

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.

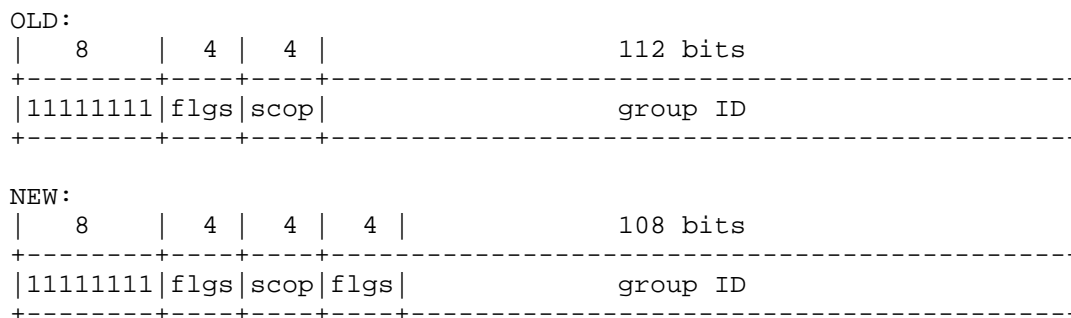


Figure 1: Updated IPv6 Multicast Addressing Architecture

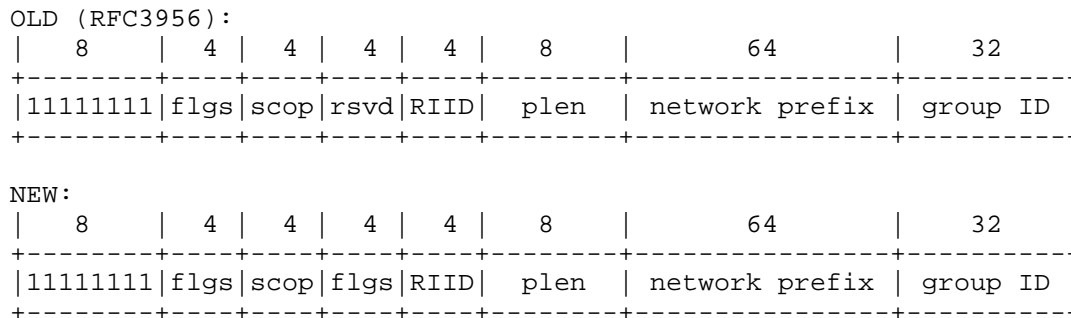


Figure 2: Embedded-RP with Updated IPv6 Multicast Address Arch.

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

IPv6 Maintenance WG
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2013

A. Brandt
J. Buron
Sigma Designs
June 18, 2013

Transmission of IPv6 packets over ITU-T G.9959 Networks
draft-brandt-6man-lowpanz-02

Abstract

This document describes the frame format for transmission of IPv6 packets and a method of forming IPv6 link-local addresses and statelessly autoconfigured IPv6 addresses on ITU-T G.9959 networks.

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].

Status of This Memo

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

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

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

This Internet-Draft will expire on December 20, 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. Author's notes	2
1.1. Reader's guidance	2
2. Introduction	3
2.1. Terms used	3
3. G.9959 parameters to use for IPv6 transport	4
3.1. Addressing mode	4
3.2. IPv6 Multicast support	4
3.3. G.9959 MAC PDU size and IPv6 MTU	5
3.4. Transmission status indications	5
3.5. Transmission security	5
4. LoWPAN Adaptation Layer and Frame Format	6
4.1. Dispatch Header	6
5. LoWPAN addressing	7
5.1. Stateless Address Autoconfiguration of routable IPv6 addresses	8
5.2. IPv6 Link Local Address	8
5.3. Unicast Address Mapping	8
5.4. On the use of Neighbor Discovery technologies	9
5.4.1. Prefix and CID management (Route-over)	10
5.4.2. Prefix and CID management (Mesh-under)	10
6. Header Compression	10
7. IANA Considerations	11
8. Security Considerations	11
9. Acknowledgements	12
10. References	12
10.1. Normative References	12
10.2. Informative References	13
Authors' Addresses	14

1. Author's notes

This chapter MUST be deleted before going for document last call.

1.1. Reader's guidance

This document borrows heavily from RFC4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks". The process of creating this document was mainly a simplification; removing the following topics:

- o EUI-64 link-layer addresses

- o Fragmentation layer
- o Mesh routing

The 16-bit short addresses of 802.15.4 have been changed to 8-bit G.9959 NodeIDs.

2. Introduction

The ITU-T G.9959 recommendation [G.9959] targets low-power Personal Area Networks (PANs). This document defines the frame format for transmission of IPv6 [RFC2460] packets as well as the formation of IPv6 link-local addresses and statelessly autoconfigured IPv6 addresses on G.9959 networks.

The general approach is to adapt elements of [RFC4944] to G.9959 networks. G.9959 provides a Segmentation and Reassembly (SAR) layer for transmission of datagrams larger than the G.9959 MAC PDU.

[RFC6775] updates [RFC4944] by specifying 6LoWPAN optimizations for IPv6 Neighbor Discovery (ND) (originally defined by [RFC4861]). This document limits the use of [RFC6775] to prefix and Context ID assignment. It is described how to construct an IID from a G.9959 link-layer address. Refer to Section 5. If using that method, Duplicate Address Detection (DAD) is not needed. Address registration is only needed in certain cases.

In addition to IPv6 application communication, the frame format defined in this document may be used by IPv6 routing protocols such as RPL [RFC6550] or P2P-RPL [P2P-RPL] to implement IPv6 routing over G.9959 networks.

G.9959 networks may implement mesh routing between nodes below the IP layer. Mesh routing is out of scope of this document.

2.1. Terms used

ABR: Authoritative Border Router ([RFC6775])

AES: Advanced Encryption Scheme

EUI-64: Extended Unique Identifier

HomeID: G.9959 Link-Layer Network Identifier

IID: Interface IDentifier

MAC: Media Access Control

MTU: Maximum Transmission Unit

NodeID: G.9959 Link-Layer Node Identifier (Short Address)

PAN: Personal Area Network

PDU: Protocol Data Unit

SAR: Segmentation And Reassembly

ULA: Unique Local Address

3. G.9959 parameters to use for IPv6 transport

This chapter outlines properties applying to the PHY and MAC of G.9959 and how to use these for IPv6 transport.

3.1. Addressing mode

G.9959 defines how a unique 32-bit HomeID network identifier is assigned by a network controller and how an 8-bit NodeID host identifier is allocated. NodeIDs are unique within the logical network identified by the HomeID. The logical network identified by the HomeID maps directly to an IPv6 subnet identified by one or more IPv6 prefixes.

An IPv6 host SHOULD construct its link-local IPv6 address and routable IPv6 addresses from the NodeID in order to facilitate IP header compression as described in [RFC6282].

A word of caution: since HomeIDs and NodeIDs are handed out by a network controller function during inclusion, identifier validity and uniqueness is limited by the lifetime of the logical network membership. This can be cut short by a mishap occurring to the network controller. Having a single point of failure at the network controller suggests that deployers of high-reliability applications should carefully consider adding redundancy to the network controller function.

3.2. IPv6 Multicast support

[RFC3819] recommends that IP subnetworks support (subnet-wide) multicast. G.9959 supports direct-range IPv6 multicast while subnet-wide multicast is not supported natively by G.9959. Subnet-wide multicast may be provided by an IP routing protocol or a mesh routing protocol operating below the 6LoWPAN layer. Routing protocols are out of scope of this document.

IPv6 multicast packets MUST be carried via G.9959 broadcast.

As per [G.9959], this is accomplished as follows:

1. The destination HomeID of the G.9959 MAC PDU MUST be the HomeID of the logical network
2. The destination NodeID of the G.9959 MAC PDU MUST be the broadcast NodeID (0xff)

G.9959 broadcast MAC PDUs are only intercepted by nodes within the logical network identified by the HomeID.

3.3. G.9959 MAC PDU size and IPv6 MTU

IPv6 packets MUST use G.9959 transmission profiles which support MAC PDU payload sizes of 150 bytes or higher, e.g. the R3 profile. G.9959 profiles R1 and R2 only supports MPDU payloads around 40 bytes and the transmission speed is down to 9.6kbit/s.

[RFC2460] specifies that IPv6 packets may be up to 1280 octets. However, a full IPv6 packet does not fit in an G.9959 MAC PDU. The maximum G.9959 R3 MAC PDU payload size is 158 octets. Link-layer security imposes an overhead, which in the extreme case leaves 130 octets available.

G.9959 provides Segmentation And Reassembly for payloads up to 1350 octets. Segmentation however adds further overhead. It is therefore desirable that datagrams can fit into a single G.9959 MAC PDU. IPv6 Header Compression [RFC6282] improves the chances that a short IPv6 packet can fit into a single G.9959 frame.

3.4. Transmission status indications

The G.9959 MAC layer provides native acknowledgement and retransmission of MAC PDUs. The G.9959 SAR layer does the same for larger datagrams. A mesh routing layer may provide a similar feature for routed communication. Acknowledgment and retransmission improves the transmission success rate and frees higher layers from the burden of implementing individual retransmission schemes. An IPv6 routing stack communicating over G.9959 may utilize link-layer status indications such as delivery confirmation and Ack timeout from the MAC layer.

3.5. Transmission security

Implementations claiming conformance with this document MUST enable G.9959 shared network key security.

The shared network key is intended to address security requirements in the home at the normal security requirements level. For applications with high or very high requirements on confidentiality and/or integrity, additional application layer security measures for end-to-end authentication and encryption may need to be applied. The availability of the network relies on the security properties of the network key in any case.

4. LoWPAN Adaptation Layer and Frame Format

The 6LoWPAN encapsulation formats defined in this chapter are the payload in the G.9959 MAC PDU. IPv6 header compression [RFC6282] MUST be supported by implementations of this specification.

All 6LoWPAN datagrams transported over G.9959 are prefixed by a 6LoWPAN encapsulation header stack. The 6LoWPAN payload (e.g. an IPv6 packet) follows this encapsulation header. Each header in the header stack contains a header type followed by zero or more header fields. An IPv6 header stack may contain, in the following order, addressing, hop-by-hop options, routing, fragmentation, destination options, and finally payload [RFC2460]. The 6LoWPAN header format is structured the same way. Currently only payload options are defined for the 6LoWPAN header format.

The definition of 6LoWPAN headers consists of the dispatch value, the definition of the header fields that follow, and their ordering constraints relative to all other headers. Although the header stack structure provides a mechanism to address future demands on the 6LoWPAN adaptation layer, it is not intended to provide general purpose extensibility. This document specifies a small set of 6LoWPAN header types using the 6LoWPAN header stack for clarity, compactness, and orthogonality.

4.1. Dispatch Header

The dispatch header is shown below:

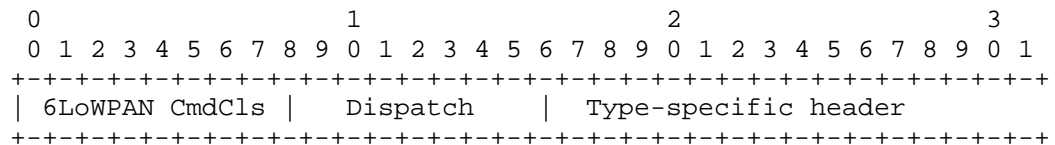


Figure 1: Dispatch Type and Header

6LoWPAN CmdCls: 6LoWPAN Command Class identifier. This field MUST carry the value 0x4F [G.9959]. The value specifies that the

following bits are a 6LoWPAN encapsulated datagram. Non-6LoWPAN protocols MUST ignore the contents following the 6LoWPAN Command Class identifier.

Dispatch: Identifies the header type immediately following the Dispatch Header.

Type-specific header: A header determined by the Dispatch Header.

The dispatch value may be treated as an unstructured namespace. Only a few symbols are required to represent current 6LoWPAN functionality. Although some additional savings could be achieved by encoding additional functionality into the dispatch byte, these measures would tend to constrain the ability to address future alternatives.

Dispatch values used in this specification are compatible with the dispatch values defined by [RFC4944] and [RFC6282].

Pattern	Header Type	Reference
01 000001	IPv6 - Uncompressed IPv6 Addresses	[RFC4944]
01 1xxxxx	6LoWPAN_IPHC - 6LoWPAN_IPHC compressed IPv6	[RFC6282]

All other Dispatch values are unassigned in this document.

Figure 2: Dispatch values

IPv6: Specifies that the following header is an uncompressed IPv6 header.

6LoWPAN_IPHC: IPv6 Header Compression. Refer to [RFC6282].

5. LoWPAN addressing

IPv6 addresses are autoconfigured from IIDs which are again constructed from link-layer address information to save memory in devices and to facilitate efficient IP header compression as per [RFC6282].

A G.9959 NodeID is 8 bits in length. A NodeID is mapped into an IEEE EUI-64 identifier as follows:

`IID = 0000:00ff:fe00:YXXX`

Figure 3: Constructing a compressible IID

where XX carries the G.9959 NodeID and YY is a one byte value chosen by the individual node. The default YY value MUST be zero. A node MAY use other values of YY than zero to form additional IIDs in order to instantiate multiple IPv6 interfaces. The YY value MUST be ignored when computing the corresponding NodeID (the XX value) from an IID.

A 6LoWPAN network typically is used for M2M-style communication. The method of constructing IIDs from the link-layer address obviously does not support addresses assigned or constructed by other means. A node MUST NOT compute the NodeID from the IID if the first 6 bytes of the IID do not comply with the format defined in Figure 3. In that case, the address resolution mechanisms of RFC 6775 apply.

5.1. Stateless Address Autoconfiguration of routable IPv6 addresses

The IID defined above MUST be used whether autoconfiguring a ULA IPv6 address [RFC4193] or a globally routable IPv6 address [RFC3587] in G.9959 subnets.

5.2. IPv6 Link Local Address

The IPv6 link-local address [RFC4291] for a G.9959 interface is formed by appending the IID defined above to the IPv6 link local prefix FE80::/64.

The "Universal/Local" (U/L) bit MUST be set to zero in keeping with the fact that this is not a globally unique value [EUI64].

The resulting link local address is formed as follows:

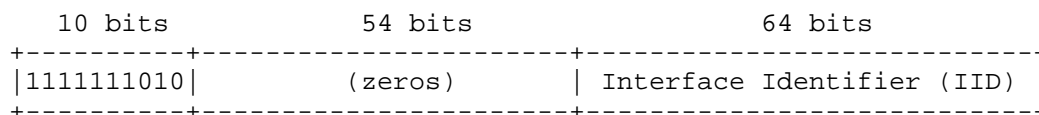


Figure 4: IPv6 Link Local Address

5.3. Unicast Address Mapping

The address resolution procedure for mapping IPv6 unicast addresses into G.9959 link-layer addresses follows the general description in

Section 7.2 of [RFC4861]. The Source/Target Link-layer Address option MUST have the following form when the link layer is G.9959.

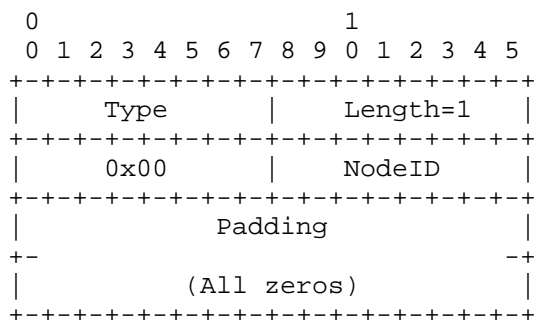


Figure 5: IPv6 Unicast Address Mapping

Option fields:

Type: The value 1 signifies the Source Link-layer address. The value 2 signifies the Destination Link-layer address.

Length: This is the length of this option (including the type and length fields) in units of 8 octets. The value of this field is always 1 for G.9959 NodeIDs.

NodeID: This is the G.9959 NodeID the actual interface currently responds to. The link-layer address may change if the interface joins another network at a later time.

5.4. On the use of Neighbor Discovery technologies

[RFC4861] specifies how IPv6 nodes may resolve link layer addresses from IPv6 addresses via the use of link-local IPv6 multicast. [RFC6775] is an optimization of [RFC4861], specifically targeting 6LoWPAN networks. [RFC6775] defines how a 6LoWPAN node may register IPv6 addresses with an authoritative border router (ABR). Generally, nodes SHOULD NOT use [RFC6775] address registration. However, address registration MUST be used if the first 6 bytes of the IID do not comply with the format defined in Figure 3.

In route-over environments, IPv6 hosts MUST use [RFC6775] address registration. [RFC6775] Duplicate Address Detection (DAD) SHOULD NOT be used, since the link-layer inclusion process of G.9959 ensures that a NodeID is unique for a given HomeID.

5.4.1. Prefix and CID management (Route-over)

A node implementation for route-over operation MAY use RFC6775 mechanisms for obtaining IPv6 prefixes and corresponding header compression context information [RFC6282]. RFC6775 Route-over requirements apply with no modifications.

5.4.2. Prefix and CID management (Mesh-under)

An implementation for mesh-under operation MUST use [RFC6775] mechanisms for managing IPv6 prefixes and corresponding header compression context information [RFC6282]. When using [RFC6775] mechanisms for sending RAs, the M flag MUST NOT be set. As stated by [RFC6775], an ABR is responsible for managing prefix(es). Global prefixes may change over time. It is RECOMMENDED that a ULA prefix is always assigned to the 6LoWPAN subnet to facilitate stable site-local application associations based on IPv6 addresses. Prefixes used in the 6LoWPAN subnet are distributed by normal RA mechanisms. The 6LoWPAN Context Option (6CO) is used according to [RFC6775] in an RA to disseminate Context IDs (CID) to use for compressing prefixes. Prefixes and corresponding Context IDs MUST be assigned during initial node inclusion. Nodes MUST renew the prefix and CID according to the lifetime signaled by the ABR. [RFC6775] specifies that the maximum value of the RA Router Lifetime field MAY be up to 0xFFFF. This document further specifies that the value 0xFFFF MUST be interpreted as infinite lifetime. This value SHOULD NOT be used by ABRs. Its use is only intended for a sleeping network controller; for instance a battery powered remote control being master for a small island-mode network of light modules. CIDs SHOULD be used in a cyclic fashion to assist battery powered nodes with no real-time clock. When updating context information, a CID may have its lifetime set to zero to obsolete it. The CID SHOULD NOT be reused immediately; rather the next vacant CID should be assigned. An ABR detecting the use of an obsoleted CID SHOULD immediately send an RA with updated Context Information. Header compression based on CIDs MUST NOT be used for RA messages carrying Context Information. An expired CID and the associated prefix SHOULD NOT be reset but rather retained in receive-only mode if there is no other current need for the CID value. This will allow an ABR to detect if a sleeping node without clock uses an expired CID and in response, the LBR SHOULD immediately return an RA with fresh Context Information to the originator. Except for the specific redefinition of the RA Router Lifetime value 0xFFFF, the above text is in compliance with [RFC6775].

6. Header Compression

IPv6 header fields SHOULD be compressed. If IPv6 header compression is used, it MUST be according to [RFC6282]. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282].

In general the following substitutions should be made:

- o Replace "802.15.4" with "G.9959"
- o Replace "802.15.4 short address" with "<Interface><G.9959 NodeID>"
- o Replace "802.15.4 PAN ID" with "G.9959 HomeID"

When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short address") it MUST be formed by prepending an Interface label byte to the G.9959 NodeID:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|   Interface   |   NodeID   |
+---+---+---+---+---+---+---+---+

```

A transmitting node may be sending to an IPv6 destination address which can be reconstructed from the link-layer destination address. If the Interface number is zero (the default value), all IPv6 address bytes may be elided. Likewise, the Interface number of a fully elided IPv6 address (i.e. SAM/DAM=11) may be reconstructed to the value zero by a receiving node.

64 bit 802.15.4 address details MUST be ignored. This document only specifies the use of short addresses.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

The method of derivation of Interface Identifiers from 8-bit NodeIDs preserves uniqueness within the logical network. However, there is no protection from duplication through forgery. Neighbor Discovery in G.9959 links may be susceptible to threats as detailed in [RFC3756]. G.9959 networks may feature mesh routing. This implies

additional threats due to ad hoc routing as per [KW03]. G.9959 provides capability for link-layer security. G.9959 nodes MUST use link-layer security with a shared key. Doing so will alleviate the majority of threats stated above. A sizeable portion of G.9959 devices is expected to always communicate within their PAN (i.e., within their subnet, in IPv6 terms). In response to cost and power consumption considerations, these devices will typically implement the minimum set of features necessary. Accordingly, security for such devices may rely on the mechanisms defined at the link layer by G.9959. G.9959 relies on the Advanced Encryption Standard (AES) for authentication and encryption of G.9959 frames and further employs challenge-response handshaking to prevent replay attacks.

It is also expected that some G.9959 devices (e.g. billing and/or safety critical products) will implement coordination or integration functions. These may communicate regularly with IPv6 peers outside the subnet. Such IPv6 devices are expected to secure their end-to-end communications with standard security mechanisms (e.g., IPsec, TLS, etc).

9. Acknowledgements

Thanks to the authors of RFC 4944 and RFC 6282 and members of the IETF 6LoWPAN working group; this document borrows extensively from their work. Thanks to Kerry Lynn, Tommas Jess Christensen and Erez Ben-Tovim for useful discussions. Thanks to Carsten Bormann for extensive feedback which improved this document significantly.

10. References

10.1. Normative References

- [EUI64] IEEE, "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", IEEE Std <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>, November 2012.
- [G.9959.llc] ITU-T, "G.9959 Contribution: Logical Link Control (LLC) layer", ITU-T draft contribution 2013-04-Q15-023.doc, April 2013.
- [G.9959.sar] ITU-T, "G.9959 Contribution: Segmentation And Reassembly (SAR) adaptation layer", ITU-T draft contribution 2013-04-Q15-024.doc, April 2013.

- [G.9959] ITU-T, "G.9959: Low-Power, narrowband radio for control applications", January 2012.
- [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.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, August 2003.
- [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.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [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.

10.2. Informative References

- [P2P-RPL] Goyal, M., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "IETF, I-D.ietf-roll-p2p-rpl-15, Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks", December 2012.

- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [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.

Authors' Addresses

Anders Brandt
Sigma Designs
Emdrupvej 26A, 1.
Copenhagen O 2100
Denmark

Email: anders_brandt@sigmadesigns.com

Jakob Buron
Sigma Designs
Emdrupvej 26A, 1.
Copenhagen O 2100
Denmark

Email: jakob_buron@sigmadesigns.com

6man
Internet-Draft
Updates: 2460, 2780 (if approved)
Intended status: Standards Track
Expires: August 26, 2013

B. Carpenter
Univ. of Auckland
S. Jiang
Huawei Technologies Co., Ltd
February 22, 2013

Transmission of IPv6 Extension Headers
draft-carpenter-6man-ext-transmit-02

Abstract

Various IPv6 extension headers have been defined since the IPv6 standard was first published. This document updates RFC 2460 to clarify how intermediate nodes should deal with such extension headers and with any that are defined in future. It also specifies how extension headers should be registered by IANA, with a corresponding minor update to RFC 2780.

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 26, 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 and Problem Statement	3
1.1. Terminology	5
2. Requirement to Transmit Extension Headers	5
2.1. All Extension Headers	5
2.2. Hop-by-Hop Options	6
3. Security Considerations	6
4. IANA Considerations	6
5. Acknowledgements	7
6. Change log [RFC Editor: Please remove]	7
7. References	7
7.1. Normative References	7
7.2. Informative References	8
Authors' Addresses	9

1. Introduction and Problem Statement

In IPv6, an extension header is any header that follows the initial 40 bytes of the packet and precedes the upper layer header (which might be a transport header, an ICMPv6 header, or a notional "No Next Header").

An initial set of IPv6 extension headers was defined by [RFC2460], which also described how they should be handled by intermediate nodes, with the exception of the hop-by-hop options header:

"...extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header."

This provision allowed for the addition of new extension headers, since it means that forwarding nodes should be completely transparent to them. Thus, new extension headers could be introduced progressively, used only by hosts that have been updated to create and interpret them. The extension header mechanism is an important part of the IPv6 architecture, and several new extension headers have been defined since RFC 2460.

Unfortunately, experience has showed that the network is not transparent to these headers. The main reason for this is that by design, some firewalls attempt to inspect the transport header or payload. This means that they need to traverse the chain of extension headers, if present, until they find the transport header (or an encrypted payload). Unfortunately, because not all IPv6 extension headers follow a uniform TLV format, this process is clumsy and requires knowledge of each extension header's format.

The process is potentially slow as well as clumsy, possibly precluding its use in nodes attempting to process packets at line speed. The present document does not intend to solve this problem, which is caused by the fundamental architecture of IPv6 extension headers. This document focuses on clarifying how the header chain should be traversed in the current IPv6 architecture.

If they encounter an unrecognised extension header type, some firewalls treat the packet as suspect and drop it. Unfortunately, it is an established fact that several widely used firewalls do not recognise some or all of the extension headers defined since RFC 2460. It has also been observed that certain firewalls do not even handle all the extension headers in RFC 2460, including the fragment header [I-D.taylor-v6ops-fragdrop], causing fundamental problems of connectivity. This applies in particular to firewalls that attempt

to inspect packets statelessly at very high speed, since they cannot take the time to reassemble fragmented packets, especially when under a denial of service attack.

Other types of middlebox, such as load balancers or packet classifiers, might also fail in the presence of extension headers that they do not recognise.

A contributory factor to this problem is that, because extension headers are numbered out of the existing IP Protocol Number space, there is no collected list of them. For this reason, it is hard for an implementor to quickly identify the full set of defined extension headers. An implementor who consults only RFC 2460 will miss all extension headers defined subsequently.

This combination of circumstances creates a "Catch-22" situation [Heller] for the deployment of any newly designed extension header. It cannot be widely deployed, because existing firewalls will render large parts of the Internet opaque to it. However, most firewalls will not be updated to allow the new header to pass until it has been proved safe and useful on the open Internet, which is impossible until the firewalls have been updated.

The uniform TLV format now defined for extension headers [RFC6564] will improve the situation, but only for future extensions. Some tricky and potentially malicious cases will be avoided by forbidding very long chains of extension headers that need to be fragmented [I-D.ietf-6man-oversized-header-chain]. This will alleviate concerns that stateless firewalls cannot handle a complete header chain as required by the present document.

However, these changes are insufficient to correct the underlying problem. The present document clarifies that the above requirement from RFC 2460 applies to all types of node that forward IPv6 packets and to all extension headers defined now and in the future. It also requests IANA to create a subsidiary registry that clearly identifies extension header types, and updates RFC 2780 accordingly. Fundamental changes to the IPv6 extension header architecture are out of scope for this document.

Also, Hop-by-Hop options are not handled by many high speed routers, or are processed only on a slow path. This document also updates the requirements for processing the Hop-by-Hop options header to make them more realistic.

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 [RFC2119].

2. Requirement to Transmit Extension Headers

2.1. All Extension Headers

Any node along an IPv6 packet's path, which forwards it for any reason, SHOULD do so regardless of any extension headers that are present, as described in RFC 2460. Exceptionally, if this node is designed to examine extension headers for any reason, such as firewalling, it MUST recognise and deal appropriately with all defined IPv6 extension header types. The list of currently defined extension header types is maintained by IANA (see Section 4) and implementors are advised to check this list regularly for updates.

RFC 2460 requires destination hosts to discard packets containing unrecognised extension headers. However, intermediate forwarding nodes MUST NOT do this by default, since that might cause them to inadvertently discard traffic using a recently defined extension header, not yet recognised by the intermediate node.

As mentioned above, firewalls that discard packets containing extension headers are known to cause connectivity failures and deployment problems. Therefore, it is important that firewalls can parse all defined IPv6 extension headers and are able to behave according to the above requirements. If a firewall discards a packet containing a defined IPv6 extension header, it MUST be the result of a configurable firewall policy, and not just the result of a failure to recognise such a header. This means that the discard policy for each defined type of extension header MUST be individually configurable. The default configuration SHOULD allow all defined extension headers. Firewalls MUST be configurable to allow packets containing unrecognised extension headers, but such packets MUST be dropped by default.

The IPv6 Routing Header Types 0 and 1 have been deprecated and SHOULD NOT be used. However, as specified in [RFC5095], this does not mean that the IPv6 Routing Header can be unconditionally dropped by forwarding nodes. Packets containing undeprecated Routing Headers SHOULD be forwarded by default. At the time of writing, these include Type 2 [RFC6275], Type 3 [RFC6554], and Types 253 and 254 [RFC4727]. Others may be defined in future.

2.2. Hop-by-Hop Options

The IPv6 Hop-by-Hop Options header SHOULD be processed by intermediate nodes as described in [RFC2460]. However, it is to be expected that high performance routers will either ignore it, or assign packets containing it to a slow processing path. Designers planning to use a Hop-by-Hop option need to be aware of this likely behaviour.

As a reminder, in RFC 2460, it is stated that the Hop-by-Hop Options header, if present, must be first.

3. Security Considerations

Firewall devices MUST conform to the requirements in the previous section in order to respect the IPv6 extension header architecture. In particular, packets containing specific extension headers are only to be discarded as a result of a configurable policy.

When new extension headers are defined in the future, those implementing and configuring firewalls will need to take account of them. It is to be expected that this process will be slow. Until it is complete, the new extension will fail in some parts of the Internet. This aspect needs to be considered when deciding to standardise a new extension.

4. IANA Considerations

IANA is requested to clearly mark in the Assigned Internet Protocol Numbers registry those values which are also IPv6 Extension Header types, for example by adding an extra column to indicate this. This will also apply to any IPv6 Extension Header types defined in the future.

Additionally, IANA is requested to replace the existing empty IPv6 Next Header Types registry by an IPv6 Extension Header Types registry. It will contain only those protocol numbers which are also marked as IPv6 Extension Header types in the Assigned Internet Protocol Numbers registry. The initial list will be as follows:

- o 0, Hop-by-Hop Options, [RFC2460]
- o 43, Routing, [RFC2460], [RFC5095]
- o 44, Fragment, [RFC2460]
- o 50, Encapsulating Security Payload, [RFC4303]
- o 51, Authentication, [RFC4302]

- o 60, Destination Options, [RFC2460]
- o 135, MIPv6, [RFC6275]
- o 139, HIP, [RFC5201]
- o 140, shim6, [RFC5533]

The references to the IPv6 Next Header field in [RFC2780] are to be interpreted as also applying to the IPv6 Extension Header field.

5. Acknowledgements

This document was triggered by mailing list discussions including John Leslie, Stefan Marksteiner and others. Valuable comments and contributions were made by Dominique Barthel, Lorenzo Colitti, Fernando Gont, Bob Hinden, Ray Hunter, Suresh Krishnan, Marc Lampo, Michael Richardson, Dave Thaler, Joe Touch, and others.

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

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

6. Change log [RFC Editor: Please remove]

draft-carpenter-6man-ext-transmission-02: clarifications following WG comments, recalibrated firewall requirements, 2013-02-22.

draft-carpenter-6man-ext-transmission-01: feedback at IETF85: clarify scope and impact on firewalls, discuss line-speed processing and lack of uniform TLV format, added references, restructured IANA considerations, 2012-11-13.

draft-carpenter-6man-ext-transmission-00: original version, 2012-08-14.

7. References

7.1. Normative References

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

- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302,
December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)",
RFC 4303, December 2005.
- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4,
ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation
of Type 0 Routing Headers in IPv6", RFC 5095,
December 2007.
- [RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson,
"Host Identity Protocol", RFC 5201, April 2008.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming
Shim Protocol for IPv6", RFC 5533, June 2009.
- [RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support
in IPv6", RFC 6275, July 2011.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and
M. Bhatia, "A Uniform Format for IPv6 Extension Headers",
RFC 6564, April 2012.

7.2. Informative References

- [Heller] Heller, J., "Catch-22", Simon and Schuster , 1961.
- [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.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-00 (work in
progress), October 2012.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.

[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.

Authors' Addresses

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

Email: brian.e.carpenter@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

6MAN

Internet-Draft

Updates: 4291 (if approved)

Intended status: Standards Track

Expires: August 25, 2013

B. Carpenter

Univ. of Auckland

S. Jiang

Huawei Technologies Co., Ltd

February 21, 2013

The U and G bits in IPv6 Interface Identifiers
draft-carpenter-6man-ug-01

Abstract

The IPv6 addressing architecture defines a method by which the Universal and Group bits of an IEEE link-layer address are mapped into an IPv6 unicast interface identifier. This document clarifies the status of those bits for interface identifiers that are not derived from an IEEE link-layer address, and updates RFC 4291 accordingly.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 25, 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
1.1. Terminology	3
2. Problem statement	3
3. Usefulness of the U and G Bits	5
4. Clarification of Specifications	6
5. Security Considerations	6
6. IANA Considerations	7
7. Acknowledgements	7
8. Change log [RFC Editor: Please remove]	7
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Authors' Addresses	9

1. Introduction

According to the IPv6 addressing architecture [RFC4291], when a 64-bit IPv6 unicast Interface Identifier (IID) is formed on the basis of an IEEE EUI-64 address, usually itself expanded from a 48-bit MAC address, a particular format must be used:

"For all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format."

The specification assumes that that the normal case is to transform an Ethernet-style address into an IID, preserving the information provided by two bits in particular:

- o The "u" bit in an IEEE address is set to 0 to indicate universal scope (implying uniqueness) or to 1 to indicate local scope (without implying uniqueness). In an IID this bit is inverted, i.e., 1 for universal scope and 0 for local scope. According to [RFC5342], the reason for this was "to make it easier for network operators to type in local-scope identifiers".
- o The "g" bit in an IEEE address is set to 1 to indicate group addressing (link-layer multicast). The value of this bit is preserved in an IID.

This document discusses problems observed with the "u" and "g" bits as a result of the above requirements. It then discusses the usefulness of the two bits, and updates RFC 4291 accordingly.

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 [RFC2119].

2. Problem statement

In addition to IIDs formed from IEEE EUI-64 addresses, various new forms of IID have been defined or proposed, such as temporary addresses [RFC4941], Cryptographically Generated Addresses (CGAs) [RFC3972], Hash-Based Addresses (HBAs) [RFC5535], stable privacy addresses [I-D.ietf-6man-stable-privacy-addresses], or mapped addresses for 4rd [I-D.ietf-softwire-4rd]. In each case, the question of how to set the "u" and "g" bits has to be decided. For example, RFC 3972 specifies that they are both zero in CGAs, and the same applies to HBAs. On the other hand, RFC 4941 specifies that "u" must be zero but leaves "g" variable.

Another case where the "u" and "g" bits are specified is in the Reserved IPv6 Subnet Anycast Address format [RFC2526], which states that "for interface identifiers in EUI-64 format, the universal/local bit in the interface identifier MUST be set to 0" (i.e., local) and requires that "g" bit to be set to 1. However, the text neither states nor implies any semantics for these bits in anycast addresses.

These cases illustrate that the statement quoted above from RFC 4291 requiring "Modified EUI-64 format" is rather meaningless when applied to forms of IID that are not in fact based on an underlying EUI-64 address. In practice, the IETF has chosen to assign some 64-bit IIDs that have nothing to do with EUI-64.

A particular case is that of /127 prefixes for point-to-point links between routers, as standardised by [RFC6164]. The addresses on these links are undoubtedly global unicast addresses, but they do not have a 64-bit IID. The bits in the positions named "u" and "g" in such an IID have no special significance and their values are not specified.

Each time a new IID format is proposed, the question arises whether these bits have any meaning. Section 2.2.1 of RFC 5342 discusses the mechanics of the bit allocations but does not explain the purpose or usefulness of these bits in an IID. There is an IANA registry for reserved IID values [RFC5453] but again there is no explanation of the purpose of the "u" and "g" bits.

There was a presumption when IPv6 was designed and the IID format was first specified that a universally unique IID might prove to be very useful, for example to contribute to solving the multihoming problem. Indeed, the addressing architecture [RFC4291] states this explicitly:

"The use of the universal/local bit in the Modified EUI-64 format identifier is to allow development of future technology that can take advantage of interface identifiers with universal scope."

However, this has not so far proved to be the case. Also, there is evidence from the field that IEEE MAC addresses with "u" = 0 are sometime incorrectly assigned to multiple MAC interfaces. Firstly, there are recurrent reports of manufacturers assigning the same MAC address to multiple devices. Secondly, significant re-use of the same virtual MAC address is reported in virtual machine environments. Once transformed into IID format (with "u" = 1) these identifiers would purport to be universally unique but would in fact be ambiguous. This has no known harmful effect as long as the replicated MAC addresses and IIDs are used on different layer 2 links. If they are used on the same link, of course there will be a problem, to be detected by duplicate address detection [RFC4862], but

such a problem can usually only be resolved by human intervention.

The conclusion from this is that the "u" bit is not a reliable indicator of universal uniqueness.

We note that Identifier-Locator Network Protocol (ILNP), a multihoming solution that might be expected to benefit from universally unique IIDs in modified EUI-64 format, does not in fact rely on them. ILNP uses its own format, defined as a Node Identifier [RFC6741]. ILNP does have the constraint that Node Identifiers must be unique within a given site, but as we have just shown, the state of the "u" bit does not in any way guarantee this.

Thus, we can conclude that the value of the "u" bit in IIDs has no particular meaning. In the case of an IID created from a MAC address according to RFC 4291, its value is determined by the MAC address, but that is all.

An IPv6 IID should not be created from a MAC group address, so the "g" bit will normally be zero, but this value also has no particular meaning. Additionally, the "u" and the "g" bits are both meaningless in the format of an IPv6 multicast group ID [RFC3306], [RFC3307].

None of the above implies that there is a problem with using the "u" and "g" bits in MAC addresses as part of the process of generating IIDs from MAC addresses, or with specifying their values in other methods of generating IIDs. What it does imply is that, after an IID is generated by any method, no reliable deductions can be made from the state of the "u" and "g" bits; in other words, these bits have no useful semantics in an IID.

Once this is recognised, we can avoid the problematic confusion caused by these bits each time that a new form of IID is proposed.

3. Usefulness of the U and G Bits

Given that the "u" and "g" bits do not have a reliable meaning in an IID, it is relevant to consider what usefulness they do have.

If an IID is known or guessed to have been created according to RFC 4291, it could be transformed back into a MAC address. This can be very helpful during operational fault diagnosis. For that reason, mapping the IEEE "u" and "g" bits into the IID has operational usefulness. However, it should be stressed that "u" = "g" = 0 does not prove that an IID was formed from a MAC address; on the contrary, it might equally result from another method. With other methods, there is no reverse transformation available.

To the extent that each method of IID creation specifies the values of the "u" and "g" bits, and that each new method is carefully designed in the light of its predecessors, these bits tend to reduce the chances of duplicate IIDs.

4. Clarification of Specifications

This section describes clarifications to the IPv6 specifications that result from the above discussion. Their aim is to reduce confusion while retaining the useful aspects of the "u" and "g" bits in IIDs.

The EUI-64 to IID transformation defined in the IPv6 addressing architecture [RFC4291] MUST be used for all cases where an IPv6 IID is derived from an IEEE MAC or EUI-64 address. With any other form of link layer address, an equivalent transformation SHOULD be used. However, the resulting "u" and "g" bits in an IID have no semantics; in other words, they have opaque values. In fact, the whole IID should be viewed as opaque by third parties.

Specifications of other forms of 64-bit IID will either specify explicitly how the "u" and "g" bits are set, or will simply include them as part of a field within the IID. In either case, a semantic meaning for these bits MUST NOT be defined.

In the following statement in section 2.5.1 of the IPv6 addressing architecture [RFC4291], the reference to "Modified EUI-64 format" applies only to cases where there is an underlying IEEE address:

"For all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format."

The following statement in section 2.5.1 of the IPv6 addressing architecture [RFC4291] is hereby obsoleted:

"The use of the universal/local bit in the Modified EUI-64 format identifier is to allow development of future technology that can take advantage of interface identifiers with universal scope."

As far as is known, no existing implementation will be affected by these changes. The benefit is that future design discussions are simplified.

5. Security Considerations

No new security exposures or issues are raised by this document.

6. IANA Considerations

This document requests no immediate action by IANA. However, the following should be noted when considering future proposed additions to the registry of reserved IID values, which requires Standards Action according to [RFC5453]. A reserved IID, or a range of reserved IIDs, will most likely specify values for both "u" and "g", since they are among the high-order bits. At the present time, none of the known methods of generating IIDs will generate "u" = "g" = 1. Reserved IIDs with "u" = "g" = 1 are therefore unlikely to collide with automatically generated IIDs.

7. Acknowledgements

Valuable comments were received from Remi Despres, Fernando Gont, Brian Haberman, Joel Halpern, Ray Hunter, Mark Smith, and other participants in the 6MAN working group.

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

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

8. Change log [RFC Editor: Please remove]

draft-carpenter-6man-ug-01: numerous clarifications following WG comments, discussed DAD, added new section on the usefulness of the u/g bits, expanded IANA considerations, set intended status, 2013-02-21.

draft-carpenter-6man-ug-00: original version, 2013-01-31.

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. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC5342] Eastlake, D., "IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters", BCP 141, RFC 5342, September 2008.

- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, February 2009.

9.2. Informative References

- [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-03 (work in progress), January 2013.
- [I-D.ietf-softwire-4rd]
Jiang, S., Despres, R., Penno, R., Lee, Y., Chen, G., and M. Chen, "IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd)", draft-ietf-softwire-4rd-04 (work in progress), October 2012.
- [RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, March 1999.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.
- [RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, August 2002.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [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.
- [RFC5535] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, June 2009.
- [RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, April 2011.
- [RFC6741] Atkinson, R.J., "Identifier-Locator Network Protocol (ILNP) Engineering Considerations", RFC 6741,

November 2012.

Authors' Addresses

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

Email: brian.e.carpenter@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 2460, 4443 (if approved)
Intended status: Standards Track
Expires: September 22, 2013

F. Gont
SI6 Networks / UTN-FRH
W. Liu
Huawei Technologies
March 21, 2013

Security Implications of IPv6 Options of Type 10xxxxxx
draft-gont-6man-ipv6-smurf-amplifier-03

Abstract

When an IPv6 node processing an IPv6 packet does not support an IPv6 option whose two-highest-order bits of the Option Type are '10', it is required to respond with an ICMPv6 Parameter Problem error message, even if the Destination Address of the packet was a multicast address. This feature provides an amplification vector, opening the door to an IPv6 version of the 'Smurf' Denial-of-Service (DoS) attack found in IPv4 networks. This document discusses the security implications of the aforementioned options, and formally updates RFC 2460 and RFC 4443 such that this attack vector is eliminated. Additionally, it describes a number of operational mitigations that could be deployed against this attack vector.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 22, 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. Updating RFC 2460 and RFC 4443	5
3. Operational mitigations	7
4. IANA Considerations	8
5. Security Considerations	9
6. Acknowledgements	10
7. References	11
7.1. Normative References	11
7.2. Informative References	11
Authors' Addresses	12

1. Introduction

IPv6 has eliminated most of the amplification vectors that were available in IPv4 to perform 'Smurf'-like Denial of Service (DoS) attacks [CERT1998]. However, an amplification vector has been left in the core IPv6 and ICMPv6 specifications ([RFC2460] and [RFC4443]) that would allow for an IPv6 version of the 'Smurf' Denial-of-Service (DoS) attacks [CERT1998] [RFC6274] found in IPv4 networks. The aforementioned vector is based on the use of unsupported IPv6 options, used in combination with multicast destinations.

[RFC2460] specifies, in Section 4.2, that when a node processing an IPv6 packet does not support an IPv6 option whose two-highest-order bits of the Option Type are '10', it should respond with an ICMPv6 Parameter Problem error message, even if the Destination Address of the packet was a multicast address. [RFC4443] specifies, in Section 2.4 (page 6), that packets destined to an IPv6 multicast address should not elicit ICMPv6 error messages, with the exception of ICMPv6 Packet Too Big messages (such that Path-MTU Discovery works for IPv6 multicast) and the Parameter Problem Message, Code 2 for reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10.

This feature provides an amplification vector, opening the door to an IPv6 version of the 'Smurf' Denial-of-Service (DoS) attack [CERT1998] [RFC6274] found in IPv4 networks.

An attacker could exploit the aforementioned amplification vector by sending forged IPv6 packets with the IPv6 address of the victim system as the Source Address of his packets, a multicast address as the Destination Address, and an unsupported option (with an Option Type of '10xxxxxx') in a Destination Options Header. Upon receipt of the forged packet, each receiving host would respond with an ICMPv6 Parameter Problem, code 2, error message, pointing to the unsupported option type. Thus, the systems belonging to the multicast group specified by the multicast address contained in the Destination Address field would serve as an 'amplifier network'.

It should be noted that if the multicast RPF check is used (e.g. to prevent routing loops), this would prevent an attacker from forging the Source Address of a packet to an arbitrary value, thus preventing an attacker from launching this attack against a remote network.

Chapter 5 of [Juniper2010] discusses multicast RPF configuration for Juniper routers.

Section 2 updates RFC 2460 [RFC2460] and RFC 4443 [RFC4443], such

that the aforementioned attack vector is eliminated. Section 3 describes a number of operational mitigations for the aforementioned attack vector.

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

2. Updating RFC 2460 and RFC 4443

Considering the security implications discussed in Section 1, and since there are no known legitimate uses of IPv6 options of type '10xxxxxx', this document updates the corresponding specifications to eliminate these issues.

The following text in Section 4.2 (page 9) of [RFC2460]:

- 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.

is replaced with:

- 10 - discard the packet and send an ICMP Parameter Problem, Code 2, message to the packet's Source Address (pointing to the unrecognized Option Type), only if (1) the packet's Destination Address was not a multicast address, or (2) the packet's Destination Address was a multicast address, but the node sending the Parameter Problem error message can assert that the Source Address of the packet eliciting the error message has not been forged.

Additionally, the following text in Section 2.4 (page 6) of [RFC4443]:

- (e.3) A packet destined to an IPv6 multicast address. (There are two exceptions to this rule: (1) the Packet Too Big Message (Section 3.2) to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 (Section 3.4) reporting an unrecognized IPv6 option (see Section 4.2 of [IPv6]) that has the Option Type highest-order two bits set to 10).

is replaced with:

- (e.3) A packet destined to an IPv6 multicast address. (There is one exception to this rule: the Packet Too Big Message (Section 3.2) to allow Path MTU discovery to work for IPv6 multicast).
- (e.3) A packet destined to an IPv6 multicast address. (There are two exceptions to this rule: (1) the Packet Too Big Message (Section 3.2) to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 (Section 3.4) reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10, *provided* the node sending the Parameter Problem message can assert that the Source Address of the packet eliciting the error message has not been forged.).

3. Operational mitigations

This section describes a number of operational mitigations that could be implemented for the aforementioned attack vector:

- o Firstly, IPv6 nodes should limit their ICMPv6 traffic. This is a general mitigation technique for any bandwidth-exhaustion attack that relies on ICMPv6 traffic. This could be enforced at the hosts themselves, or at any router connecting such hosts to the public network.
- o Secondly, as noted in Section 1 of this document, the multicast RPF check could be enabled such that an attacker cannot forge the Source Address of a packet to an arbitrary value, thus preventing an attacker from launching this attack against a remote network.

4. 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.

5. Security Considerations

This document describes how IPv6 options whose two-highest-order bits of the Option Type are '10' could be exploited to perform an IPv6 version of the 'Smurf' Denial-of-Service (DoS) attack [CERT1998] [RFC6274] found in IPv4 networks. It formally updates RFC 2460 [RFC2460] such that this attack vector is eliminated, and also describes a number of operational mitigations that could be deployed against this attack vector.

6. Acknowledgements

The authors would like to thank (in alphabetical order) Francis Dupont, Joel Halpern, Suresh Krishnan, Simon Perreault, Dave Thaler, and Ole Troan, for providing valuable comments on earlier versions of this document.

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

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

7. References

7.1. Normative References

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

7.2. Informative References

- [RFC6274] Gont, F., "Security Assessment of the Internet Protocol Version 4", RFC 6274, July 2011.
- [CPNI-IPv6] Gont, F., "Security Assessment of the Internet Protocol version 6 (IPv6)", UK Centre for the Protection of National Infrastructure, (available on request).
- [CERT1998] CERT, "CERT Advisory CA-1998-01: Smurf IP Denial-of-Service Attacks", 1998, <<http://www.cert.org/advisories/CA-1998-01.html>>.
- [Juniper2010] Juniper, "JunosE Software for E Series Broadband Services Routers Multicast Routing Configuration Guide", 2010, <http://www.juniper.net/techpubs/en_US/junosell.2/information-products/topic-collections/swconfig-multicast-routing/book-swconfig-multicast.pdf>.

Authors' Addresses

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

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

Will (Shucheng) Liu
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: liushucheng@huawei.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(Prefix, Net_Iface, Network_ID, DAD_Counter, 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-IID]
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>

Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: March 21, 2014

B. Liu
Huawei Technologies
R. Bonica
Juniper Networks
September 16, 2013

DHCPv6/SLAAC Address Configuration Interaction Problem Statement
draft-liu-bonica-dhcpv6-slaac-problem-02.txt

Status of this Memo

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

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

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

This Internet-Draft will expire on March 21, 2014.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Abstract

This document analyzes the host behavior of DHCPv6/SLAAC interaction issue. It reviews the standard definition of the host behaviors and

provides the test results of current mainstream implementations. Some potential operational gaps of the interaction are also described.

Table of Contents

1. Introduction	3
2. Host Behavior of DHCPv6/SLAAC Interaction	3
2.1. Relevant RA Flags Defined in Standards	4
2.1.1. A (Autonomous) Flag	4
2.1.2. M (Managed) Flag	4
2.1.3. O (Otherconfig) Flag	4
2.2. Behaviors of Current Implementations	5
2.2.1. A flag	5
2.2.2. M flag	5
2.2.3. O flag	6
3. Possible Operational Gaps of DHCPv6/SLAAC Interaction	6
3.1. Renumbering	6
3.2. Cold Start Problems	7
3.3. Strong Management	7
4. Conclusions	7
5. Security Considerations	7
6. IANA Considerations	7
7. References	8
7.1. Normative References	8
7.2. Informative References	8
8. Acknowledgments	8
Appendix A. Test Details of Host Behaviors	10
A.1 Host Initialing Behavior	10
A.2 Host SLAAC/DHCPv6 Switching Behavior	11
A.3 Host Stateful/Stateless DHCPv6 Behavior	12
Authors' Addresses	13

1. Introduction

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

In ND protocol, there is a M (ManagedFlag) flag defined in RA message, indicating the hosts there is DHCPv6 service in the network if the flag is set. And there is an O "OtherConfigFlag", if set, indicating configure information other than addresses (e.g. DNS, Route .etc) is available through DHCPv6 configuration. Moreover, there's another A (Autonomous) flag defined in ND, which indicating the hosts to do SLAAC, may also influent the behavior of hosts.

So with the A/M/O flags, the two separated address configuration modes are somehow correlated. But for some reason, the ND protocol didn't define the flags as prescriptive but only advisory. This ambiguous definition may vary the behavior of hosts when interpreting the flags. In section 2, we provided a brief test result to identify different host operating systems have taken different approaches. This would add additional complexity for both the hosts and the network management.

This draft reviews the standard definition of the above mentioned flags, and provides a test result of several major desktop operating systems' behavior. And then identifies potential requirement/gaps of DHCPv6/SLAAC interaction.

2. Host Behavior of DHCPv6/SLAAC Interaction

In this section, we analyzed A/M/O flags definition, and provide the test result of host behavior of interpreting these flags in mainstream operating systems implementations.

Please note that, A flag has no direct relationship with DHCPv6, but it is somewhat correlated with M/O flags.

2.1. Relevant RA Flags Defined in Standards

2.1.1. A (Autonomous) Flag

In ND Prefix Information Option, the autonomous address-configuration flag (A flag). When set indicates that this prefix can be used for stateless address configuration as specified in SLAAC.

For the host behavior, there is an explicit rule in the SLAAC specification [RFC4862]: "If the Autonomous flag is not set, silently ignore the Prefix Information option."

But when A flag is set, the SLAAC protocol didn't provide a prescriptive definition.

2.1.2. M (Managed) Flag

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

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

But in the updated SLAAC specification [RFC4862], the relative description was removed, the reason was "considering the maturity of implementations and operational experiences. ManagedFlag and OtherConfigFlag were removed accordingly. (Note that this change does not mean the use of these flags is deprecated.)"

2.1.3. O (Otherconfig) Flag

As mentioned above, the situation of O flag is similar with M. In earlier SLAAC [RFC2462], the host behavior is clear:

"If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting

information (excluding addresses if ManagedFlag is set to FALSE). If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement."

And there's another description of the relationship of M and O flags in [RFC2462]:

"In addition, when the value of the ManagedFlag is TRUE, the value of OtherConfigFlag is implicitly TRUE as well. It is not a valid configuration for a host to use stateful address autoconfiguration to request addresses only, without also accepting other configuration information."

2.2. Behaviors of Current Implementations

We did tests of current 3 mainstream desktop operating systems on the behaviors; please refer to the appendix for details. This section illustrates the important results of the tests.

2.2.1. A flag

A flag is a switch to control whether to do SLAAC, and it is independent with M/O flags, in another word, A is independent with DHCPv6.

At the non-SLAAC-config state (either non-configured or DHCPv6-configured only), the 3 Oses acted the same with A flag, if A set, they all configured SLAAC, it is obvious and reasonable. But when SLAAC-configured, and A changed from 1 to 0, the behaviors varied, some deprecated SLAAC while some ignored the RA messages.

2.2.2. M flag

M is a key flag to interact ND/DHCPv6, but the host behaviors on M flag were quite different.

In our test, one OS treats the flag as instruction, it even released DHCPv6 session when M=0. But the other two just treat the flag as advisory, when SLAAC was done, it won't care about M=1, and M=0 won't cause operation for the already configured DHCPv6 addresses. Moreover, the two Oses even would not initiate DHCPv6 session until they

receives RA messages with M=1, this behavior has an implication that DHCPv6 somehow depends on ND.

Please refer to [I-D.liu-6renum-dhcpv6-slaac-switching] for more details.

2.2.3. O flag

In our tests, when M flag is set, the O flag is implicitly set as well; in another word, the hosts would not initial stateful DHCPv6 and stateless DHCPv6 respectively. This is a reasonable behavior.

But the O flag is not independent from A flag in some Oses. In our test, there are two Oses won't initiate stateless DHCPv6 when A flag is not set, that is to say, it is not applicable to have a "stateless DHCPv6 only" configuration state for some operating systems; it is also not applicable for these two Oses to switch between stateful DHCPv6 and stateless DHCPv6 (according to O flag changing from 0 to 1 or verse vice).

3. Possible Operational Gaps of DHCPv6/SLAAC Interaction

According to the abovementioned tests, there are possible operational issues as the following.

3.1. Renumbering

During IPv6 renumbering, the SLAAC-configured hosts can reconfigure IP addresses by receiving ND Router Advertisement (RA) messages containing new prefix information. The DHCPv6-configured hosts can reconfigure addresses by initialing RENEW sessions when the current addresses' lease time is expired or receiving the reconfiguration messages initialed by the DHCPv6 servers.

The above mechanisms have an implicit assumption that SLAAC-configured hosts will remain SLAAC while DHCPv6-managed hosts will remain DHCPv6-managed. But in some situations, SLAAC-configured hosts may need to switch to DHCPv6-managed, or verse vice. In [I-D.ietf-6renum-enterprise], it described several renumbering scenarios in enterprise network for this requirement; for example, the network may split, merge, relocate or reorganize. But due to current implementations, this requirement is not applicable and has been identified as a gap in [I-D.ietf-6renum-gap-analysis].

3.2. Cold Start Problems

If all nodes, or many nodes, restart at the same time after a power cut, the results might not be consistent.

3.3. Strong Management

Since the host behavior of address configuration is somehow uncontrolled by the network side, it might cause gaps to the networks that need strong management (for example, the enterprise networks and the ISP CPE networks). Examples are:

- the network wants the hosts to do DHCPv6-only configuration, it is not applicable for some operating systems due to current implementation unless manually configure the hosts to DHCPv6-only model
- the hosts have been SLAAC-configured, then the network needs the hosts to do DHCPv6 simultaneously (e.g. for multihoming)
- the network wants the hosts to do stateless DHCPv6-only; for example, the hosts are configured with self-generated addresses (e.g. ULA), and they also need to contact the DHCPv6 server for information configuration

4. Conclusions

- The host behavior of SLAAC/DHCPv6 interaction is ambiguous in standard.
- The implementations have been varied on this issue. In [RFC4862] it is said "Removed the text regarding the M and O flags, considering the maturity of implementations and operational experiences." The description seems not true anymore.
- It is foreseeable that the un-uniformed host behavior can cause operational gaps, e.g. in renumbering and strong management.

5. Security Considerations

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

6. IANA Considerations

None.

7. References

7.1. Normative References

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

7.2. Informative References

- [RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.
- [RFC3315] R. Droms, Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.
- [RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", RFC 5887, May 2010.
- [I-D.ietf-6renum-gap-analysis] Liu, B., and Jiang, S., "IPv6 Site Renumbering Gap Analysis", Working in Progress, March 2012
- [I-D.ietf-6renum-enterprise] Jiang, S., and B. Liu, "IPv6 Enterprise Network Renumbering Scenarios and Guidelines ", Working in Progress, March 2012.

8. Acknowledgments

The test was done by our research partner BNRC-BUPT (Broad Network Research Centre in Beijing University of Posts and Telecommunications). Thanks for the hard efficient work of student Xudong Shi and the tutors Prof. Wendong Wang and Prof. Xiangyang Gong.

Valuable comment was received from Brian E Carpenter to improve the draft.

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

Appendix A. Test Details of Host Behaviors

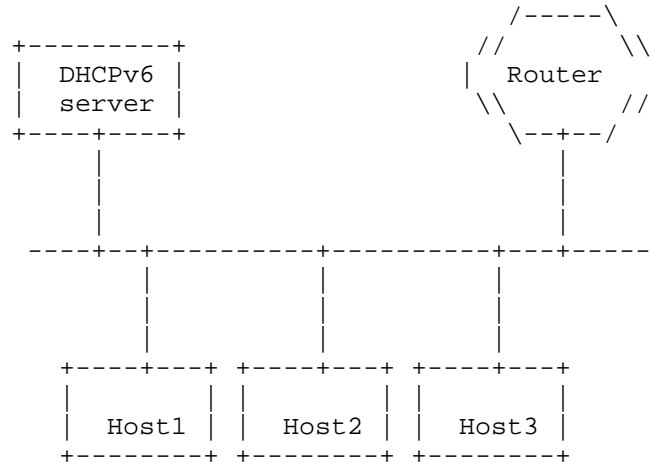


Figure 1 Test Environment

The 5 elements were all created in Vmware in one computer, for ease of operation.

- Router quagga 0.99-19 soft router installed on Ubuntu 11.04 virtual host
- DHCPv6 Server: dibbler-server installed on Ubuntu 11.04 virtual host
- Host A Window 7 Virtual Host
- Host B Ubuntu 12.10 Virtual Host
- Host C Mac OS X v10.7 Virtual Host

A.1 Host Initialing Behavior

Host from non-configured to configured, we tested different A/M/O combinations in each OS platform. The states are enumerated as the following, 3 operation systems respectively:

- o Window 7
 - A=0&M=0&O=0, non-config
 - A=1&M=0&O=0, SLAAC only
 - A=1&M=0&O=1, SLAAC + Stateless DHCPv6
 - A=1&M=1&O=0, SLAAC + DHCPv6

- A=1&M=1&O=1, SLAAC + DHCPv6
 - A=0&M=1&O=0, DHCPv6 only (A=0 or Non-PIO)
 - A=0&M=1&O=1, DHCPv6 only (A=0 or Non-PIO)
 - A=0&M=0&O=1, Stateless DHCPv6 only
- o Linux/MAC OS X
- A=0&M=0&O=0, non-config
 - A=1&M=0&O=0, SLAAC only
 - A=1&M=0&O=1, SLAAC + Stateless DHCPv6
 - A=1&M=1&O=0, SLAAC + DHCPv6
 - A=1&M=1&O=1, SLAAC + DHCPv6
 - A=0&M=1&O=0, DHCPv6 only (A=0 or Non-PIO)
 - A=0&M=1&O=1, DHCPv6 only (A=0 or Non-PIO)
 - A=0&M=0&O=1, non-config

As showed above, Linux and MAC OSX acted the same way, but differated from Windows 7. The only difference is when A=0&M=0&O=1, Windows 7 did stateless DHCPv6 while Linux/MAC OSX did nothing.

Result summary:

- A is interpreted as prescript in each OS at the initial state
- M is interpreted as prescript in each OS at the initial state
- O is interpreted as prescript in Windows 7
- A and M are independent in each OS at the initial state
- A and O are not totally independent in Linux and Mac, A=1 is required for O=1 triggering DHCPv6 info-request
- M and O are not totally independent in each OS. M=1 has the implication O=1

A.2 Host SLAAC/DHCPv6 Switching Behavior

- o SLAAC-only host receiving A=0&M=1
 - Window 7 would deprecate SLAAC and initiate DHCPv6
 - Linux/MAC would keep SLAAC and don't initiate DHCPv6 unless SLAAC is expired and no continuous RA
- o DHCPv6-only host receiving A=1&M=0
 - Window 7 would release DHCPv6 and do SLAAC
 - Linux/MAC would keep DHCPv6 and do SLAAC

When the host has been configured, either by SLAAC or DHCPv6, the operating systems interpreting the M flag quite differently. Windows 7 treats the flag as instruction, it even released DHCPv6 session when M=0. Linux and OS X were likely to treat the flag as advisory,

when SLAAC was done, it won't care about M=1, and M=0 won't cause operation for the already configured DHCPv6 addresses.

Please refer to [I-D.liu-6renum-dhcpv6-slaac-switching] for more details.

A.3 Host Stateful/Stateless DHCPv6 Behavior

- o StatelessDHCPv6-configured host receiving M=1 (while keeping O=1)
 - Window 7 would initiate stateful DHCPv6, configuring address as well as re-configuring other information
 - Linux/MAC no action
- o StatefulDHCPv6-configured host receiving M=0 (while keeping O=1)
 - Window 7 would release all DHCPv6 configurations including address and other information, and initiate stateless DHCPv6
 - Linux/MAC no action

Authors' Addresses

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

Email: leo.liubing@huawei.com

Ron Bonica
Juniper Networks
Sterling, Virginia 20164
USA

Email: rbonica@juniper.net

IPv6 maintenance Working Group (6man)
INTERNET-DRAFT
Updates RFC 3971 , RFC 3972, RFC 4941
(if approved)
Intended status: Standard Track
Expires: August 25, 2013

H. Rafiee
C. Meinel
Hasso Plattner Institute

February 25, 2013

A Simple Secure Addressing Generation Scheme for IPv6 AutoConfiguration
(SSAS)
<draft-rafiee-6man-ssas-02.txt>

Abstract

The default method for IPv6 address generation uses an Organizationally Unique Identifier (OUI) assigned by the IEEE Standards Association and an Extension Identifier assigned to the hardware manufacturer [1] (section 2.5.1 RFC-4291) [RFC4291]. This means that a node will always have the same Interface ID (IID) whenever it connects to a new network. Since the node's IP address does not change, the node is vulnerable to privacy related attacks. To address this problem there are currently two mechanisms being used to randomize the IID that do not use the MAC address or other unique values in the IID generation; Cryptographically Generated Addresses (CGA) [RFC3972] and Privacy Extension [RFC4941]. The problem with the former approach is the computational cost involved for the IID generation and verification. The problem with the latter approach is that it lacks the necessary security and provides the node with only partial protection against privacy related attacks. This document proposes the use of a new algorithm for use in the generation of the IID while, at the same time, securing the node against some types of attack, like IP spoofing. These attacks are prevented with the addition of a signature to the messages sent over the network and by direct use of a public key in the IP address.

Status of this Memo

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

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

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

material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 21, 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. Conventions used in this document	3
2. Problem Statement	4
2.1. SSAS Applications	5
2.1.1. Preventing Attacks	5
2.1.1.1. Replay attack	5
2.1.1.2. IP spoofing	5
2.1.1.3. Denial of Service (DoS) attacks	5
2.1.1.4. Spoofed Redirect Message	6
2.1.2. Nodes with limited resources	6
3. Algorithm Overview	6
3.1. Interface ID (IID) Generation	6
3.2. Signature Generation	9
3.3. Generation of NDP Messages	9
3.3.1. SSAS signature data field	10
3.4. SSAS verification process	11
4. Security Considerations	12
5. IANA Considerations	13
6. Conclusions	13
7. References	14
7.1. Normative	14
7.2. Informative	14
Authors' Addresses	15

Introduction

IPv6 addresses consist of two parts; the subnet prefix, which is the 64 leftmost bits of the IPv6 address, and the Interface ID (IID), which is the 64 rightmost bits of the IPv6 address. The IEEE Standards Association [1] (section 2.5.1 RFC-4291) [RFC4291] offered a standard for the generation of the IPv6 Interface IDs (IID) which it called the Extended Unique Identifier (EUI-64). EUI-64s are generated by the concatenation of an Organizationally Unique Identifier (OUI) assigned by the IEEE Registration Authority (IEEE RA) with the Extension Identifier assigned by the hardware manufacturer. For example, if a manufacturer's OUI-36 hexadecimal value is 00-5A-D1-02-3, and the manufacture hexadecimal value, for the Extension Identifier for a given component, is 4-42-61-71, then the EUI-64 value generated from these two numbers will be 00-5A-D1-02-34-42-61-71. If the OUI is 24 bits and the extension identifier is also 24 bits (this constitutes the MAC address), then to form the 64-bit EUI address, the OUI portion of the MAC address is inserted into the leftmost 24 bits of the EUI-64 8 byte field and the Extension Identifier is inserted into the rightmost 24 bits of the EUI-64 8 byte field, and then a value of 0xFFFE is inserted between these two 24-bit items. IEEE has chosen 0xFFFE as a reserved value which can only appear in an EUI-64 generated from an EUI-48 MAC address. Then bit 7 (u bit) in the OUI portion of the address should be set. Globally unique addresses assigned by the IEEE set this bit to zero by default indicating global uniqueness. This bit will be set to 1 for locally created addresses, such as those used for virtual interfaces or a MAC address manually configured by an administrator.

There are two mechanisms used to generate a randomized IID that do not make use of a MAC address; CGA [RFC3972] and Privacy Extension [RFC4941]. In this document we discuss the problem inherent with using the current mechanisms and then we explain our solution to the problem, which is to randomize the IID and observing privacy, while, at the same time, providing security to Neighbor Discovery Protocol (NDP) messages, for nodes, in the IP layer. DHCPv6 [RFC3315] can also benefit from this approach for the generation of a random IID or for authentication purposes.

1. Conventions used in this document

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

2. Problem Statement

The drawback to using IIDs that do not change over time is one of privacy. The node will generate the same IID whenever it joins a new network thus making it easy for an attacker to track that node when it moves to different networks.

The main problem with the privacy extension mechanism, when using the first approach as explained in section 3.2.1 RFC-4941 [RFC4941], i.e., using stable storage, is the lack of a provision for the use of a security mechanism. The Privacy Extension RFC can partly prevent attacks related to privacy issues, but it cannot prevent attacks related to security issues. For instance, it cannot prevent IP spoofing attacks and it cannot provide proof of the IP address ownership of a node. If one wants to use a secure method, with the privacy extension, then one needs to use CGA. The problem with using CGA is in the computational overhead necessary to compute it when a higher sec value is used and the time that is needed in the verification process. This time is based on the reverse of the steps required to regenerate CGA during the verification process, in addition to the signature verification.

What is clear here is that it is not possible to generate the CGA offline or before hand. This is because the subnet prefix (router prefix) is one of the inputs to the SHA1 algorithm. The other problem with CGA is the apparent lack of a defense against Denial of Service (DoS) types of attack against verifier nodes. In the CGA RFC, there is no explanation as to how to prevent these types of attacks. This means that an attacker can overwhelm the verifier node with false CGA values thus rendering it unable to process further messages. This document also proposes a solution for this type of attack.

To overcome the problem with using the other mechanisms the time needed for IP address generation and verification needs to be reduced. We propose the use of the SSAS algorithm, along with the SSAS signature, to provide a node with the protection it needs to protect it against IP spoofing and spoofing types of attack in the IP layer. Our experimental results [2] show that SSAS is 5 times faster than CGA, when using a sec value of ,0 and 600 times faster than CGA when using the sec value of 1. This will be the same when, in the future, we have faster CPUs because SSAS will also benefit from the future technologies. Currently the generation time for SSAS is less than 1 millisecond so with future new technologies it will be even less.

Note: It is not the intent of this document to obsolete CGA but to propose a simpler and a faster addressing mechanism to use in the randomization of the IID and the for the protection of nodes against the attacks explained below.

2.1. SSAS Applications

2.1.1. Preventing Attacks

The following sections detail some of the attacks that SSAS can prevent.

2.1.1.1. Replay attack

In this type of attack, an attacker might sniff the Neighbor Discovery Protocol enabled networks (NDP) messages and try to copy the legitimate signature and public key to his NDP message and then send this to the sender. But by using the SSAS algorithm, this is prevented with the addition of a timestamp to the NDP message and also with inclusion of this timestamp in the signature. The use of the timestamp works because the timestamp will be valid for a short period of time. (this accounts for clock skews.)

2.1.1.2. IP spoofing

This is a well-known type of attack in NDP. This type of attack is used to attack the Duplicate Address Detection process. In this attack, when a node joins the network and generates a new IP address, the node sends a Neighbor Solicitation (NS) message to check for address collisions in the network. The attacker, in this scenario, spoofs the IP address and responds back to the node with a Neighbor Advertisement (NA) message claiming ownership of this IP address. The SSAS algorithm allows this node to verify other nodes in the network. An attacker does not have the private key for this node, which is needed to generate a SSAS signature, so the verification process will fail.

2.1.1.3. Denial of Service (DoS) attacks

An attacker might send many NDP messages, using invalid signatures, to the victim's node which then forces the node to busy itself with the verification process. To mitigate this attack, a node SHOULD set a limit on the number of messages (x) that it can verify, per a certain period of time. Implementations MUST provide a conservative default and SHOULD provide a way for detecting when this limit is reached.

2.1.1.4. Spoofed Redirect Message

Redirect messages, imitating the end host needing redirection, can be sent from any router on the same broadcast segment. The attacker uses the link-local address of the current first-hop router in order to send a Redirect message to a legitimate node. Since that node identifies the message as coming from its first hop router, by use of the link-local address, it accepts the Redirect. The Redirect will remain in effect as long as the attacker responds to the Neighbor Unreachability Detection probes sent to the link-layer address. To preclude this from occurring, the address ownership of the first-hop router should be verified. The use of the SSAS verification process will prevent such an attack.

2.1.2. Nodes with limited resources

SSAS can be used in nodes where limited resources are available for computation. It can provide protection for these nodes against the attacks stated above. Sensor networks are examples of nodes with limited resources (such as battery, CPU, and etc); see RFC-4919 [RFC4919] for the usage of IPv6 in these networks.

Another example could be the use of SSAS in mobile networks during the generation of IP addresses as explained in section 4.4 RFC-6275. The current problem with addressing mechanism in mobile node is that there is no privacy observation as the node usually keeps its Home Address when it moves to another network. If there is a fast secure mechanism, then it is possible set this Home Address and change it and re-register it to the Home network.

3. Algorithm Overview

As explained earlier, one of the problems with the current IID generation approach is the compute intensive processing needed for the IID algorithm generation. Another concern is the lack of security. Since, we assume that a node needs to generate and keep its address for a short time, we tried to keep the IID generation process to a minimum. We also tried to remain within the confines of NDP protocol.

3.1. Interface ID (IID) Generation

To generate the IID, a node needs to execute the following steps.

1. Generate a 16 byte random number called modifier.

2. Generate a 1024-bit key pair (public/private key). These keys SHOULD be stored in a safe place on a local hard disk and the path to this data, and the validation time for these keys, SHOULD be saved in a XML file. It is RECOMMENDED that the public key be generated, on the fly, during the start-up phase of the algorithm generation.

Once a node generates key pairs, it can make use of these keys for a short period of time. It is RECOMMENDED not to use the same keys for more than 10 days in order to prevent the node from being tracked through the use of its public keys. When time expires for the use of these key pairs, the node should generate new key pairs and replace the old one in the XML file. It SHOULD then use the new value for IP address and signature generation.

It is also possible to use ECC [3] with a 192 bit key size. This is equivalent to a 1280 bit RSA key size. In this case the packet size would be decreased by a factor 5 times smaller than when using RSA. However, with key sizes 1024 bit and 1280 bit, RSA generation and verification is much faster than ECC. The other problem with the use of ECC is that it could be patented and might not be royalty free.

3. Concatenate the modifier with the timestamp and the public key. The timestamp is a 64-bit unsigned integer field containing a timestamp. The value indicates the number of seconds since January 1, 1970, 00:00 UTC, by using a fixed point format. The format of the timestamp data field is the same as that outlined in section 5.3.1 RFC-3971 [RFC3971].

```
R1=(modifier(16 bytes)||timestamp(8 bytes)||public key)
```

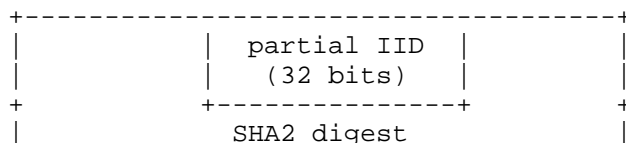
4. Execute SHA2 (256) on the result from step 3.

```
digest=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).

5. Generate a random number between 0 and 20 and call it the start index. This number is used as an index for the SHA2 array of bytes. This value helps randomize the IID and to minimize the chance of a collision in the network. The length of this number is one byte.

6. Take the 32 leftmost bits (starting at the start index) from the resulting output from step 5 (SHA2 digest) and set bits u and g (bits 7 and 8) and call this the partial IID.



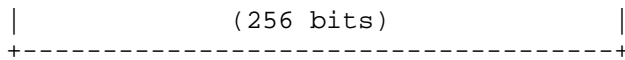


Figure 1 Partial Interface ID

7. Obtain the second byte of the partial IID and call it the start field pubkey. If the value of the start field pubkey is between 0 and the size of public key length, in bytes, minus 4, use this number as an index for the public key array of bytes. Otherwise choose that byte and shift its contents 2 bits to the right (the first two bits will be zero) and set the start field pubkey to this number. This ensures that the value of the start field pubkey will be less than the size of the public key array of bytes, minus 4. This value helps randomize the IID and minimize the chance of a collision in the network. For example, if the second byte of partial IID is 110, the

start field pubkey value will be 110. This value helps randomize the IID and minimize the chance of a collision in the network. For example, if the second byte of the partial IID is 110, then the start field pubkey value will be 110.

If ECC is used for key generation, then the content of the start field pubkey SHOULD be shifted 3 bits to the right. This insures that its value is less than the size of public key array of bytes, minus 4.

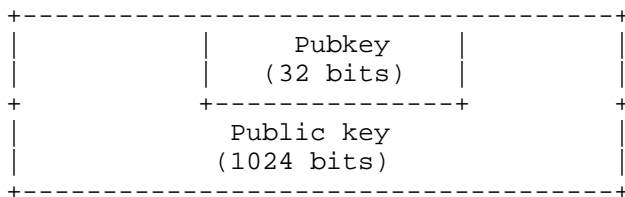


Figure 2 Public key part of Interface ID

8. Concatenate the partial IID with the four bytes from the public key (starting at the start field pubkey) and call this the IID.

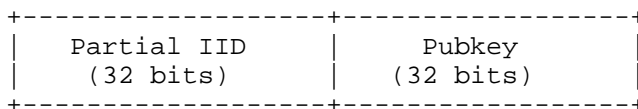


Figure 3 Interface ID

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

10. Concatenate the IID with the router subnet prefix (Global subnet prefix), obtained from the RA message, and set it as a tentative

global IP address. (This IP will be permanent after Duplicate Address Detection (DAD) processing. (for more information about DAD refer to section 4.3.)

3.2. Signature Generation

The SSAS signature is added to NDP messages in order to protect them from IP spoofing and spoofing types of attack. SSAS will prove IP address ownership, as does the CGA generation algorithm, but using fewer steps. To generate the SSAS signature, the node needs to execute the following steps:

1. Concatenate the timestamp with the 16 byte public key (that starts at the start field pubkey) (see figure 4) and the global IP address. The start field pubkey is one of the numbers that was introduced in step 7 of section 4.1.
2. Sign the resulting value from step 1, using the RSA private key unless we use ECC, and call the resulting output the SSAS signature.

timestamp	Public key	Global IP Address	Other Options
(8 bytes)	(16 bytes)	(16 bytes)	(variable)

Figure 4 SSAS Signature

If NDP messages contain other data that must be protected, such as important routing information, 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, 16 byte public key and Global IP address (Router subnet prefix plus IID).

3.3. Generation of NDP Messages

After a node generates its IP address, it should then process Duplicate Address Detection in order to avoid address collisions in the network. To do this, the node generates a Neighbor Solicitation (NS) message. The format of a NS message is shown in figure 5. 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 type, length, reserved, Other Len, pubkey len, public key, SSAS signature, and padding.

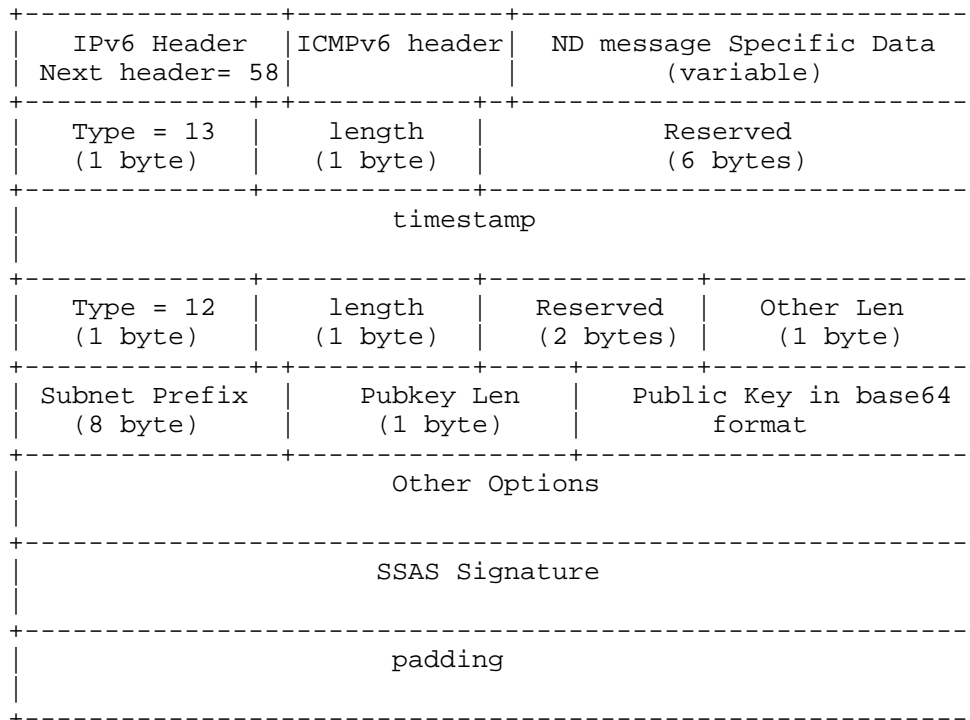


Figure 5 NDP Message Format with SSAS Signature Data Field

This document proposes an update to the SEND RFC in order to replace the RSA signature field with the SSAS signature data field and to add SSAS as a new option to SEND messages.

3.3.1. SSAS signature data field

- Type: This option should be set to 12.
- Length: The length of the Signature Data field, including the Type, Length, Reserved, pubkey Len, public key, Signature and padding, should 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 MUST be ignored by the receiver.
- Other Len: The length of other options in multiples of eight. The length of this is 1 byte.
- Subnet Prefix: This is the router subnet prefix.

- PubKey Len. The length of the public key in multiples of eight.
- Public key. Base64 format of the public key
- Other Options. This variable-length field contains important data that needs to be protected in the packet . The padding would be added, as many bytes long as remain after the end of the field, if the Other options is not a multiple of eight.
- Padding. This variable-length field contains padding, as many bytes long as remain after the end of the signature, if the signature is not a multiple of eight.

All NDP 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 modifier by one and again repeats steps 3 thru 8 of section 4.1 . If, for a second time, the node receives the same claim, then it considers it an attack and will use that IP address.

3.4. SSAS verification process

A node's verification process should start when it receives NDP messages.

Following are the verification steps:

1. Obtain the timestamp from the NDP message and call this value t1.
2. Obtain the timestamp from the node's system, convert it to UTC, and call this value t2.
3. If $(t2 - x) \leq t1 \leq (t2 + x)$ go to stop 4. Otherwise, the message SHOULD be discarded without further processing. (The value of x is dependent on network delays and network policy. One might choose 10 minutes (600 seconds) as a flexible way of handling network delays.)
4. Obtain the public key from the SSAS signature data field.
5. 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.
6. Obtain the second byte of the partial IID and call it the start field pubkey. If the value of the start field pubkey is between 0 and the size of public key length, in bytes, minus 4, use this number as an index for the public key array of bytes. Otherwise choose that byte and shift its contents 2 bits to the right (the first two bits

will be zero) and consider this number the starting index of the public key array of bytes. This ensures that the value of that byte will be less than the size of the public key array of bytes, minus 4. Set the start field pubkey to this number.

If ECC is used for key generation, then the content of the start field pubkey SHOULD be shifted 3 bits to the right. This insures that its value is less than the size of public key array of bytes, minus 4.

7. Obtain the IID from the sender's source IP address. (64 rightmost bits of the IPv6 address)

8. Compare the 32 leftmost bits, starting at the start field pubkey of the public key, to the 32 rightmost bits of the IID of the sender's IP address. If they are the same, go to the next step. Otherwise, the message should be discarded without further processing

9. Obtain the subnet prefix from the SSAS signature data field.

10. Concatenate the timestamp with the 16 bytes of the public key, (starting from start field pubkey), the subnet prefix, the sender's IID, and other options (if any) and call this entity the plain message.

11. Obtain the SSAS signature from the SSAS signature data field.

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.

4. Security Considerations

As a security consideration what one might ask is what are the odds of an attacker being able to generate a public key having four sequential bytes the same as the last rightmost 32 bits of the IID If he could, he could then generate the signature using his own private key and thus break the SSAS.

Mathematically it has been shown that the probability of matching 32 bits in the public key against 32 bits in the IID is about

$\text{pow}(1/2, 32)$ where pow is the power function, 2 is a base and 32 is a exponent. Since the use of a public key and IP address with a maximum lifetime of 10 days is RECOMMENDED, the probability of an attacker finding the same value is 0.0008, a very small value. When one also considers the probability of an attacker being able to generate a public key whose 32 bits, starting from an arbitrary point, matches the 32 bits of the public key generated using the SSAS algorithm, then the probability of his success is diminished even further. This shows the strength of this algorithm against brute force attacks while, at the same time, by using the signature and finding a binding between the IP address and the public key, it provides proof of IP address ownership at a speed that is about 600 times faster than that of the CGA algorithm [2]. (based on the implementation results, the average time to generate SSAS is 882.77 microseconds).

Another consideration concerns Routers wanting to use this algorithm in place of CGA. As explained in RFC SEND, for routers, the use of a Trusted Authority is RECOMMENDED along with verifying router certificates using these third parties. This will prevent a node from claiming to be a router. But for nodes, rather than routers, SSAS can provide protection against the types of attacks explained above.

5. IANA Considerations

This document defines a new algorithm for the generation of an Interface ID in IPv6 networks.

6. Conclusions

Privacy has become a very important issue in recent years. A solution for preventing a node from being tracked by an attacker is to change the node's IP address frequently and by generating a random IID each time a node wants to generate a new IP address. There are two solutions available for randomizing the IID; CGA and Privacy Extension. The former algorithm is compute intensive and the latter algorithm is lacking in security. This document introduced a new algorithm as a solution for providing privacy by randomizing the IID and for providing security with the addition of a SSAS signature to the NDP message and finding a binding between the public key and the IP address. Our experimental results [2] show a definite improvement in the computation time for the SSAS algorithm as compared to that for the CGA algorithm. We also note that the probability of having collisions with IP addresses, when using the SHA2 digest and the public key, with a randomized 62 bit selection, approximates $\text{pow}(1/2, 62)$ where pow is the power function, 2 is a base and 62 is a

exponent (u and g bits are ignored) . Moreover, the probability of an attacker finding the public key which matches 32 rightmost bits of the IID within 10 days approximates 0.0008. This means this algorithm is secure enough for wide usage.

7. References

7.1. Normative References

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

7.2. Informative References

- [1] IEEE Standards Association,
<http://standards.ieee.org/develop/regauth/tut/eui64.pdf>, 2012
- [2] Rafiee, H., "Research Results",
http://ipv6sra.rozanak.com/Jan2013_CGA_SSAS_Comparison.pdf, 2013
- [3] Brown, R., L., D. : SEC 1: Elliptic Curve Cryptography, Certicom Research,
<http://www.secg.org/download/aid-780/sec1-v2.pdf>, 2009

Authors' Addresses

Hosnieh Rafiee
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
Potsdam, Germany
Phone: +49 (0)331-5509-546
Email: ietf@rozanak.com

Dr. Christoph Meinel
(Professor)
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	Mac address	Collision Count	Algorithm type
8 bytes	6 bytes	3 bits	1 byte
Global IP address		Other Options	
16 bytes		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	Length	Reserved	Other len
1 byte	1 byte	2 bytes	1 byte
Algorithm type	Collision count	Subnet prefix	Other Options
1 byte	3 bits	8bytes	
Hash Function	Response No.	SSAS Signature	

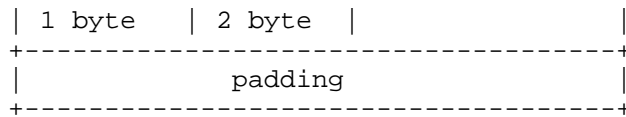


Figure 2 NDP Message Format with SSAS Signature Data Field

- 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 => ~ $1.16 \cdot 10^6$ years

sec value 5 => ~ $1 \cdot 10^{11}$ years

sec value 6 => ~ $6.8 \cdot 10^{15}$ 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] Gundavell, 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-rafee-6man-cga-attack>, 2014

[RAgaurd] 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-rafee-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-rafiiee-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

6LoWPAN
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

P. Thubert
Cisco
February 25, 2013

6LoWPAN Backbone Router
draft-thubert-6lowpan-backbone-router-03

Abstract

Some LLN subnets are expected to scale up to the thousands of nodes and hundreds of routers. This paper proposes an IPv6 version of the Backbone Router concept that enables such a degree of scalability using a high speed network as a backbone to the subnet.

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 29, 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. Terminology	5
3. Overview	7
4. New types and formats	9
4.1. The Enhanced Address Registration Option (EARO)	9
5. Backbone Router Operations	11
5.1. Backbone Link and Router	11
5.2. ND Proxy Operations	11
5.3. Claiming and Defending Addresses	12
5.4. Conflict Resolution	13
5.5. Assessing an entry	14
6. Security Considerations	15
7. IANA Considerations	16
8. Acknowledgments	17
9. References	18
9.1. Normative References	18
9.2. Informative References	19
9.3. External Informative References	19
Author's Address	20

1. Introduction

In order to meet industrial requirements for non-critical monitoring, alerting, supervisory control, open loop control and some closed loop control applications, both [ISA100.11a] and wireless [HART] standards leverage advanced technology at every layer, including the Time Synchronized Channel Hopping (TSCH) Medium Access Control (MAC) operation that enables deterministic behaviours for time-sensitive flows. Additionally, [ISA100.11a] endorsed the 6LoWPAN Header Compression [RFC6282] format for the network header, making it possible to utilize IPv6 based protocols such as BACnet IP, Profibus IP and Modbus TCP without significant changes to those protocols.

[ISA100.11a] has introduced the concept of a Backbone Router that would interconnect small LLNs over a high speed Backbone network and scale a single ISA100.11a network up to the thousands of nodes. In that model the LLNs and the backbone form a single subnet in which nodes can move freely without the need of renumbering, and the Backbone Router is a special kind of Border Router designed to manage the interaction between the LLNs and the backbone at layer 3. Similar scalability requirements exist in the metering and monitoring industries. In a network that large, it is impossible for a node to register to all Border Routers as suggested for smaller topologies in Neighbor Discovery Optimization for Low-power and Lossy Networks [RFC6775].

This paper specifies IP layer functionalities that are required to implement the concept of a Backbone Router with IPv6, in particular the application of the "IP Version 6 Addressing Architecture" [RFC4291], " the Neighbor Discovery Protocol" [RFC4861] and "IPv6 Stateless Address Autoconfiguration" [RFC4862].

The use of EUI-64 based link local addresses, Neighbor Discovery Proxying [RFC4389], Neighbor Discovery Optimization for Low-power and Lossy Networks [RFC6775], the IPv6 Routing Protocol for Low power and Lossy Networks [RFC6550] , the mixed mode of Efficiency aware IPv6 Neighbor Discovery Optimizations [I-D.chakrabarti-nordmark-6man-efficient-nd] and Optimistic Duplicate Address Detection [RFC4429] are discussed. Also, the concept of Backbone Link is introduced to implement the backbone network that was envisioned by ISA100.11a.

This operation of the Backbone Router requires that some protocol operates over the LLNs from which node registrations can be obtained, and that can disseminate the location of the backbone Router over the LLN. Further expectations will be detailed.

The way the PAN IDs and 16-bit short addresses are allocated and

distributed in the case of an 802.15.4 network is not covered by this specification. Similarly, the aspects of joining and securing the network are out of scope. The way the nodes in the LLN learn about their Backbone Router depends on the protocol used in the LLN. In the case of RPL, a Border Router is the root of the DODAG that it serves and represents all nodes attached to that DODAG.

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].

Readers are expected to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861], "IPv6 Stateless Address Autoconfiguration" [RFC4862], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], "Neighbor Discovery Optimization for Low-power and Lossy Networks" [RFC6775] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

Readers would benefit from reading "Mobility Support in IPv6" [RFC3775], "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] and "Optimistic Duplicate Address Detection" [RFC4429] prior to this specification for a clear understanding of the art in ND-proxying and binding.

Additionally, this document uses terminology from [I-D.ietf-roll-terminology], and introduces the following terminology:

Backbone This is an IPv6 Backbone link that interconnects 2 or more Backbone Routers. It is expected to be deployed as a high speed backbone in order to federate a potentially large set of LLNs. Also referred to as a LLN backbone or Backbone network.

Backbone Router An IPv6 router that federates the LLN using a Backbone link as a backbone. A BBR acts as a 6LoWPAN Border Routers (6LBR) and an Energy Aware Default Router (NEAR).

Extended LLN This is the aggregation of multiple LLNs as defined in [RFC4919] interconnected by a Backbone Link via Backbone Routers and forming a single IPv6 link.

Energy-Constrained Node An IPv6 node that operates ND registration in order to save energy. This can be a LLN node, or an Efficiency-Aware Host(EAH) on the backbone.

Binding The association of the Energy-Constrained Node IPv6 address and Interface ID with associated proxying states including the remaining lifetime of that association.

Registration The process during which a Energy-Constrained Node injects its address in a protocol through which the Border Router can learn the address and proxy ND for it.

Primary BBR

The BBR that will defend a registered address for the purpose of DAD over the backbone

Secondary BBR

A BBR to which the address is registered. A Secondary Router MAY advertise the address over the backbone and proxy for it.

3. Overview

The scope of this draft is a Backbone Link that federates multiple LLNs as a single IPv6 subnet. Each LLN in the subnet is anchored at a Backbone Router (BBR). The Backbone Routers interconnect the LLNs over the Backbone Link and emulate that the LLN nodes are present on the Backbone by proxy-ND operations. An LLN node can move freely from an LLN anchored at a Backbone Router to an LLN anchored at another Backbone Router on the same backbone and conserve any of the IPv6 addresses that it has formed. In a same fashion, an Efficiency-Aware Host (EAH) residing on the Backbone may change its BBR - acting as IPv6 ND-efficiency-aware Router (NEAR) - and conserve its addresses with no disruption.

Energy-Constrained Nodes are often associated with radios and therefore may change their point of attachment in the network. Virtual devices - typically virtual machines in a datacenter - also move though in a different fashion, from a physical device to the next. In the case if a movement, it might be difficult for Stateful devices in the network such as the NEAR and SAVI switches to differentiate a duplication from a movement. And if indeed it is a movement, then it might be difficult to select to freshest information to know where the device actually is.

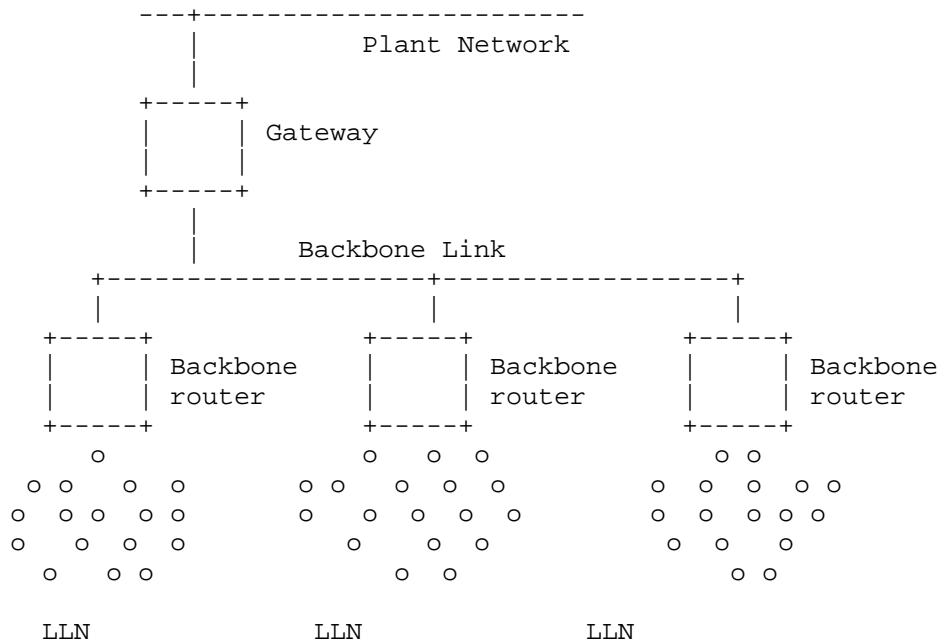


Figure 1: Backbone Link and Backbone Routers

The Backbone Link is used as reference for Neighbor Discovery operations, by extending the concept of a Home Link as defined in [RFC3775] for Mobile IPv6. In particular, Backbone Routers perform ND proxying for the Energy-Constrained Nodes in the LLNs they own through a node registration.

The Backbone Router operation is compatible with that of a Home Agent. This enables mobility support for LLN devices that would move outside of the network delimited by the Backbone link. This also enables collocation of Home Agent functionality within Backbone Router functionality on the same interface of a router.

A Energy-Constrained Node registers and claims ownership of its address(es) using proactive acknowledged registration exchanges with a neighboring router. In case of a complex LLN topology, the router might be an intermediate LLN Router that relays the registration to the BBR (acting as LBR) as described in [RFC6775]. In turn, the Backbone Routers operate as a distributed database of all the Energy-Constrained Nodes whether they reside on the LLNs or the backbone, and use the Neighbor Discovery Protocol to share that information across the Backbone in the classical ND reactive fashion.

For the purpose of Neighbor Discovery proxying, this specification documents the LLN Master Neighbor Registry, a conceptual data structure that is similar to the MIP6 binding cache. The Master Neighbor Registry is fed by redistributing addresses learnt from the registration protocol used over the LLN.

Another function of the Backbone Router is to perform 6LoWPAN compression and expansion between the LLN and the Backbone Link and ensure MTU compatibility. Packets flow uncompressed over the Backbone Link and are routed normally towards a Gateway or an Application sitting on the Backbone link or on a different link that is reachable over the IP network.

4. New types and formats

The specification expects that the protocol running on the LLN can provide a sequence number called Transaction ID (TID) that is associated to the registration. When a node registers to multiple BBRs, it is expected that the same TID is used, to enable the BBR to correlate the registrations as being a single one, and differentiate that situation from a movement. Otherwise, the resolution makes it so that only the most recent registration was perceived from the highest TID is kept.

The specification expects that the protocol running on the LLN can provide a unique ID for the owner of the address that is being registered. The Owner Unique ID enables to differentiate a duplicate registration from a double registration. In case of a duplicate, the last registration loses. The Owner Unique ID can be as simple as a EUI-64 burnin address, if the device manufacturer is convinced that there can not be a manuf error that would cause duplicate EUI64 addresses. Alternatively, the unique ID can be a hash of supposedly unique information from multiple orthogonal sources, for instance:

- o Burn in address.
- o configured address, id, security keys...
- o (pseudo) Random number, radio link metrics ...

In any fashion, it is recommended that the device stores the unique Id in persistent memory. Otherwise, it will be prevented to reregister after a reboot that would cause a loss of memory until the Backbone Router times out the registration.

The unique ID and the sequence number are placed in a new ND option that is used by the Backbone Routers over the Backbone link to detect duplicates and movements. The option format is as follows:

4.1. The Enhanced Address Registration Option (EARO)

This option is designed to be used with standard NS and NA messages between backbone Routers over a backbone link and may be used between LRs and LBRs over the LLN. By using this option, the binding in question can be uniquely identified and matched with the Master Neighbor Registry entries of each Backbone Router.

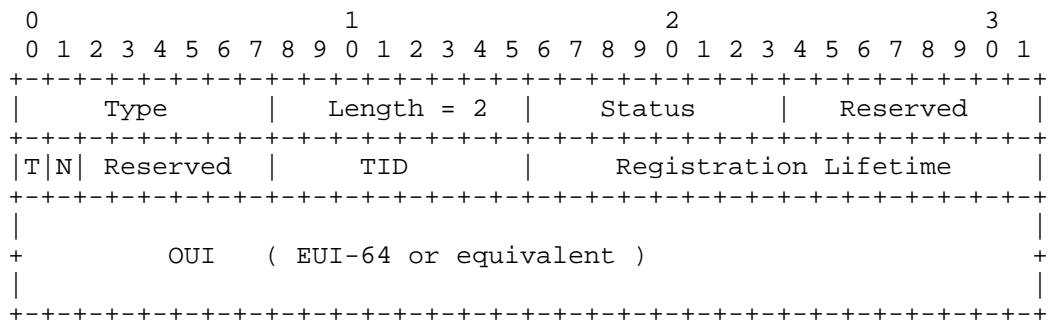


Figure 2: EARO

Option Fields

Type:

Length: 2

T: One bit flag. Set if the next octet is a used as a TID.

N: One bit flag. et if the device moved. If not set, the router will refrain from sending gratuitous NA(0) over the backbone, for instance after the DAD operation upon entry creation.

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

TID: 1-byte integer; a transaction id that is maintained by the device and incremented with each transaction. it is recommended that the device maintains the TID in a persistent storage.

Owner Unique Identifier: A globally unique identifier for the host's interface associated with the binding for the NS/NA message in question. This can be the EUI-64 derived IID of an interface, which can be hashed with other supposedly unique information from multiple orthogonal sources.

5. Backbone Router Operations

5.1. Backbone Link and Router

The Backbone Router is a specific kind of Border Router that performs proxy Neighbor Discovery on its backbone interface on behalf of the nodes that it has discovered on its Low Power Lossy Network interfaces. On the LLN side, the Backbone Router acquires its states about the nodes by terminating protocols such as RPL [RFC6550] or 6LoWPAN ND [RFC6775] as a LLN Border Router. It is expected that the backbone is the medium used to connect the subnet to the rest of the infrastructure, and that all the LBRs are connected to that backbone and support the Backbone Router feature as specified in this document.

The backbone is expected to be a high speed, reliable Backbone link, with affordable multicast capabilities, such as an Ethernet Network or a fully meshed NBMA network with multicast emulation, which allows a full support of classical ND as specified in [RFC4861] and subsequent RFCs. In other words, the backbone is not a LLN. Still, some restrictions of the attached LLNs will apply to the backbone. In particular, it is expected that the MTU is set to the same value on the backbone and all attached LLNs.

5.2. ND Proxy Operations

This specification enables a Backbone Router to proxy Neighbor Discovery operations over the backbone on behalf of the nodes that are registered to it, allowing any device on the backbone to reach a Energy-Constrained Node as if it was on-link.

In the context of this specification, proxy ND means:

- o defending a registered address over the backbone using NA messages with the Override bit set
- o advertising a registered address over the backbone using NA messages, asynchronously or as a response to a Neighbor Solicitation messages.
- o Looking up a destination over the backbone in order to deliver packets arriving from the LLN using Neighbor Solicitation messages.
- o Forwarding packets from the LLN over the backbone, and the other way around.

- o Eventually triggering a look up for a destination over the LLN that would not be registered at a given point of time, or as a verification of a registration.

The draft introduces the concept of primary and secondary BBRs. The concept is defined with the granularity of an address, that is a given BBR can be primary for a given address and secondary or another one, regardless on whether the addresses belong to the same node or not. The primary Backbone Router is in charge of protecting the address for DAD over the Backbone. Any of the Primary and Secondary BBR may claim the address over the backbone, since they are all capable to route from the backbone to the LLN device.

When the protocol used to register the nodes over the LLN is RPL [RFC6550], it is expected that one BBR acts as virtual root coordinating LLN BBRs (with the same DODAGID) over the non-LLN backbone. In that case, the virtual root may act as primary BBR for all addresses that it cares to support, whereas the physical roots to which the node is attached are secondary BBRs. It is also possible in a given deployment that the DODAGs are not coordinated. In that case, there is no virtual root and no secondary BBR; the DODAG root is primary all the nodes registered to it over the backbone.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [RFC6775], the Backbone Routers act as a distributed DAD table, using classical ND over the backbone to detect duplication. This specification requires that:

1. Registrations for all addresses that can be required to reach the device over the backbone, including registrations for IPv6 addresses based on burn-in EUI64 addresses are passed to the DAD table.
2. Nodes include the EARO in their NS used for registering those addresses and the LRs propagate that option to the LBRs.

A false positive duplicate detection may arise over the backbone, for instance if the node registers to more than one LBR, or if the node has moved. Both situations are handled gracefully unbeknownst to the node. In the former case, one LBR becomes primary to defend the address over the backbone while the others become secondary and may still forward packets back and forth. In the latter case the LBR that receives the newest registration wins and becomes primary.

5.3. Claiming and Defending Addresses

Upon a new or an updated registration, the BBR performs a DAD operation. If either a TID or a OUI is available, the BBR places

them in a EARO in all its ND messages over the backbone. If content is not available for a given field, it is set to 0.

If a primary already exists over the backbone, it will answer the DAD with an RA.

- o If a RA is received with the O bit set, the primary rejects the DAD and the DAD fails. the BBR needs to inform the LLN protocol that the address is a duplicate.
- o If a RA is received with the O bit reset, the primary accepts the BBR as secondary and DAD succeeds. The BBR may install or maintain its proxy states for that address. This router MAY advertise the address using a NA. during a registration flow, it MAY set the O bit.
- o If no RA is received, this router assumes the role of primary and DAD succeeds. The BBR may install or maintain its proxy states for that address. This router advertises the address using a NA with the O bit reset.

When the BBR installs or maintains its proxy states for an address, it sends an NA with a SLLA option for that address. The Primary BR MAY set the O bit if it wished to attract the traffic for that address.

5.4. Conflict Resolution

A conflict arise when multiple BBRs get a registration from a same address. This situation might arise when a node moves from a BBR to another, when a node registers to multiple BBRs, or in the RPL case when the BRs belong to a single coordinated DODAG.

The primary looks up the EARO in ND messages with a SLLA option.

- o If there is no option, normal ND operation takes place and the primary defends the address with a NA with the O bit set, adding the EARO with its own information.
- o If there is a EARO and the OUIs are different, then the conflict apparently happens between different nodes, and the the primary defends the address with a NA with the O bit set, adding the EARO with its own information. If the TID in the EARO is in the straight part of the lollipop, it is possible that the request comes from the same node that has rebooted and formed a new OUI. The primary BBR may assess its registered entry prior to answering.

- o If there is a EARO and the OUIs are the same:
 - * If the TID in the ND message is newer than the most recent one known by the primary router, this is interpreted as a node moving. The primary cleans up its states and stops defending the address.
 - * If the TID in the ND message is the same as the most recent one known by the primary router, this is interpreted as a double registration. In case of a DAD, the primary responds with a NA with the O bit reset, to confirm its position as primary, including the EARO.
 - * If the TID in the ND message is older than the most recent one known by the primary router, this is interpreted as a stale information. The primary defends the address with a NA with the O bit set, adding the EARO with its own information.
 - * If the TIDs are very different (more than 16 apart, discounting the straight part of the lollipop), it is impossible to resolve for sure. The primary BBR should assess its registered entry prior to answering.

5.5. Assessing an entry

In a number of cases, it might happen that the information at the primary BBR is stale and obsolete. In particular, a node with no permanent storage might reboot and form a different OUI, in which case the information at the BBR is inaccurate and should be removed. In another case, the BBR and the node have been out of reach for too long and the TID that the BBR maintains is so far out that it is impossible to compare it with that stored at the BBR.

In such situation, the primary Backbone Router has the possibility to assess the registration. This is performed by the protocol in use to register the node over the LLN.

When the protocol used to register the nodes over the LLN is RPL [RFC6550], the BBR sends a targeted DIS to the registered address over the registered path. A DAO back indicates that the current registration is still valid and provides the adequate data to resolve the conflict.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [RFC6775].

6. Security Considerations

This specification expects that the link layer is sufficiently protected, either by means of physical or IP security for the Backbone Link or MAC sublayer cryptography. In particular, it is expected that the LLN MAC provides secure unicast to/from the Backbone Router and secure BBroadcast from the Backbone Router in a way that prevents tempering with or replaying the RA messages.

The use of EUI-64 for forming the Interface ID in the link local address prevents the usage of Secure ND ([RFC3971] and [RFC3972]) and address privacy techniques. Considering the envisioned deployments and the MAC layer security applied, this is not considered an issue at this time.

7. IANA Considerations

A new type is requested for an ND option.

8. Acknowledgments

TBD

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.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.
- [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.
- [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.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [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.
- [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.

9.2. Informative References

- [I-D.chakrabarti-nordmark-6man-efficient-nd]
Chakrabarti, S., Nordmark, E., and M. Wasserman,
"Efficiency aware IPv6 Neighbor Discovery Optimizations",
draft-chakrabarti-nordmark-6man-efficient-nd-01 (work in
progress), November 2012.
- [I-D.ietf-roll-terminology]
Vasseur, J., "Terminology in Low power And Lossy
Networks", draft-ietf-roll-terminology-11 (work in
progress), February 2013.
- [I-D.van-beijnum-multi-mtu]
Beijnum, I., "Extensions for Multi-MTU Subnets",
draft-van-beijnum-multi-mtu-03 (work in progress),
July 2010.
- [RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P.
Thubert, "Network Mobility (NEMO) Basic Support Protocol",
RFC 3963, January 2005.
- [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.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery
Proxies (ND Proxy)", RFC 4389, April 2006.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6
over Low-Power Wireless Personal Area Networks (6LoWPANs):
Overview, Assumptions, Problem Statement, and Goals",
RFC 4919, August 2007.

9.3. External Informative References

- [HART] www.hartcomm.org, "Highway Addressable Remote Transducer,
a group of specifications for industrial process and
control devices administered by the HART Foundation".
- [ISA100.11a]
ISA, "ISA100, Wireless Systems for Automation", May 2008,
<[http://www.isa.org/Community/
SP100WirelessSystemsforAutomation](http://www.isa.org/Community/SP100WirelessSystemsforAutomation)>.

Author's Address

Pascal Thubert
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

Phone: +33 4 97 23 26 34
Email: pthubert@cisco.com

