 second

sec value 2 => ~ 3 hours

sec value 3 => ~ 24 years

sec value 4 => ~ 1.16×10^6 years

sec value 5 => ~ 1×10^{11} years

sec value 6 => ~ 6.8×10^{15} years

The above information is based on the fact that one uses RSA key sizes less than 1280 bits. If one needs to use the higher security, then it needs more time for the generation of CGA value. Using RSA higher key sizes also increases the packet size needs to carry the public key. Here is our evaluation of ECC and RSA key generation time in a standard computer with 2.6 GHz CPU.

SSAS generation time is about the time needed to generate key pairs. Since, by default, SSAS uses ECC 192 bits, the following values compares ECC with RSA. RSA is the algorithm uses in CGA. As explained earlier, the security of ECC 192 bits is equivalent to the security of RSA 7680 bits.

ECC 192 bits: Average key generation time = 195011 microseconds

RSA 1280 bits: Average key generation time = 681039 microseconds

RSA 7680 bits: Average key generation time = 163473350 microseconds

9. Appendix B

9.1. Network-based protection vs. Node-based protection

Node-based protection is the ability of the node to protect against some types of attacks such as IP spoofing, MITM attack. On the other hand, network-based protection is the use of some devices in the network edges to protect the nodes inside this network against router advertisement spoofing attacks or other types of attack. Both of these protection is required and both can complement each other. This is because the attacker might be inside the network and play a role of MITM, spoof the other nodes' IP address, prevent other nodes from configuring their IP address and cause many delays and problems in the local network (Not all the nodes in the network is ever trustee). One important consideration about node-based protection is that, it should support any node and apply to any nodes (Including nodes with limited energy resources or limited memory resources). This is why there is a need for a good mechanism to provide this protection with less cost. The proposed mechanism in this document, i.e., SSAS can provide the node with node-based protection. With only node-based protection, the malicious node inside this network can claim to be a router and the node does not have any means to authorize him. This is

why, the network-based protection is also the complement solution to a node-based protection. There are some approaches to provide the node with network-based protection. One such approach might be RA-gaurd [RAgaurd] which limits subnet prefixes. Unfortunately with this approach, still the node inside this network can maliciously claim to be a router and play the MITM attack inside the network by sending unicast router advertisement messages. So, the attack is still possible. The other approach is the use of RPKI as explained in RFC 6494 and RFC 6495. Unfortunately these RFCs only explain the possibility of using them but not the detail of implementation. The detail implementation is explained in [SSASAnalysis]. The local RPKI node also can play a role of monitoring device in the network.

10. Acknowledgements

The Authors would like to acknowledge Erik Nordmard and Joel M. Halpern for their supports and assistance to improve this document. The authors also would like to acknowledge Michael Richardson, Dan Wing, Tim Chown, Christian Huitema, Joel M. Halpern for their comments to improve this document

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4291] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture," RFC 4291, February 2006.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4941] Narten, T., Draves, R., Krishnan, S., "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and Nikander, P., "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3756] Nikander, P., Kempf, J., Nordmark, E., "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3972, May 2004.
- [RFC4919] Kushalnagar, N., Montenegro, G., Schumacher, C., "IPv6 over Low-Power Wireless Personal Area Networks

(6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., Bormann, C., "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[RFC6275] Perkins, C., Johnson, D., Arkko, J., "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6543] Gundavelli, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, May 2012.

[RFC6090] McGrew, D., Igoe, K., Salter, M., "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2012.

[RFC3756] Nikander, F., Kempf, J., Nordmark, E., "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[RFC7136] Carpenter, B., Jiang, S., "Significance of IPv6 Interface Identifiers", RFC 7136, 2013

11.2. Informative References

[SSASAnalysis] Rafiee, H., Meinel, C., "'SSAS: a Simple Secure Addressing Scheme for IPv6 AutoConfiguration". In Proceedings of the 11th IEEE International conference on Privacy, Security and Trust (PST), IEEE Catalog number: CFP1304F-ART, ISBN: 978-1-4673-5839-2.

[cgaattack] Rafiee, H., Meinel, C., "Possible Attack on Cryptographically Generated Addresses (CGA)", <http://tools.ietf.org/html/draft-rafiee-6man-cga-attack>, 2014

[RAguard] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", <http://tools.ietf.org/html/draft-ietf-v6ops-ra-guard-implementation>, 2012

[localSecurity] Rafiee, H., Meinel, C., "Recommendations for Local Security Deployments", <http://tools.ietf.org/html/draft-rafiee-6man-local-security>, 2013

[cgatime] Bos, J., Oezen, O., Hubaux, J., "Analysis and Optimization of Cryptographically Generated Addresses", In Proceedings of the 12th International conference on Information Security (2009), ACM, pp. 17 ? 32.

[variableprefix] Carpenter, B., Chown, T, Gont, F.,
Jiang, S., Petrescu, A., Yourtchenko, A., " Analysis
of the 64-bit Boundary in IPv6 Addressing",
<http://tools.ietf.org/html/draft-ietf-6man-why64> ,
April 2014

[cgabis] Rafiee,H., Zhang, D., "CGA Security Improvement"
,<http://tools.ietf.org/html/draft-rafiee-rfc3972-bis>, 2014

Authors' Addresses

Hosnieh Rafiee
HUAWEI TECHNOLOGIES Duesseldorf GmbH
Riesstrasse 25, 80992,
Munich, Germany
Phone: +49 (0)162 204 74 58
Email: hosnieh.rafiiee@huawei.com

Christoph Meinel
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
Email: meinel@hpi.uni-potsdam.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2014

B. Sarikaya
Huawei USA
July 8, 2013

IPv6 RA Options for Next Hop Routes
draft-sarikaya-6man-rfc4191bis-00

Abstract

This proposes an update on RFC 4191 in order to define new Router Advertisement options for configuring next hop routes on the mobile or fixed nodes. Using these options, an operator can easily configure nodes with multiple interfaces (or otherwise multi-homed) to enable them to select the routes to a destination. Each option is defined together with definitions of host and router behaviors.

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 9, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Default Route Configuration	4
4. Source Address Dependent Routing	5
5. Host Configuration	5
6. Router Configuration	6
7. Route Prefix option	7
8. Next Hop Address option	7
9. Source Address option	8
10. Source Prefix option	8
11. Next Hop Address with Route Prefix option	9
12. Next Hop Address with Source Prefix and Route Prefix option .	10
13. Next Hop Address with Source Address and Route Prefix option	11
14. Security Considerations	11
15. IANA Considerations	11
16. Acknowledgements	12
17. References	13
17.1. Normative References	13
17.2. Informative References	13
Author's Address	15

1. Introduction

IPv6 Neighbor Discovery and IPv6 Stateless Address Autoconfiguration protocols can be used to configure fixed and mobile nodes with various parameters related to addressing and routing [RFC4861], [RFC4862], [RFC4191]. DNS Recursive Server Addresses and Domain Name Search Lists are additional parameters that can be configured using router advertisements [RFC6106].

Router Advertisements can also be used to configure fixed and mobile nodes in multi-homed scenarios with route information and next hop address. Different scenarios exist such as the node is simultaneously connected to multiple access network of e.g. WiFi and 3G. The node may also be connected to more than one gateway. Such connectivity may be realized by means of dedicated physical or logical links that may also be shared with other users nodes such as in residential access networks.

2. Terminology

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

3. Default Route Configuration

A host, usually a mobile host interested in obtaining routing information usually sends a Router Solicitation (RS) message on the link. The router, when configured to do so, provides the route information using zero, one or more Next Hop Address and Route Information options in the router advertisement (RA) messages sent in response.

The route options are extensible, as well as convey detailed information for routes.

RS and RA exchange is for next hop address and route information determination and not for determining the link-layer address of the router. Subsequent Neighbor Solicitation and Neighbor Advertisement exchange can be used to determine link-layer address of the router.

It should be noted that the proposed options in this document will need a central site-wide configuration mechanism. The required values can not automatically be derived from routing tables.

Next hop address and related route information may be provided by some other means such as directly by the next hop routers. In this document we assume that next hop routers are not able to provide this information. One solution would be to develop an inter-router protocol to instigate the next hop routers to provide this information. However, such a solution has been singled out due to the complexities involved.

A non-trustworthy network may be available at the same time as a trustworthy network, with the risk of bad consequences if the host gets confused between the two. These are basically the two models for hosts with multiple interfaces, both of which are valid, but which are incompatible with each other. In the first model, an interface is connected to something like a corporate network, over a Virtual Private Network (VPN). This connection is trusted because it has been authenticated. Routes obtained over such a connection can probably be trusted, and indeed it may be important to use those routes. This is because in the VPN case, you may also be connected to a network that's offered you a default route, and you could be attacked over that connection if you attempt to connect to resources on the enterprise network over it.

On the other, non-trustworthy network scenario, none of the networks to which the host is connected are meaningfully more or less trustworthy. In this scenario, the untrustworthy network may hand out routes to other hosts, e.g. those in the VPN going through some malicious nodes. This will have bad consequences because the host's

traffic intended for the corporate VPN may be hijacked by the intermediate nodes.

Router advertisement extensions described in this document can be used to install the routes. However, the use of such a technique makes sense only in the former case above, i.e. trusted network. So the host MUST have an authenticated connection to the network it connects so that the router advertisements can be trusted before establishing routes.

4. Source Address Dependent Routing

In multihomed networks there is a need to do source address based routing if some providers are performing the ingress filtering defined in BCP38 [RFC2827]. This requires the routers to consider the source addresses as well as the destination addresses in determining the next hop to send the packet to.

The routers may be informed about the source addresses to use in routing using extensions to the routing protocols like IS-IS defined in [ISO.10589.1992] [I-D.baker-ipv6-isis-dst-src-routing] and OSPF defined in [RFC5340] [I-D.baker-ipv6-ospf-dst-src-routing]. In this document we define the router advertisement extensions for source address dependent routing.

Routing protocol extensions for source address dependent routing does not avoid a host using a source address that may be subject to ingress filtering when sending a packet to one of the next hops. In that case the host receives an ICMP source address failed ingress/egress policy error message in which case the host must resend the packet trying a different source address. The extensions defined in this document aims at avoiding this inefficiency in packet forwarding at the host.

5. Host Configuration

Router advertisement options defined in this document are used by Type C hosts.

As defined in [RFC4191] Type C host uses a Routing Table instead of a Default Router List.

The hosts set up their routing tables based on the router advertisement extensions defined in this document. The routes established are used in forwarding the packets to a next hop based on the destination prefix/address using the longest match algorithm.

In case the host receives Next Hop Address with Source Address and Route Prefix option, the host uses source and destination prefix/address using the longest match algorithm in order to select the next hop to forward the packet to.

6. Router Configuration

The router MAY send one or more Next Hop Address options that specify the IPv6 next hop addresses. Each Next Hop Address option may be associated with zero, one or more Route Prefix options that represent the IPv6 destination prefixes reachable via the given next hop. Router includes Route Prefix option in message to indicate that given prefix is available directly on-link.

Router MAY send a single Next Hop Address without any Route Prefix options. When router sends Next Hop Address option that is associated with Router Prefix option, the router MUST use Next Hop and Route Prefix option defined in Section 11. The Route Prefix MAY contain ::/0, i.e. with Prefix Length set to zero to indicate available default route.

The router MAY send one or more Next Hop Address options that specify the IPv6 next hop addresses and source address. Each Next Hop Address option may be associated with zero, one or more Source Address options that represent the source addresses that are assigned from the prefixes that belong to this next hop defined in Section 13. In addition, the option MAY contain Route Prefix options that represent the IPv6 destination prefixes reachable via the given next hop. Router includes Source Address option and Route Prefix option in the message to indicate that given prefix is available directly on-link and that the source address will not be subject to ingress filtering.

The router MAY send one or more Next Hop Address options that specify the IPv6 next hop addresses and source address. Each Next Hop Address option may be associated with zero, one or more Source Prefix options that represent the source addresses that are assigned from the prefixes that belong to this next hop defined in Section 12. In addition, the option MAY contain Route Prefix options that represent the IPv6 destination prefixes reachable via the given next hop. Router includes Source Prefix option and Route Prefix option in the message to indicate that given prefix is available directly on-link and that any source addresses derived from the source prefix will not be subject to ingress filtering on these routes supported by these next hops.

7. Route Prefix option

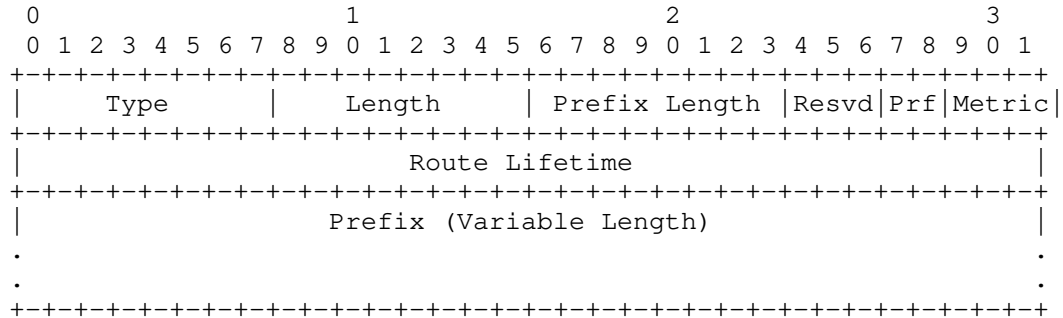


Figure 1: Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the Type and Length fields) in units of 8 octets.

Other fields are as in [RFC4191] except:

Metric Route Metric. 3-bit signed integer. The Route Metric indicates whether to prefer the next hop associated with this prefix over others, when multiple identical prefixes (for different next hops) have been received.

8. Next Hop Address option

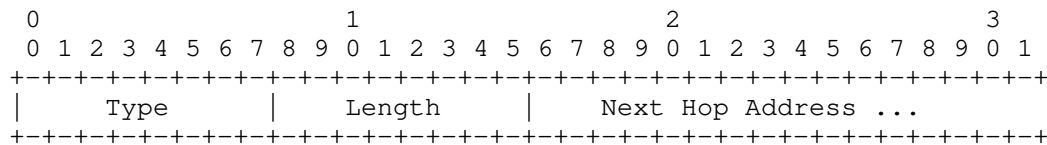


Figure 2: Next Hop Address option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. It's value is 3.

Next Hop Address: An IPv6 address that specifies IPv6 address of the next hop. It is 16 octets in length.

9. Source Address option

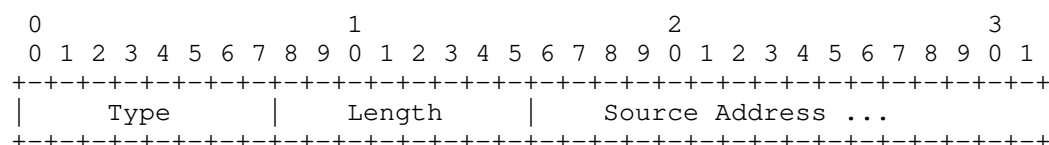


Figure 3: Source Address option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. It's value is 3.

Source Address: An IPv6 address that specifies the source IPv6 address. It is 16 octets in length.

10. Source Prefix option

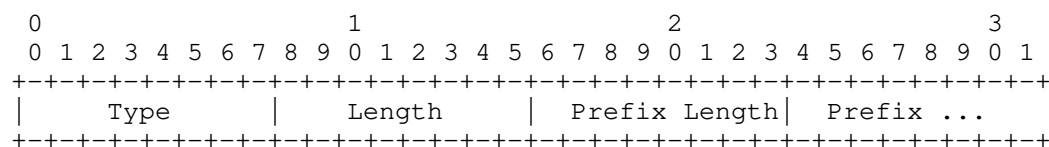


Figure 4: Source Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. It's value is 3.

Prefix Length: An IPv6 prefix length in bits, from 0 to 128.

Prefix: An IPv6 prefix that specifies the source IPv6 prefix. It is 16 octets or less in length.

11. Next Hop Address with Route Prefix option

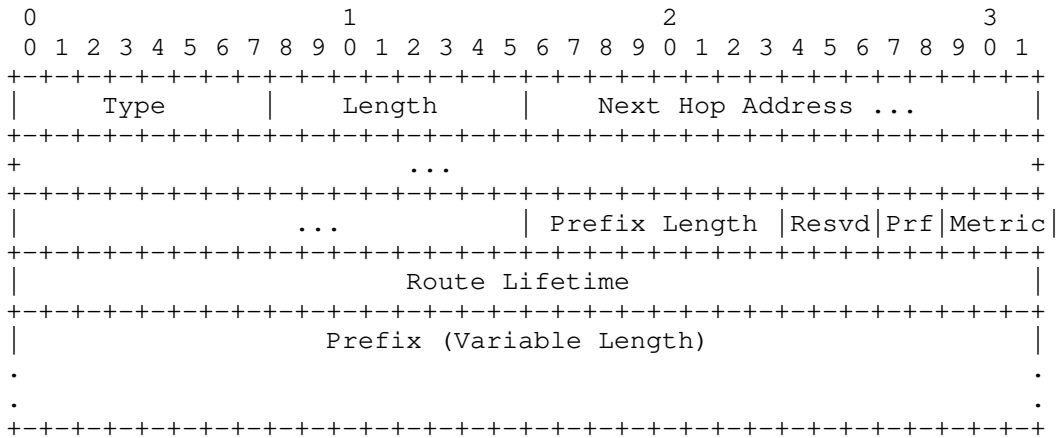


Figure 5: Next Hop Address with Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. For example, the length for a prefix of length 16 is 5.

Other fields are as in Section 7 and Section 8.

12. Next Hop Address with Source Prefix and Route Prefix option

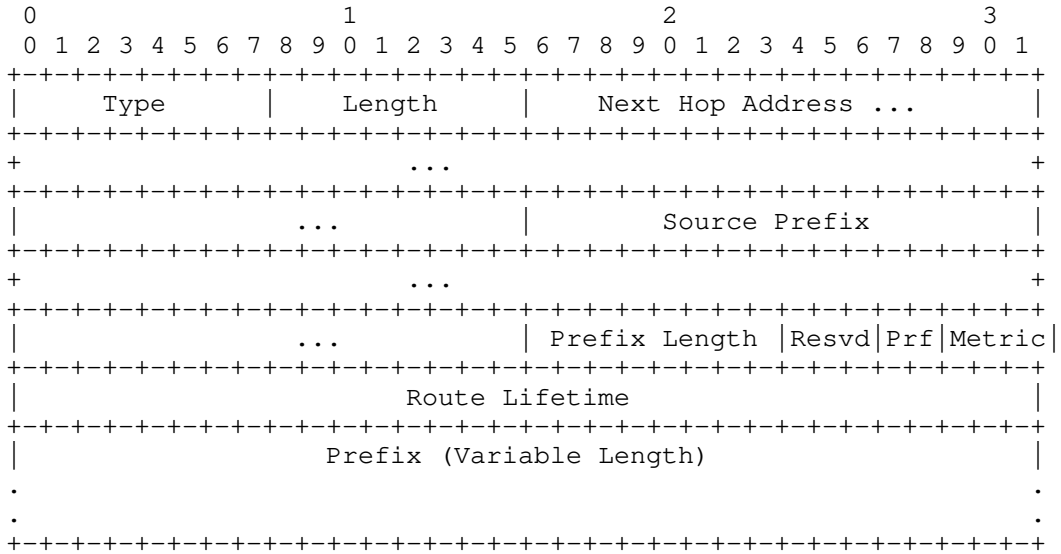


Figure 6: Next Hop Address with Source Prefix and Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. For example, the length for a prefix of length 16 is 5.

Other fields are as in Section 7, Section 8 and Section 10.

13. Next Hop Address with Source Address and Route Prefix option

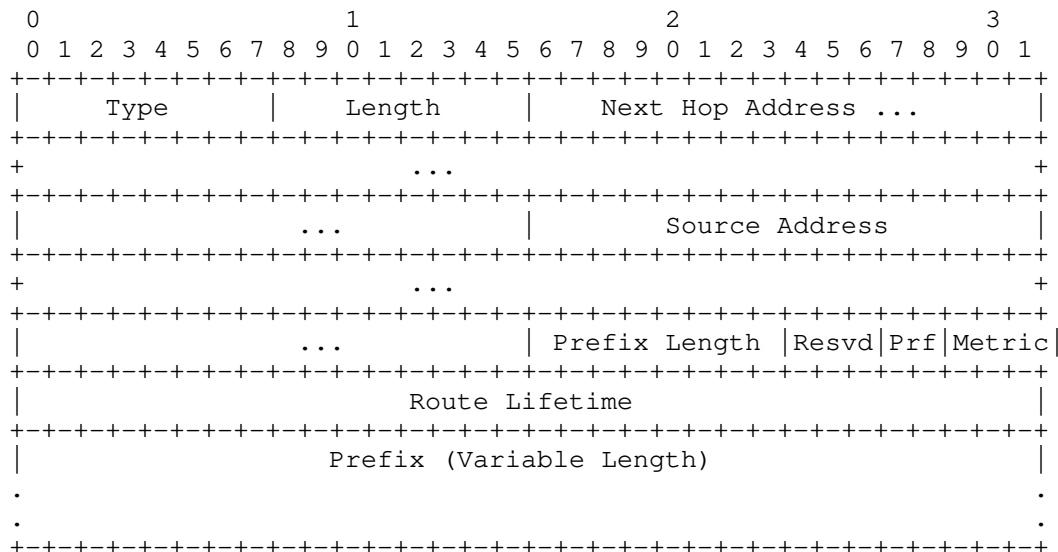


Figure 7: Next Hop Address with Source Address and Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. For example, the length for a prefix of length 16 is 5.

Other fields are as in Section 7, Section 8 and Section 9.

14. Security Considerations

Neighbor Discovery is subject to attacks that cause IP packets to flow to unexpected places. Because of this, neighbor discovery messages MUST be secured, possibly using Secure Neighbor Discovery (SEND) protocol [RFC3971].

15. IANA Considerations

Authors of this document request IANA to assign the following new RA options:

Option Name	Type
Route Prefix	
Next Hop Address	
Source Address	
Source Prefix	
Next Hop Address and Route Prefix	
Next Hop Address with Source Address and Route Prefix	
Next Hop Address with Source Prefix and Route Prefix	

Table 1:

16. Acknowledgements

TBD.

17. References

17.1. Normative References

- [ISO.10589.1992]
International Organization for Standardization,
"Intermediate system to intermediate system intra-domain-
routing routine information exchange protocol for use in
conjunction with the protocol for providing the
connectionless-mode Network Service (ISO 8473), ISO
Standard 10589", ISO ISO.10589.1992, 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:
Defeating Denial of Service Attacks which employ IP Source
Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure
Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and
More-Specific Routes", RFC 4191, November 2005.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick,
"Internet Group Management Protocol (IGMP) / Multicast
Listener Discovery (MLD)-Based Multicast Forwarding
("IGMP/MLD Proxying")", RFC 4605, August 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.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF
for IPv6", RFC 5340, July 2008.

17.2. Informative References

- [I-D.baker-ipv6-isis-dst-src-routing]
Baker, F., "IPv6 Source/Destination Routing using IS-IS",
draft-baker-ipv6-isis-dst-src-routing-00 (work in

progress), February 2013.

[I-D.baker-ipv6-ospf-dst-src-routing]

Baker, F., "IPv6 Source/Destination Routing using OSPFv3",
draft-baker-ipv6-ospf-dst-src-routing-02 (work in
progress), May 2013.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli,
"IPv6 Router Advertisement Options for DNS Configuration",
RFC 6106, November 2010.

Author's Address

Behcet Sarikaya
Huawei USA
5340 Legacy Dr. Building 175
Plano, TX 75024

Phone:
Email: sarikaya@ieee.org

6MAN
Internet-Draft
Intended status: Best Current Practice
Expires: December 17, 2015

W. Kumari
Google
J. Jaeggli
Zynga
R. Bonica
Juniper Networks
J. Linkova
Google
June 15, 2015

Operational Issues Associated With Long IPv6 Header Chains
draft-wkumari-long-headers-03

Abstract

This memo specifies requirements for IPv6 forwarders as they process packets with long header chains. It also provides guidance for application developers whose applications might rely on long headers chains.

As background, this memo explains how many ASIC-based IPv6 forwarders process packets and why processing of packets with long header chains might be problematic.

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 December 17, 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
1.1. Terminology	4
2. Forwarder Information Requirements	4
3. Requirements For IPv6 Forwarders	5
4. Recommendations For Application Developers	7
5. IANA Considerations	7
6. Security Considerations	7
7. Acknowledgements	8
8. References	8
8.1. Normative References	8
8.2. Informative References	8
Appendix A. Changes / Author Notes.	8
Authors' Addresses	9

1. Introduction

IPv6 [RFC2460] forwarders can acquire information from the following sources:

- o The IPv6 header
- o One or more IPv6 extension headers
- o An upper-layer header

Section 2 of this document explains how IPv6 forwarders use information from the IPv6 header and IPv6 extension headers to provide traditional forwarding services. It also explains how IPv6 forwarders use information from the upper-layer header to provide enhanced forwarding services.

When a software-based forwarder processes an IPv6 datagram, it parses the header chain, regardless of its length, acquires the required information and makes a forwarding decision. Typically, software-based forwarders process a relatively small number of packets per second. Therefore, they can perform the above mentioned procedure within the constraints of their processing budget.

By contrast, ASIC-based forwarders process many more packets per second. In order to fulfill this requirement, ASIC-based forwarders copy a fixed number of bytes from the beginning of the packet to on-chip memory. Forwarders do this because they can access on-chip memory much more quickly than they can access off-chip memory. Once the beginning of the packet has been transferred to on-chip memory, subsequent processing can proceed very quickly.

The act of copying bytes from the beginning of a packet to on-chip memory consumes:

- o Processor cycles
- o On-chip memory
- o Wall-time

Therefore, the number of bytes copied to on-chip memory must be chosen wisely. If a forwarder copies more bytes than it needs, it wastes resources and adversely impacts performance. If it copies too few bytes, it may not have sufficient information to make a correct forwarding decision.

The IPv6 header chain is a variable-length data structure, whose size can exceed 64 kilobytes. However, packets with header chains exceeding 256 bytes are rarely observed on the Internet. Therefore, most ASIC-based forwarders copy a relatively small number of bytes from the beginning of a packet into on-chip memory. While this small number varies from platform to platform, it is generally much closer to 256 bytes than it is to 64 kilobytes.

IPv6 forwarders MUST behave in a predictable manner when they process a packet whose header chain length exceeds the number of bytes copied to on-chip memory. Section 3 of this memo defines required behaviors.

Application developers should be aware of how ASIC-based forwarders process packets with long extension header chains. Therefore, Section 4 of this document provides guidance to application developers.

1.1. Terminology

For the purposes of this document, the terms "header chain" and "upper-layer" header are used as defined in [RFC7112].

This document also introduces the following terms:

- o forwarding service - a service that accepts a packet from one interface and forwards it through another
- o traditional forwarding service - a forwarding service in which all parameters to the forwarding algorithm are drawn from the IPv6 header, the hop-by-hop extension header, and the routing extension header
- o enhanced forwarding service - a forwarding service in which parameters to the forwarding algorithm can be drawn from any portion of the IPv6 header chain

2. Forwarder Information Requirements

When an IPv6 forwarder provides traditional forwarding services, it extracts all information required by the forwarding algorithm from the IPv6 header, the hop-by-hop extension header (if present), and the routing extension header (if present). In the nominal case, the IPv6 header contains all information required by the forwarding algorithm. However, the hop-by-hop and routing extension headers can also impact forwarding behavior.

Section 4.2 of [RFC2460] explains how the hop-by-hop extension header impacts forwarding behavior. When the forwarder processes a hop-by-hop extension header, it examines each option contained by the header. If forwarder encounters an unrecognized hop-by-hop option, and the high-order bits of the option type are "00", the forwarder skips over the option and continues to process subsequent options. However, if an forwarder encounters an unrecognized option, and the high-order bits of the option type are "01", "10" or "11", the forwarder discards the packet.

Section 4.4 of [RFC2460] explains how the routing extension header impacts forwarding behavior. When the forwarder processes a packet whose destination address is local to itself, it scans the header chain, searching for a routing extension header. If the packet contains a routing extension header and the forwarder recognizes the routing header type, it processes the header. If the forwarder does not recognize the routing header type, the required behavior depends upon the Segments Left field. If the Segments Left field is equal to zero, the forwarder ignores the routing extension header. Otherwise,

the forwarder discards the packet. [RFC6275] and [RFC6554] describe currently defined routing extension header types.

Some IPv6 forwarders provide enhanced forwarding services, such as firewall filtering, rate limiting and load balancing. In order to provide these services, the forwarder requires access to an upper layer header. The following are examples of enhanced services that require the forwarder to examine the upper layer header:

- o Discard all packets directed to TCP port 25
- o Rate limit packets destined for a particular address whose payload is TCP and have the TCP SYN bit set
- o Load balance packets across parallel links so that all packet belonging to particular TCP session traverse the same link.

3. Requirements For IPv6 Forwarders

The following requirements apply to all IPv6 forwarders:

- o REQ-1: By default an IPv6 forwarder SHOULD NOT discard a valid packet because of its header chain length. However, the forwarder MAY support a configuration option that causes it to discard packets whose header chain length exceeds a specified value.
- o REQ-2: When processing packet that contains a hop-by-hop extension header, an IPv6 forwarder MUST process the entire hop-by-hop extension header, regardless of its length. The forwarder MUST process each option as specified in Section 4.2 of [RFC2460]. If an IPv6 forwarder is not able to process the entire hop-by-hop extension header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them. If an IPv6 forwarder is not able to process the entire hop-by-hop extension header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them.
- o REQ-3: When processing a packet whose destination address is local to itself, an IPv6 forwarder MUST scan the entire header chain, regardless of its length, in order to determine whether the packet contains a routing extension header. If the packet contains a routing extension header, the forwarder MUST process routing extension header as specified in Section 4.4 of [RFC2460]. If an IPv6 forwarder is not able to process the entire routing extension

header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them.

The length of the IPv6 header plus the length of the hop-by-hop extension header can exceed the number of bytes that an ASIC-based forwarder copies into on-chip memory. Therefore, in order to support REQ-2, ASIC-based forwarders typically support a special processing mechanism for packets containing hop-by-hop extensions.

Also, the combined length of all headers preceding the routing extension header may exceed the number of bytes that an ASIC-based forwarder copies into on-chip memory. Therefore, in order to support REQ-3, ASIC-based forwarders typically support a special processing mechanism for packets whose IPv6 destination address is local to the forwarder. This forwarding mechanism is capable of processing the routing extension header, even if it begins beyond of the portion of the packet that was copied to on-chip memory.

The following requirements apply to IPv6 forwarders that provide enhanced forwarding services:

- o REQ-4: If a forwarder's ability to deliver enhanced services is limited in any way by extension header length, that limitation MUST be reflected in user documentation. For example, assume that a forwarder provides a load balancing service, and that it acquires information required by the service from the IPv6 header and the upper-layer header. If the service behaves in one manner when all required information is contained by the first N bytes of the header chain and in another manner when all required information is not contained by the first N bytes of the header chain, user documentation MUST reflect both behaviors as well as the value of N.
- o REQ-5: If a forwarder's ability to deliver an enhanced service is limited by extension header length, the policy specification language used to configure the enhanced service MUST be sufficiently robust to address the limitation. For example, assume that the forwarder provides a firewall service. The firewall service is capable of filtering packets directed to a particular TCP port, but only if the TCP header is contained by the first N bytes of the header chain. In this case, it MUST be possible to configure one policy for packets directed to the specified port, another policy for packet not directed to the specified port, and a third policy for packets whose TCP destination port is unknown.

4. Recommendations For Application Developers

Applications developers should be aware that many ISPs and enterprises filter or severely rate limit packets containing long header chains. They do this because of limitations imposed by the ASIC-based forwarders deployed at their edges. ISPs and enterprises accept these limitations as part of an engineering trade off, in which high-speed forwarding is achieved at the cost of limiting enhanced services for packets with long extension headers.

For example, assume that an enterprise deploys the following firewall filtering policy at its edge:

- o Permit all packets whose destination is TCP port 80
- o Discard all packets whose destination is not TCP port 80
- o Discard all packets whose header chain is so long that TCP port information is not accessible to the filtering function

In this case, the enterprise discards all packets whose destination cannot be determined by the filtering function.

Aside from the issue of header chain length, operators may filter packets containing extension headers that may either compromise the network's security posture or require inordinate processing resources.

This memo does not specify a maximum header chain length. However, this memo does note that at the time of its publication, the number of bytes that ASIC-based forwarders copy from the beginning of a packet to on-chip memory varies from platform to platform. Typical platforms copy between 128 and 384 bytes. Therefore, application developers should avoid sending packets whose header chain length is in that range, unless they have some assurance that their packets will not be discarded.

5. IANA Considerations

This document makes no requests of the IANA

6. Security Considerations

TBD

7. Acknowledgements

The authors wish to thank Paul Hoffman, KK and Fernando Gont. The authors also express their gratitude to an anonymous donor, without whom this document would not have been written.

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.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, January 2014.

8.2. Informative References

- [RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, March 2012.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

Template to -00

- o Initial submission.

- o

-00 to -01

- o Added maximum header chain recommendation.

- o Rewrite the forwarding description.
- 02 to -03
- o Updating REQ2 and REQ3 with sending ICMPv6 messages part.

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Joel Jaeggli
Zynga
675 East Middlefield
Mountain View, CA
USA

Email: jjaeggli@zynga.com

Ronald P Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, VA
USA

Email: rbonica@juniper.net

J. Linkova
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Email: furry@google.com