

IPv6 Maintenance
Internet-Draft
Updates: 2460,7045 (if approved)
Intended status: Standards Track
Expires: April 8, 2016

F. Baker
Cisco Systems
October 6, 2015

IPv6 Hop-by-Hop Header Handling
draft-baker-6man-hbh-header-handling-03

Abstract

This note updates the IPv6 Specification (RFC 2460), specifically commenting on the Hop-by-Hop Options Header (section 4.3) and option format and handling (section 4.2).

It also updates RFC 7045, which noted that RFC 2460 is widely violated in this respect, but merely legitimized this situation with a SHOULD. The present document tries to address the issue more fundamentally.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 8, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Handling of options in extension headers	3
2.1. Hop-by_hop Options	3
2.2. Changing options in transit	4
2.3. Adding headers or options in transit	5
2.4. Interactions with the Security Extension Header	5
3. Interoperation with RFC 2460	5
4. IANA Considerations	6
5. Security Considerations	6
6. Privacy Considerations	6
7. Acknowledgements	7
8. References	7
8.1. Normative References	7
8.2. Informative References	7
Appendix A. Change Log	8
Author's Address	9

1. Introduction

The IPv6 Specification [RFC2460] specifies a number of extension headers. These, and the ordering considerations given, were defined based on experience with IPv4 options. They were, however, prescient with respect to their actual use - the IETF community did not know how they would be used. In at least one case, the Hop-by-Hop option, most if not all implementations implement it by punting to a software path. In the words of [RFC7045],

The IPv6 Hop-by-Hop Options header SHOULD be processed by intermediate forwarding 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.

Fernando Gont, in his Observations on IPv6 EH Filtering in the Real World [I-D.ietf-v6ops-ipv6-ehs-in-real-world], and the operational community in IPv6 Operations, consider any punt to a software path to be an attack vector. Hence, IPv6 packets containing the Hop-by-Hop Extension Header (and in some cases, any extension header) get dropped in transit.

The subject of this document is implementation approaches to obviate or mitigate the attack vector, and updating the Hop-by-Hop option with respect to current issues.

1.1. Requirements Language

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

2. Handling of options in extension headers

Packets containing the Hop-by-Hop Extension Header SHOULD be processed at substantially the same rate as packets that do not.

If a hop-by-hop header option is not implemented, or is not in use, in a given system (such as, for example, an interface that is not configured for RSVP receiving an RSVP Alert Option), the option MUST be skipped.

If a hop-by-hop header is present in a packet's extension header chain and it is not the first extension header, the packet MUST be discarded by the first system that observes the fact (Section 2.2 of [RFC7045]). This will normally be in the system using the IPv6 address in the Destination Address, as [RFC2460] precludes other routers from parsing the header chain. The only obvious exception to that is a router or firewall configured to parse the IPv6 header chain.

2.1. Hop-by_hop Options

At this writing, there are several defined Hop-by-Hop options:

PAD Options: The PAD1 and PADn options [RFC2460] define empty space.

Router Alert Option: The IPv6 Router Alert Option [RFC2711] [RFC6398] is intended to force the punting of a datagram to software, in cases in which RSVP or other protocols need that to happen.

Jumbo Payload: Carries a length field for a packet whose length exceeds 0xFFFF octets. [RFC2675]

RPL Option: The RPL option carries routing information used in a RPL network[RFC6553]

Quickstart Option Identifies TCP quick-start configuration, and allows an intermediate router to reduce the configuration parameters as appropriate. [RFC4782]

Common Architecture Label IPv6 Security Option: Encodes security labels on packets [RFC5570]

SMF Option: Simplified Multicast Forwarding Option[RFC6621]

MPL Option: Supports multicast in an RPL network [I-D.ietf-roll-trickle-mcast]

DFF Option: Depth-First Forwarding [RFC6971]

There are also options that have been defined for the Destination Options header. These are not listed here.

While this is not true of older implementations, modern equipment is capable of parsing the Extension Header chain, and can be extended to perform at least a cursory examination of the Hop-by-Hop options. For example, such implementations SHOULD be able to identify and skip the PAD1 and PADn options, and perform more complicated processing only if configured by software to do so. More to the point: it isn't clear what the purpose of the JumboFrame option is if not to be understood by anyone that looks at it.

Question asked by a reviewer: "Is this configurable? How will router know that HbH needs to be skipped on one interface and not on others."

Answer: the system knows whether RSVP has been configured on an interface. When such configuration is present, it can configure the hardware with what it wants done with the Router Alert. In the absence of such configuration, hardware should be configured to skip the option if found.

2.2. Changing options in transit

Section 4.2 of [RFC2460] explicitly allows for options that may be updated in transit. It is likely that the original authors intended that to be very simple, such as having the originating end system provide the container, and having intermediate systems update it - perhaps performing some calculation, and in any event storing the resulting value. Examples of such a use might be in [XCP] or [RCP].

As a side comment, the Routing Header, which is an extension header rather than a list of options, is treated similarly; when a system is the destination of a packet and not the last one in the Routing

Header's list, it swaps the destination address with the indicated address in the list, and updates the hop count and the list depth accordingly.

Such options must be marked appropriately (their option type is of the form XX1XXXXX), and are excluded from checksum calculations in AH and ESP.

2.3. Adding headers or options in transit

Use cases under current consideration take this a step further: a router or middleware process MAY add an extension header, MAY add an option to the header, which may extend the length of the Hop-by-Hop Extension Header, or MAY process such an option in a manner that extends both the length of the option and the Extension Header containing it. The obvious implication is that other equipment in the network may not understand or implement the new option type. As such, the Option Type value of such an option MUST indicate that it is to be skipped by a system that does not understand it. Since, by definition, it is being updated in transit and not included in any AH or ESP integrity check if present, the Option Type MUST also indicate that it may be updated in transit, and so is excluded from AH and ESP processing. By implication, such an Option Type MUST be of the form 001XXXXX.

2.4. Interactions with the Security Extension Header

The interactions with the IP Authentication Header [RFC4302] and IP Encapsulating Security Payload (ESP) [RFC4303], as in the case of existing option uses, is minimally defined. AH and ESP call for the exclusion of mutable data in their calculations by zeroing it out prior to performing the integrity check calculation. However, in the case that network operation has changed the length of the option or the extension header, that may still cause the integrity check to fail. Specifications that define such options SHOULD consider the implications of this for AH and ESP. An option whose insertion would affect the integrity check MUST be removed prior to the integrity check, and as a result the packet restored to its state as originally sent.

3. Interoperation with RFC 2460

There are four possible modes of interaction with routers that don't implement the Hop-By-Hop Option in the fast path:

1. Presume that they cannot handle the Hop-By-Hop option at close to wire speed, and that's OK.

2. Presume that they will drop traffic containing Hop-By-Hop options.
3. Presume that they can handle the Hop-By-Hop option at or close to wire speed, and are configured to do so.
4. Presume that they don't exist, perhaps because older routers are configured to ignore all Hop-by-Hop options.

If the first model actually works in a given network, it may be acceptable in that domain. It is not a model that will work in the general Internet, however.

The second model (which is most probable at this writing) is a description of the general Internet in 2015.

The third and fourth models, if applicable in a given context, are what one might hope for. Vendors are in a position to either have an option to ignore the Hop-By-Hop header in older equipment, or add such an option in upgraded software (fourth model). New equipment is expected to follow the third model by implementing the recommendations in Section 2.

4. IANA Considerations

This memo asks the IANA for no new parameters.

5. Security Considerations

In general, modification of a datagram in transit is considered very closely from the viewpoint of the End-to-End Principle, which in this context may be summarized as "the network should do nothing that is of concern to the communicating applications or introduces operational issues." The concept of changing the length of an Extension Header or an option contained within it (Section 2.3) is of concern in that context. The obvious concern is around the interaction with AH or ESP, and a less obvious concern relates to Path MTU, which might change if the size of an underlying header changes. Section 2.4 is intended to mitigate that issue. However, some ramifications, such as with Path MTU, may not be completely solvable in the general Internet, but require use cases to be confined to a network or set of consenting networks.

6. Privacy Considerations

Data formats in this memo reveal no personally identifying information.

7. Acknowledgements

This note grew out of a discussion among the author, Ole Troan, Mark Townsley, Frank Brockners, and Shwetha Bhandari, and benefited from comments by Dennis Ferguson, Brian Carpenter, Panos Kampanakis, JINMEI Tatuya, and Joe Touch.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

8.2. Informative References

- [I-D.ietf-roll-trickle-mcast] Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-12 (work in progress), June 2015.
- [I-D.ietf-v6ops-ipv6-ehs-in-real-world] Gont, F., Linkova, J., Chown, T., and S. LIU, "Observations on IPv6 EH Filtering in the Real World", draft-ietf-v6ops-ipv6-ehs-in-real-world-00 (work in progress), April 2015.
- [RCP] Dukkupati, N., "Rate Control Protocol (RCP): Congestion control to make flows complete quickly", Stanford University, 2006.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<http://www.rfc-editor.org/info/rfc2675>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<http://www.rfc-editor.org/info/rfc2711>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <<http://www.rfc-editor.org/info/rfc4782>>.
- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<http://www.rfc-editor.org/info/rfc5570>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<http://www.rfc-editor.org/info/rfc6398>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <<http://www.rfc-editor.org/info/rfc6621>>.
- [RFC6971] Herberg, U., Ed., Cardenas, A., Iwao, T., Dow, M., and S. Cespedes, "Depth-First Forwarding (DFF) in Unreliable Networks", RFC 6971, DOI 10.17487/RFC6971, June 2013, <<http://www.rfc-editor.org/info/rfc6971>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.
- [XCP] Katabi, D., Handley, M., and C. Rohrs, "Congestion control for high bandwidth-delay product networks", SIGCOMM Symposium proceedings on Communications architectures and protocols , 2002.

Appendix A. Change Log

Initial Version: June 2015

01 Version: June 2015, responding to list discussion

Internet-Draft

October 2015

02 Version: July 2015, discussed at IETF 93

03 Version: October 2015

Author's Address

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: 2464, 2467, 2470, 2491, 2492,
2497, 2590, 3146, 3572, 4291,
4338, 4391, 5072, 5121 (if
approved)
Intended status: Standards Track
Expires: March 29, 2017

F. Gont
SI6 Networks / UTN-FRH
A. Cooper
Cisco
D. Thaler
Microsoft
W. Liu
Huawei Technologies
September 28, 2016

Recommendation on Stable IPv6 Interface Identifiers
draft-ietf-6man-default-iids-16

Abstract

This document changes the recommended default IID generation scheme for cases where SLAAC is used to generate a stable IPv6 address. It recommends using the mechanism specified in RFC7217 in such cases, and recommends against embedding stable link-layer addresses in IPv6 Interface Identifiers. It formally updates RFC2464, RFC2467, RFC2470, RFC2491, RFC2492, RFC2497, RFC2590, RFC3146, RFC3572, RFC4291, RFC4338, RFC4391, RFC5072, and RFC5121. This document does not change any existing recommendations concerning the use of temporary addresses as specified in RFC 4941.

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 February 21, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	4
3. Generation of IPv6 Interface Identifiers with SLAAC	4
4. Future Work	4
5. IANA Considerations	5
6. Security Considerations	5
7. Acknowledgements	5
8. References	5
Authors' Addresses	8

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) [RFC4291] that typically embeds a stable link-layer address (e.g., an IEEE LAN MAC address).

In some network technologies and adaptation layers, the use of an IID based on a link-layer address may offer some advantages. For example, the IP-over-IEEE802.15.4 standard in [RFC6775] allows for compression of IPv6 addresses when the IID is based on the underlying link-layer address.

The security and privacy implications of embedding a stable link-layer address in an IPv6 IID have been known for some time now, and are discussed in great detail in [RFC7721]. They include:

- o Network activity correlation
- o Location tracking
- o Address scanning
- o Device-specific vulnerability exploitation

More generally, the reuse of identifiers that have their own semantics or properties across different contexts or scopes can be detrimental for security and privacy [I-D.gont-predictable-numeric-ids]. In the case of traditional stable IPv6 IIDs, some of the security and privacy implications are dependent on the properties of the underlying link-layer addresses (e.g., whether the link-layer address is ephemeral or randomly generated), while other implications (e.g., reduction of the entropy of the IID) depend on the algorithm for generating the IID itself. In standardized recommendations for stable IPv6 IID generation meant to achieve particular security and privacy properties, it is therefore necessary to recommend against embedding stable link-layer addresses in IPv6 IIDs.

Furthermore, some popular IPv6 implementations have already deviated from the traditional stable IID generation scheme to mitigate the aforementioned security and privacy implications [Microsoft].

As a result of the aforementioned issues, this document changes the recommended default IID generation scheme for generating stable IPv6 addresses with SLAAC to that specified in [RFC7217], and recommends against embedding stable link-layer addresses in IPv6 Interface Identifiers, such that the aforementioned issues are mitigated. That is, this document simply replaces the default algorithm that is recommended to be employed when generating stable IPv6 IIDs.

NOTE: [RFC4291] defines the "Modified EUI-64 format" for IIDs. Appendix A of [RFC4291] then describes how to transform an IEEE EUI-64 identifier, or an IEEE 802 48-bit MAC address from which an EUI-64 identifier is derived, into an IID in the Modified EUI-64 format.

In a variety of scenarios, addresses that remain stable for the lifetime of a host's connection to a single subnet, are viewed as desirable. For example, stable addresses may be viewed as beneficial for network management, event logging, enforcement of access control, provision of quality of service, or for server or routing interfaces. Similarly, stable addresses (as opposed to temporary addresses [RFC4941]) allow for long-lived TCP connections, and are also usually desirable when performing server-like functions (i.e., receiving incoming connections).

The recommendations in this document apply only in cases where implementations otherwise would have configured a stable IPv6 IID containing a link layer address. For example, this document does not change any existing recommendations concerning the use of temporary addresses as specified in [RFC4941], nor do the recommendations apply to cases where SLAAC is employed to generate non-stable IPv6

addresses (e.g. by embedding a link-layer address that is periodically randomized), nor does it introduce any new requirements regarding when stable addresses are to be configured. Thus, the recommendations in this document simply improve the security and privacy properties of stable addresses.

2. Terminology

Stable address:

An address that does not vary over time within the same network (as defined in [RFC7721]).

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. Generation of IPv6 Interface Identifiers with SLAAC

Nodes SHOULD implement and employ [RFC7217] as the default scheme for generating stable IPv6 addresses with SLAAC. A link layer MAY also define a mechanism for stable IPv6 address generation that is more efficient and does not address the security and privacy considerations discussed in Section 1. The choice of whether to enable the security- and privacy-preserving mechanism or not SHOULD be configurable in such a case.

By default, nodes SHOULD NOT employ IPv6 address generation schemes that embed a stable link-layer address in the IID. In particular, this document RECOMMENDS that nodes do not generate stable IIDs with the schemes specified in [RFC2464], [RFC2467], [RFC2470], [RFC2491], [RFC2492], [RFC2497], [RFC2590], [RFC3146], [RFC3572], [RFC4338], [RFC4391], [RFC5121], and [RFC5072].

4. Future Work

At the time of this writing, the mechanisms specified in the following documents might require updates to be fully compatible with the recommendations in this document:

- o "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks" [RFC6282]
- o "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944]
- o "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)" [RFC6775]

- o "Transmission of IPv6 Packets over ITU-T G.9959 Networks" [RFC7428]

Future revisions or updates of these documents should take the issues of privacy and security mentioned in Section 1 and explain any design and engineering considerations that lead to the use of stable IIDs based on a node's link-layer address.

5. IANA Considerations

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

6. Security Considerations

This recommends against the (default) use of predictable Interface Identifiers in IPv6 addresses. It recommends [RFC7217] as the default scheme for generating IPv6 stable addresses with SLAAC, such that the security and privacy issues of IIDs that embed stable link-layer addresses are mitigated.

7. Acknowledgements

The authors would like to thank (in alphabetical order) Bob Hinden, Ray Hunter and Erik Nordmark, for providing a detailed review of this document.

The authors would like to thank (in alphabetical order) Fred Baker, Carsten Bormann, Scott Brim, Brian Carpenter, Samita Chakrabarti, Tim Chown, Lorenzo Colitti, Jean-Michel Combes, Greg Daley, Esko Dijk, Ralph Droms, David Farmer, Brian Haberman, Ulrich Herberg, Philip Homburg, Jahangir Hossain, Jonathan Hui, Christian Huitema, Ray Hunter, Erik Kline, Sheng Jiang, Roger Jorgensen, Dan Luedtke, Kerry Lynn, George Mitchel, Gabriel Montenegro, Erik Nordmark, Simon Perreault, Tom Petch, Alexandru Petrescu, Michael Richardson, Arturo Servin, Mark Smith, Tom Taylor, Ole Troan, Tina Tsou, Glen Turner, Randy Turner, James Woodyatt, and Juan Carlos Zuniga, for providing valuable comments on earlier versions of this document.

8. References

8.1. Normative References

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

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<http://www.rfc-editor.org/info/rfc2464>>.
- [RFC2467] Crawford, M., "Transmission of IPv6 Packets over FDDI Networks", RFC 2467, DOI 10.17487/RFC2467, December 1998, <<http://www.rfc-editor.org/info/rfc2467>>.
- [RFC2470] Crawford, M., Narten, T., and S. Thomas, "Transmission of IPv6 Packets over Token Ring Networks", RFC 2470, DOI 10.17487/RFC2470, December 1998, <<http://www.rfc-editor.org/info/rfc2470>>.
- [RFC2491] Armitage, G., Schuler, P., Jork, M., and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, DOI 10.17487/RFC2491, January 1999, <<http://www.rfc-editor.org/info/rfc2491>>.
- [RFC2492] Armitage, G., Schuler, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <<http://www.rfc-editor.org/info/rfc2492>>.
- [RFC2497] Souvatzis, I., "Transmission of IPv6 Packets over ARCnet Networks", RFC 2497, DOI 10.17487/RFC2497, January 1999, <<http://www.rfc-editor.org/info/rfc2497>>.
- [RFC2590] Conta, A., Malis, A., and M. Mueller, "Transmission of IPv6 Packets over Frame Relay Networks Specification", RFC 2590, DOI 10.17487/RFC2590, May 1999, <<http://www.rfc-editor.org/info/rfc2590>>.
- [RFC3146] Fujisawa, K. and A. Onoe, "Transmission of IPv6 Packets over IEEE 1394 Networks", RFC 3146, DOI 10.17487/RFC3146, October 2001, <<http://www.rfc-editor.org/info/rfc3146>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4338] DeSanti, C., Carlson, C., and R. Nixon, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel", RFC 4338, DOI 10.17487/RFC4338, January 2006, <<http://www.rfc-editor.org/info/rfc4338>>.

- [RFC4391] Chu, J. and V. Kashyap, "Transmission of IP over InfiniBand (IPoIB)", RFC 4391, DOI 10.17487/RFC4391, April 2006, <<http://www.rfc-editor.org/info/rfc4391>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<http://www.rfc-editor.org/info/rfc4944>>.
- [RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, DOI 10.17487/RFC5072, September 2007, <<http://www.rfc-editor.org/info/rfc5072>>.
- [RFC5121] Patil, B., Xia, F., Sarikaya, B., Choi, JH., and S. Madanapalli, "Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks", RFC 5121, DOI 10.17487/RFC5121, February 2008, <<http://www.rfc-editor.org/info/rfc5121>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<http://www.rfc-editor.org/info/rfc6282>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.

- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <<http://www.rfc-editor.org/info/rfc7428>>.

8.2. Informative References

- [I-D.gont-predictable-numeric-ids]
Gont, F. and I. Arce, "Security and Privacy Implications of Numeric Identifiers Employed in Network Protocols", draft-gont-predictable-numeric-ids-00 (work in progress), February 2016.
- [Microsoft]
Davies, J., "Understanding IPv6, 3rd. ed", page 83, Microsoft Press, 2012, <<http://it-ebooks.info/book/1022/>>.
- [RFC3572] Ogura, T., Maruyama, M., and T. Yoshida, "Internet Protocol Version 6 over MAPOS (Multiple Access Protocol Over SONET/SDH)", RFC 3572, DOI 10.17487/RFC3572, July 2003, <<http://www.rfc-editor.org/info/rfc3572>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<http://www.rfc-editor.org/info/rfc7721>>.

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>

Alissa Cooper
Cisco
707 Tasman Drive
Milpitas, CA 95035
US

Phone: +1-408-902-3950
Email: alcoop@cisco.com
URI: <https://www.cisco.com/>

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

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

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

Email: liushucheng@huawei.com

IPv6 Maintenance
Internet-Draft
Updates: 2460,7045 (if approved)
Intended status: Standards Track
Expires: September 17, 2016

F. Baker
Cisco Systems
R. Bonica
Juniper Networks
March 16, 2016

IPv6 Hop-by-Hop Options Extension Header
draft-ietf-6man-hbh-header-handling-03

Abstract

This document clarifies requirements for IPv6 routers with respect to the Hop-by-Hop (HBH) Options Extension Header. These requirements are applicable to all IPv6 routers, regardless of whether they maintain a strict separation between forwarding and control plane hardware. In this respect, this document updates RFC 2460 and RFC 7045.

This document also describes forwarding plane procedures for processing the HBH Options Extension Header. These procedures are applicable to implementations that maintain a strict separation between forwarding and control plane implementations.

The procedures described herein satisfy the above mentioned requirements by processing HBH Options on the forwarding plane to the greatest degree possible. If a packet containing HBH Options must be dispatched to the control plane, it is rate limited before dispatching. In order to comply with the requirements of this specification, implementations may execute the procedures described herein or any other procedures that result in compliant behavior.

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 September 17, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	4
2. Requirements	4
3. Proposed Procedures	6
4. IANA Considerations	7
5. Security Considerations	7
6. Acknowledgements	7
7. References	7
7.1. Normative References	7
7.2. Informative References	8
Appendix A. Change Log	9
Appendix B. HBH Options	10
Authors' Addresses	10

1. Introduction

In IPv6 [RFC2460], optional Internet-layer information is encoded in extension headers that may be placed between the IPv6 header and the upper-layer header. Currently, eleven extension headers are defined. Among them is the Hop-by-Hop (HBH) Options Extension header. Unlike any other extension header, the HBH Options Extension header is examined by every node that a packet visits en route to its destination.

The HBH Extension Header contains one or more HBH Options. Each HBH Option contains a type identifier. Appendix B of this document provides a list of currently defined HBH options.

Some HBH Options contain information that is useful to a router's forwarding plane. In this document, we call these options "HBH forwarding options". Among these is the Jumbo Payload Option

[RFC2675]. The Jumbo Payload Option indicates the payload length of the packet that carries it. While this information is required to forward the packet, it can be discarded as soon as the packet has been forwarded.

By contrast, other HBH Options contain information that is useful to a router's control plane. In this document, we call these options "HBH control options". Among these is the Router Alert Option [RFC2711]. The Router Alert Option informs transit routers that the packet carrying it contains information to be consumed by the router's control plane. In many cases, this information is used to forward subsequent packets.

Finally, the Pad and Pad1 options contain no information at all. These are included to ensure word-alignment of subsequent options and headers.

Many modern routers maintain a strict separation between forwarding plane hardware and control plane hardware. In these routers, forwarding plane bandwidth is plentiful, while control plane bandwidth is constrained. In order to protect scarce control plane resources, these routers enforce policies that restrict access from the forwarding plane to the control plane. Effective policies address packets containing the HBH Options Extension header, because HBH control options require access from the forwarding plane to the control plane.

Many network operators perceive HBH Options to be a breach of the separation between the forwarding and control planes [I-D.ietf-v6ops-ipv6-ehs-in-real-world]. Therefore, some network operators discard all packets containing the HBH Options Extension Header, while others forward the packets but ignore the HBH Options. Still other operators severely rate-limit packets containing the HBH Options Extension Header. In addition, some (notably older) implementations send all packets containing a HBH header to the control plane even if they contain only pad options, resulting in an effect DoS on the router and inconsistent drops among those packets due to rate limiting or other factors.

[RFC7045] legitimizes the current state of affairs, severely limiting the utility of HBH options. In the words of RFC 7045:

"The IPv6 Hop-by-Hop Options header SHOULD be processed by intermediate forwarding 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."

This document clarifies requirements for IPv6 routers with respect to the HBH Options Extension Header. These requirements are applicable to all IPv6 routers, regardless of whether they maintain a strict separation between forwarding and control plane hardware. In this respect, this document updates RFC 2460 and RFC 7045.

This document also describes forwarding plane procedures for processing the HBH Options Extension Header. These procedures are applicable to implementations that maintain a strict separation between forwarding and control plane hardware.

The procedures described herein satisfy the above mentioned requirements by processing HBH Options on the forwarding plane to the greatest degree possible. If a packet containing HBH Options must be dispatched to the control plane, it is rate limited before dispatching. In order to comply with the requirements of this specification, implementations can execute the procedures described herein or any other procedures that result in compliant behavior.

1.1. Requirements Language

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

2. Requirements

This section clarifies requirements for IPv6 routers with respect to the HBH Options Extension Header. These requirements are applicable to all IPv6 routers, regardless of whether they maintain a strict separation between forwarding and control plane hardware.

- o REQ1: Implementations MUST NOT discard otherwise forwardable packets because they contain the HBH Options Extension header. However, an implementation MAY be configured to discard packets containing the HBH Options Extension Header, so long as this is not the default behavior.
- o REQ 2: Implementations MUST process unrecognized HBH Options as described in Section 4.2 of RFC 2460. If an implementation receives a packet that contains an unrecognized HBH Option, that implementation MUST examine the first two bits of the HBH Option Type indicator. Those bits determine whether the implementation a) continues to process the packet, b) discards the packet without sending an ICMP message or c) discards the packet and sends an ICMP message.

- o REQ 3: Unrecognized HBH Options MUST be evaluated sequentially. For example, assume that an implementation receives a packet that carries two unrecognized HBH Options. The Type indicator of the first unrecognized option begins with 01 while the Type indicator of the second unrecognized option begins with 10. In this case, the implementation MUST discard the packet without sending an ICMP message to the source. However, if the Type indicator of the first unrecognized option begins with 10 and the Type indicator of the second unrecognized option begins with 01, the implementation MUST discard the packet and send an ICMP Parameter Problem message to the source.
- o REQ 4: Implementations MUST protect themselves against denial of service attacks that are propagated through HBH Options. These protections MUST be enabled by default, without special configuration.
- o REQ 5: The originator of a packet MAY insert the HBH Options Extension header between the IPv6 header and the upper-layer header. It MAY also insert HBH Options inside of the HBH Options header. Transit routers MUST NOT insert the HBH Options Extension header between the IPv6 header and the upper-layer header. Furthermore, they MUST NOT add or delete HBH Options inside of the HBH Options Extension header.
- o REQ 6: Implementations SHOULD support a configuration option that limits the set of HBH Options that they recognize. For example, assume that an implementation recognizes a particular HBH Option. Using this configuration option, an operator can cause the implementation to behave as if it does not recognize that option. This MAY be configured as a side effect of other functionality. For example, an implementation might not recognize the Router Alert Option unless a protocol that relies on the Router Alert Option (e.g., RSVP) is configured.
- o REQ 7: The HBH Options Extension Header can contain as many as 2056 bytes. Some implementations are not capable of processing extension headers of that length [I-D.gont-v6ops-ipv6-ehs-packet-drops]. When an implementation receives a packet that it cannot process due to its HBH Options Extension Header length, the implementation MUST discard the packet and send an ICMP Parameter Problem message to the packet source. ICMP Parameter Problem Code MUST be "Long Extension Header" (value TBD) and the ICMP Parameter Problem Pointer MUST contain the offset of HBH Options Extension Header.

3. Proposed Procedures

This section describes forwarding plane procedures for processing the HBH Options Extension Header. These procedures are applicable to implementations that maintain a strict separation between forwarding and control plane hardware.

The procedures described below process HBH Options on the forwarding plane to the greatest degree possible. If a packet containing HBH Options must be dispatched to the control plane, it is rate limited before dispatching. In order to comply with the requirements of Section 2, implementations can execute the procedures described herein or any other procedures that result in compliant behavior.

Having received a packet containing the HBH Options Extension header, the forwarding plane determines whether the HBH Options Extension Header is too long for it to process. If so, the forwarding plane discards the packet and sends an ICMP Parameter Problem message to the packet source. ICMP Parameter Problem Code is set to "Long Extension Header" and the ICMP Parameter Problem Pointer is set to the offset of HBH Options Extension Header.

If the HBH Options Extension Header is not too long to process, the forwarding plane hardware scans the header, assigning it to one of the following classes:

- o Discard
- o Dispatch to control plane
- o Forward, ignoring all HBH Option
- o Forward, processing selected HBH Options

Forwarding plane hardware discards the packet if the HBH Options Extension Header contains an unrecognized option whose Type indicator begins with 01, 10 or 11. Forwarding plane hardware sends an ICMP message if required. See Section 2 REQ 2 and REQ 3 for details.

If the packet is not discarded, and the HBH Options Extension header contains at least one recognized control option, the forwarding plane subjects the packet to a rate-limit and dispatches it to the control plane

Otherwise, if the HBH Options Extension header contains only the following option types, the packet is forwarded without further HBH Option processing:

- o Pad or Pad1
- o Unrecognized options whose Type indicator begins with 00

Otherwise, the forwarding plane process forwarding options and forwards the packet

4. IANA Considerations

IANA is requested to assign a new entry to the ICMP Parameter Problem Code registry. The name of this code is "Long Extension Header".

5. Security Considerations

This document contributes to the security of IPv6 routers, by defining forwarding plane procedures for the processing of HBH Options. These procedures are applicable to implementations that maintain a strict separation between forwarding and control plane hardware.

The procedures described below process HBH Options on the forwarding plane to the greatest degree possible. If a packet containing HBH Options must be dispatched to the control plane, it is rate limited before dispatching.

6. Acknowledgements

This note grew out of a discussion among the author, Ole Troan, Mark Townsley, Frank Brockners, and Shwetha Bhandari, and benefited from comments by Dennis Ferguson, Brian Carpenter, Panos Kampanakis, Jinmei Tatuya, and Joe Touch. Thanks to Fernando Gont for his thoughtful review.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.

7.2. Informative References

- [I-D.gont-v6ops-ipv6-ehs-packet-drops]
Gont, F., Hilliard, N., Doering, G., LIU, S., and W. Kumari, "Operational Implications of IPv6 Packets with Extension Headers", draft-gont-v6ops-ipv6-ehs-packet-drops-03 (work in progress), March 2016.
- [I-D.ietf-6man-rfc2460bis]
Deering, S. and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-03 (work in progress), January 2016.
- [I-D.ietf-roll-trickle-mcast]
Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-12 (work in progress), June 2015.
- [I-D.ietf-v6ops-ipv6-ehs-in-real-world]
Gont, F., Linkova, J., Chown, T., and S. LIU, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", draft-ietf-v6ops-ipv6-ehs-in-real-world-02 (work in progress), December 2015.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<http://www.rfc-editor.org/info/rfc2675>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<http://www.rfc-editor.org/info/rfc2711>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.

- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <<http://www.rfc-editor.org/info/rfc4782>>.
- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<http://www.rfc-editor.org/info/rfc5570>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<http://www.rfc-editor.org/info/rfc6398>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <<http://www.rfc-editor.org/info/rfc6621>>.
- [RFC6971] Herberg, U., Ed., Cardenas, A., Iwao, T., Dow, M., and S. Cespedes, "Depth-First Forwarding (DFF) in Unreliable Networks", RFC 6971, DOI 10.17487/RFC6971, June 2013, <<http://www.rfc-editor.org/info/rfc6971>>.

Appendix A. Change Log

RFC Editor: this section need not be published in any RFC.

Initial Version: October 2015: text copied from draft-baker-6man-hbh-header-handling-03.txt and discussed in IETF 94

IETF 94 Update: Sections 2.2, 2.3, and 2.4 moved to an appendix reflecting (negative) working group viewpoint on the modification of packet length in flight.

The content of this document is likely to be subsumed into 2460bis [I-D.ietf-6man-rfc2460bis], but is held separate for the present discussion.

A new section 2.2 added detailing conceptual processing model for HBH options.

version 2 Addressed editorial comments

Appendix B. HBH Options

At this writing, there are several defined Hop-by-Hop options:

PAD Options: The PAD1 and PADn [RFC2460]

Router Alert Option: The IPv6 Router Alert Option [RFC2711]
[RFC6398]

Jumbo Payload: [RFC2675]

RPL Option: [RFC6553]

Quickstart Option [RFC4782]

Common Architecture Label IPv6 Security Option: [RFC5570]

SMF Option: [RFC6621]

MPL Option: [I-D.ietf-roll-trickle-mcast]

DFF Option: [RFC6971]

Authors' Addresses

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Ron Bonica
Juniper Networks
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Network Working Group
Internet-Draft
Obsoletes: 1981 (if approved)
Intended status: Standards Track
Expires: November 28, 2017

J. McCann
Digital Equipment Corporation
S. Deering
Retired
J. Mogul
Digital Equipment Corporation
R. Hinden, Ed.
Check Point Software
May 27, 2017

Path MTU Discovery for IP version 6
draft-ietf-6man-rfc1981bis-08

Abstract

This document describes Path MTU Discovery for IP version 6. It is largely derived from RFC 1191, which describes Path MTU Discovery for IP version 4. It obsoletes RFC1981.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 28, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Protocol Overview	5
4. Protocol Requirements	6
5. Implementation Issues	7
5.1. Layering	7
5.2. Storing PMTU information	8
5.3. Purging stale PMTU information	10
5.4. Packetization layer actions	11
5.5. Issues for other transport protocols	12
5.6. Management interface	12
6. Security Considerations	13
7. Acknowledgements	13
8. IANA Considerations	14
9. References	14
9.1. Normative References	14
9.2. Informative References	14
Appendix A. Comparison to RFC 1191	15
Appendix B. Changes Since RFC 1981	16
B.1. Change History Since RFC1981	17
Authors' Addresses	21

1. Introduction

When one IPv6 node has a large amount of data to send to another node, the data is transmitted in a series of IPv6 packets. These packets can have a size less than or equal to the Path MTU (PMTU). Alternatively, they can be larger packets that are fragmented into a series of fragments each with a size less than or equal to the PMTU.

It is usually preferable that these packets be of the largest size that can successfully traverse the path from the source node to the destination node without the need for IPv6 fragmentation. This packet size is referred to as the Path MTU, and it is equal to the minimum link MTU of all the links in a path. This document defines a standard mechanism for a node to discover the PMTU of an arbitrary path.

IPv6 nodes should implement Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU [I-D.ietf-6man-rfc2460bis]. A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery.

Nodes not implementing Path MTU Discovery must use the IPv6 minimum link MTU defined in [I-D.ietf-6man-rfc2460bis] as the maximum packet size. In most cases, this will result in the use of smaller packets than necessary, because most paths have a PMTU greater than the IPv6 minimum link MTU. A node sending packets much smaller than the Path MTU allows is wasting network resources and probably getting suboptimal throughput.

Nodes implementing Path MTU Discovery and sending packets larger than the IPv6 minimum link MTU are susceptible to problematic connectivity if ICMPv6 [ICMPv6] messages are blocked or not transmitted. For example, this will result in connections that complete the TCP three-way handshake correctly but then hang when data is transferred. This state is referred to as a black hole connection [RFC2923]. Path MTU Discovery relies on ICMPv6 Packet Too Big (PTB) to determine the MTU of the path.

An extension to Path MTU Discovery defined in this document can be found in [RFC4821]. RFC4821 defines a method for Packetization Layer Path MTU Discovery (PLPMTUD) designed for use over paths where delivery of ICMPv6 messages to a host is not assured.

Note: This document is an update to [RFC1981] that was published prior to [RFC2119] being published. Consequently although RFC1981 used the "should/must" style language in upper and lower case, this document does not cite the RFC2119 definitions and only uses lower case for these words.

2. Terminology

node	a device that implements IPv6.
router	a node that forwards IPv6 packets not explicitly addressed to itself.

host	any node that is not a router.
upper layer	a protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMPv6, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.
link	a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.
interface	a node's attachment to a link.
address	an IPv6-layer identifier for an interface or a set of interfaces.
packet	an IPv6 header plus payload. The packet can have a size less than or equal to the PMTU. Alternatively, this can be a larger packet that is fragmented into a series of fragments each with a size less than or equal to the PMTU.
link MTU	the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed in one piece over a link.
path	the set of links traversed by a packet between a source node and a destination node.
path MTU	the minimum link MTU of all the links in a path between a source node and a destination node.
PMTU	path MTU
Path MTU Discovery	process by which a node learns the PMTU of a path
EMTU_S	Effective MTU for sending, used by upper layer protocols to limit the size of IP packets they queue for sending [RFC6691] [RFC1122].

EMTU_R	Effective MTU for receiving, the largest packet that can be reassembled at the receiver [RFC1122].
flow	a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.
flow id	a combination of a source address and a non-zero flow label.

3. Protocol Overview

This memo describes a technique to dynamically discover the PMTU of a path. The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message. The decreased PMTU causes the source to send smaller packets or change EMTU_S to cause upper layer to reduce the size of IP packets it sends.

The Path MTU Discovery process ends when the source node's estimate of the PMTU is less than or equal to the actual PMTU. Note that several iterations of the packet-sent/Package-Too-Big-message-received cycle may occur before the Path MTU Discovery process ends, as there may be links with smaller MTUs further along the path.

Alternatively, the node may elect to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU.

The PMTU of a path may change over time, due to changes in the routing topology. Reductions of the PMTU are detected by Packet Too Big messages. To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed. Therefore, attempts to detect increases in a path's PMTU should be done infrequently.

Path MTU Discovery supports multicast as well as unicast destinations. In the case of a multicast destination, copies of a packet may traverse many different paths to many different nodes. Each path may have a different PMTU, and a single multicast packet

may result in multiple Packet Too Big messages, each reporting a different next-hop MTU. The minimum PMTU value across the set of paths in use determines the size of subsequent packets sent to the multicast destination.

Note that Path MTU Discovery must be performed even in cases where a node "thinks" a destination is attached to the same link as itself, it might have a PMTU lower than the link MTU. In a situation such as when a neighboring router acts as proxy [ND] for some destination, the destination can appear to be directly connected but it is in fact more than one hop away.

4. Protocol Requirements

As discussed in Section 1, IPv6 nodes are not required to implement Path MTU Discovery. The requirements in this section apply only to those implementations that include Path MTU Discovery.

Nodes should appropriately validate the payload of ICMPv6 PTB messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to an IPv6 packet actually sent by the application) per [ICMPv6].

If a node receives a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU, it must discard it. A node must not reduce its estimate of the Path MTU below the IPv6 minimum link MTU on receipt of an Packet Too Big message.

When a node receives a Packet Too Big message, it must reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message. The precise behavior of a node in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies.

After receiving a Packet Too Big message, a node must attempt to avoid eliciting more such messages in the near future. The node must reduce the size of the packets it is sending along the path. Using a PMTU estimate larger than the IPv6 minimum link MTU may continue to elicit Packet Too Big messages. Because each of these messages (and the dropped packets they respond to) consume network resources, Nodes using Path MTU Discovery must detect decreases in PMTU as fast as possible.

Nodes may detect increases in PMTU, but because doing so requires sending packets larger than the current estimated PMTU, and because the likelihood is that the PMTU will not have increased, this must be done at infrequent intervals. An attempt to detect an increase (by

sending a packet larger than the current estimate) must not be done less than 5 minutes after a Packet Too Big message has been received for the given path. The recommended setting for this timer is twice its minimum value (10 minutes).

A node must not increase its estimate of the Path MTU in response to the contents of a Packet Too Big message. A message purporting to announce an increase in the Path MTU might be a stale packet that has been floating around in the network, a false packet injected as part of a denial-of-service attack, or the result of having multiple paths to the destination, each with a different PMTU.

5. Implementation Issues

This section discusses a number of issues related to the implementation of Path MTU Discovery. This is not a specification, but rather a set of notes provided as an aid for implementers.

The issues include:

- What layer or layers implement Path MTU Discovery?
- How is the PMTU information cached?
- How is stale PMTU information removed?
- What must transport and higher layers do?

5.1. Layering

In the IP architecture, the choice of what size packet to send is made by a protocol at a layer above IP. This memo refers to such a protocol as a "packetization protocol". Packetization protocols are usually transport protocols (for example, TCP) but can also be higher-layer protocols (for example, protocols built on top of UDP).

Implementing Path MTU Discovery in the packetization layers simplifies some of the inter-layer issues, but has several drawbacks: the implementation may have to be redone for each packetization protocol, it becomes hard to share PMTU information between different packetization layers, and the connection-oriented state maintained by some packetization layers may not easily extend to save PMTU information for long periods.

It is therefore suggested that the IP layer store PMTU information and that the ICMPv6 layer process received Packet Too Big messages. The packetization layers may respond to changes in the PMTU by changing the size of the messages they send. To support this

layering, packetization layers require a way to learn of changes in the value of MMS_S, the "maximum send transport-message size" [RFC1122].

MMS_S is a transport message size calculated by subtracting the size of the IPv6 header (including IPv6 extension headers) from the largest IP packet that can be sent, EMTU_S. MMS_S is limited by a combination of factors, including the PMTU, support for packet fragmentation and reassembly, and the packet reassembly limit (see [I-D.ietf-6man-rfc2460bis] section "Fragment Header"). When source fragmentation is available, EMTU_S is set to EMTU_R, as indicated by the receiver using an upper layer protocol or based on protocol requirements (1500 octets for IPv6). When a message larger than PMTU is to be transmitted, the source creates fragments, each limited by PMTU. When source fragmentation is not desired, EMTU_S is set to PMTU, and the upper layer protocol is expected to either perform its own fragmentation and reassembly or otherwise limit the size of its messages accordingly.

However, packetization layers are encouraged to avoid sending messages that will require source fragmentation (for the case against fragmentation, see [FRAG]).

5.2. Storing PMTU information

Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. However, in most cases a node will not have enough information to completely and accurately identify such a path. Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path. For nodes with multiple interfaces, Path MTU information should be maintained for each IPv6 link.

In the case of a multicast destination address, copies of a packet may traverse many different paths to reach many different nodes. The local representation of the "path" to a multicast destination must represent a potentially large set of paths.

Minimally, an implementation could maintain a single PMTU value to be used for all packets originated from the node. This PMTU value would be the minimum PMTU learned across the set of all paths in use by the node. This approach is likely to result in the use of smaller packets than is necessary for many paths. In the case of multipath routing (e.g., Equal Cost Multipath Routing (ECMP)), a set of paths can exist even for a single source and destination pair.

An implementation could use the destination address as the local representation of a path. The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. This approach will result in the use of optimally sized packets on a per-destination basis. This approach integrates nicely with the conceptual model of a host as described in [ND]: a PMTU value could be stored with the corresponding entry in the destination cache.

If flows [I-D.ietf-6man-rfc2460bis] are in use, an implementation could use the flow id as the local representation of a path. Packets sent to a particular destination but belonging to different flows may use different paths, as with ECMP, in which the choice of path might depend on the flow id. This approach might result in the use of optimally sized packets on a per-flow basis, providing finer granularity than PMTU values maintained on a per-destination basis.

For source routed packets (i.e. packets containing an IPv6 Routing header [I-D.ietf-6man-rfc2460bis]), the source route may further qualify the local representation of a path.

Initially, the PMTU value for a path is assumed to be the (known) MTU of the first-hop link.

When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to.

Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet. If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header. If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header.

The node then uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value or the IPv6 minimum link MTU if that is larger, and compares the tentative PMTU to the existing PMTU. If the tentative PMTU is less than the existing PMTU estimate, the tentative PMTU replaces the existing PMTU as the PMTU value for the path.

The packetization layers must be notified about decreases in the PMTU. Any packetization layer instance (for example, a TCP

connection) that is actively using the path must be notified if the PMTU estimate is decreased.

Note: even if the Packet Too Big message contains an Original Packet Header that refers to a UDP packet, the TCP layer must be notified if any of its connections use the given path.

Also, the instance that sent the packet that elicited the Packet Too Big message should be notified that its packet has been dropped, even if the PMTU estimate has not changed, so that it may retransmit the dropped data.

Note: An implementation can avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate. In this approach, when an attempt is made to SEND a packet that is larger than the PMTU estimate, the SEND function should fail and return a suitable error indication. This approach may be more suitable to a connectionless packetization layer (such as one using UDP), which (in some implementations) may be hard to "notify" from the ICMPv6 layer. In this case, the normal timeout-based retransmission mechanisms would be used to recover from the dropped packets.

It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet.

5.3. Purging stale PMTU information

Internetwork topology is dynamic; routes change over time. While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale.

If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. When a PMTU value has not been decreased for a while (on the order of 10 minutes), it should probe to find if a larger PMTU is supported.

Note: an implementation should provide a means for changing the timeout duration, including setting it to "infinity". For

example, nodes attached to a link with a large MTU which is then attached to the rest of the Internet via a link with a small MTU are never going to discover a new non-local PMTU, so they should not have to put up with dropped packets every 10 minutes.

5.4. Packetization layer actions

A packetization layer (e.g., TCP) must use the PMTU for the path(s) in use by a connection; it should not send segments that would result in packets larger than the PMTU, except to probe during PMTU discovery (this probe packet must not be fragmented to the PMTU). A simple implementation could ask the IP layer for this value each time it created a new segment, but this could be inefficient. An implementation typically caches other values derived from the PMTU. It may be simpler to receive asynchronous notification when the PMTU changes, so that these variables may be also updated.

A TCP implementation must also store the Maximum Segment Size (MSS) value received from its peer, which represents the EMTU_R, the largest packet that can be reassembled by the receiver, and must not send any segment larger than this MSS, regardless of the PMTU.

The value sent in the TCP MSS option is independent of the PMTU; it is determined by the receiver reassembly limit EMTU_R. This MSS option value is used by the other end of the connection, which may be using an unrelated PMTU value. See [I-D.ietf-6man-rfc2460bis] sections "Packet Size Issues" and "Maximum Upper-Layer Payload Size" for information on selecting a value for the TCP MSS option.

Reception of a Packet Too Big message implies that a packet was dropped by the node that sent the ICMPv6 message. A reliable upper layer protocol will detect this loss by its own means, and recover it by its normal retransmission methods. The retransmission could result in delay, depending on the loss detection method used by the upper layer protocol. If the Path MTU Discovery process requires several steps to find the PMTU of the full path, this could finally delay the retransmission by many round-trip times.

Alternatively, the retransmission could be done in immediate response to a notification that the Path MTU was decreased, but only for the specific connection specified by the Packet Too Big message, but only based on the message and connection. The packet size used in the retransmission should be no larger than the new PMTU.

Note: A packetization layer that determines a probe packet is lost, needs to adapt the segment size of the retransmission. Using the reported size in the last Packet Too Big message, however, can lead to further losses as there might be smaller PMTU

limits at the routers further along the path. This would lead to loss of all retransmitted segments and therefore cause unnecessary congestion as well as additional packets to be sent each time a new router announces a smaller MTU. Any packetization layer that uses retransmission is therefore also responsible for congestion control of its retransmissions [RFC8085].

A loss caused by a PMTU probe indicated by the reception of a Packet Too Big message must not be considered as a congestion notification and hence the congestion window may not change.

5.5. Issues for other transport protocols

Some transport protocols are not allowed to repacketize when doing a retransmission. That is, once an attempt is made to transmit a segment of a certain size, the transport cannot split the contents of the segment into smaller segments for retransmission. In such a case, the original segment can be fragmented by the IP layer during retransmission. Subsequent segments, when transmitted for the first time, should be no larger than allowed by the Path MTU.

Path MTU Discovery for IPv4 [RFC1191] used NFS as an example of a UDP-based application that benefits from PMTU discovery. Since then [RFC7530], states the supported transport layer between NFS and IP must be an IETF standardized transport protocol that is specified to avoid network congestion; such transports include TCP, Stream Control Transmission Protocol (SCTP) [RFC4960], and the Datagram Congestion Control Protocol (DCCP) [RFC4340]. In this case, the transport is responsible for ensuring that transmitted segments (except probes) conform to the the Path MTU, including supporting PMTU discovery probe transmissions as needed.

5.6. Management interface

It is suggested that an implementation provide a way for a system utility program to:

- Specify that Path MTU Discovery not be done on a given path.
- Change the PMTU value associated with a given path.

The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU.

These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values.

The implementation should also provide a way to change the timeout period for aging stale PMTU information.

6. Security Considerations

This Path MTU Discovery mechanism makes possible two denial-of-service attacks, both based on a malicious party sending false Packet Too Big messages to a node.

In the first attack, the false message indicates a PMTU much smaller than reality. In response, the victim node should never set its PMTU estimate below the IPv6 minimum link MTU. A sender that falsely reduces to this MTU would observe suboptimal performance.

In the second attack, the false message indicates a PMTU larger than reality. If believed, this could cause temporary blockage as the victim sends packets that will be dropped by some router. Within one round-trip time, the node would discover its mistake (receiving Packet Too Big messages from that router), but frequent repetition of this attack could cause lots of packets to be dropped. A node, however, must not raise its estimate of the PMTU based on a Packet Too Big message, so should not be vulnerable to this attack.

Both of these attacks can cause a black hole connection, that is, the TCP three-way handshake completes correctly but the connection hangs when data is transferred.

A malicious party could also cause problems if it could stop a victim from receiving legitimate Packet Too Big messages, but in this case there are simpler denial-of-service attacks available.

If ICMPv6 filtering prevents reception of ICMPv6 Packet Too Big messages, the source will not learn the actual path MTU. Packetization Layer Path MTU Discovery [RFC4821] does not rely upon network support for ICMPv6 messages and is therefore considered more robust than standard PMTUD. It is not susceptible to "black holed" connections caused by filtering of ICMPv6 message. See [RFC4890] for recommendations regarding filtering ICMPv6 messages.

7. Acknowledgements

We would like to acknowledge the authors of and contributors to [RFC1191], from which the majority of this document was derived. We would also like to acknowledge the members of the IPng working group for their careful review and constructive criticisms.

We would also like to acknowledge the contributors to this update of "Path MTU Discovery for IP version 6". This includes members of the 6MAN w.g., area directorate reviewers, the IESG, and especially to Joe Touch and Gorrry Fairhurst.

8. IANA Considerations

This document does not have any IANA actions

9. References

9.1. Normative References

- [I-D.ietf-6man-rfc2460bis] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-13 (work in progress), May 2017.
- [ICMPv6] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.

9.2. Informative References

- [FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology , August 1987.
- [ND] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<http://www.rfc-editor.org/info/rfc1191>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<http://www.rfc-editor.org/info/rfc1981>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <<http://www.rfc-editor.org/info/rfc2923>>.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<http://www.rfc-editor.org/info/rfc4340>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<http://www.rfc-editor.org/info/rfc4821>>.
- [RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <<http://www.rfc-editor.org/info/rfc4890>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC6691] Borman, D., "TCP Options and Maximum Segment Size (MSS)", RFC 6691, DOI 10.17487/RFC6691, July 2012, <<http://www.rfc-editor.org/info/rfc6691>>.
- [RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <<http://www.rfc-editor.org/info/rfc7530>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<http://www.rfc-editor.org/info/rfc8085>>.

Appendix A. Comparison to RFC 1191

This document is based in large part on RFC 1191, which describes Path MTU Discovery for IPv4. Certain portions of RFC 1191 were not needed in this document:

router specification Packet Too Big messages and corresponding
 router behavior are defined in [ICMPv6]

Don't Fragment bit	there is no DF bit in IPv6 packets
TCP MSS discussion	selecting a value to send in the TCP MSS option is discussed in [I-D.ietf-6man-rfc2460bis]
old-style messages	all Packet Too Big messages report the MTU of the constricting link
MTU plateau tables	not needed because there are no old-style messages

Appendix B. Changes Since RFC 1981

This document is based on RFC1981 has the following changes from RFC1981:

- o Clarified Section 1 "Introduction" that the purpose of PMTUD is to reduce the need for IPv6 fragmentation.
- o Added text to Section 1 "Introduction" about the effects on PMTUD when ICMPv6 messages are blocked.
- o Added Note to Introduction that document that this document doesn't cite RFC2119 and only uses lower case "should/must" language. Changed all upper case "should/must" to lower case.
- o Added a short summary to the Section 1 "Introduction" of Packetization Layer Path MTU Discovery ((PLPMTUD) and a reference to RFC4821 that defines it.
- o Aligned text in Section 2 "Terminology" to match current packetization layer terminology.
- o Added clarification in Section 4 "Protocol Requirements" that nodes should validate the payload of ICMP PTB message per RFC4443, and that nodes should detect decreases in PMTU as fast as possible.
- o Remove Note from Section 4 "Protocol Requirements" about a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU because this was removed from [I-D.ietf-6man-rfc2460bis].
- o Added clarification in Section 5.2 "Storing PMTU information" to discard an ICMPv6 Packet Too Big message if it contains a MTU less than the IPv6 minimum link MTU.

- o Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.
- o Removed text in Section 5.2 "Storing PMTU information" about the RH0 routing header because it was deprecated by RFC5095.
- o Removed text about obsolete security classification from Section 5.2 "Storing PMTU information".
- o Changed title of Section 5.4 to "Packetization Layer actions" and changed to text in the first paragraph to generalize this section to cover all packetization layers, not just TCP.
- o Clarified text in Section 5.4 "Packetization Layer actions" to use normal packetization layer retransmission methods.
- o Removed text in Section 5.4 "Packetization Layer actions" that described 4.2 BSD because it is obsolete, and removed reference to TP4.
- o Updated text in Section 5.5 "Issues for other transport protocols" about NFS including adding a current reference to NFS and removing obsolete text.
- o Added paragraph to Section 6 "Security Considerations" about black hole connections if PTB messages are not received, and comparison to PLPMTD.
- o Updated Section 7 "Acknowledgements".
- o Editorial Changes.

B.1. Change History Since RFC1981

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

- 08) Based on IESG comments, cleaned up text in Section 5.3 regarding suggested action when PMTU value has not been decreased recently.
- 08) Revision of Note in Section 5.4 to make text clearer.
- 08) Updated Section 7 "Acknowledgements".
- 08) Editorial Changes.
- 07) Changes from the IESG Discuss comments from IESG reviews. The changes include:
 - o Added Note to Introduction that document that this document doesn't cite RFC2119 and only uses lower case "should/must" language. Changed all upper case "should/must" to lower case.
 - o Added references for EMTU_S and EMTU_R.
 - o Added clarification to Section 4 "Protocol Requirements" that nodes should detect decreases in PMTU as fast as possible.
 - o Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.
 - o Removed text in Section 5.2 about Retransmission because it was unneeded.
 - o Removed text in Section 5.3 about Retransmission because it was unneeded.
 - o Rewrote text in Section 5.4 "Packetization Layer actions" regarding reception to make it clearer.
 - o Rewrote the text at the end of Section 5.4 to remove unnecessary details and clarify not change congestion window.
 - o Added references in Section 5.5 for SCTP and added DCCP (and reference) the list of examples.

- o Added paragraph to Section 5.5 "Security Considerations" about black hole connections if PTB messages are not received, and comparison to PLPMTD.
- 07) Editorial changes.
- 06) Revised Appendix B "Changes since RFC1981" to have a summary of changes since RFC1981 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.
- 06) Editorial changes based on comments received after publishing the -05 draft.
- 05) Changes based on IETF last call reviews by Gorrry Fairhurst, Joe Touch, Susan Hares, Stewart Bryant, Rifaat Shekh-Yusef, and Donald Eastlake. This includes includes:
- o Clarify that the purpose of PMTUD is to reduce the need for IPv6 Fragmentation.
 - o Added text to Introduction about effects on PMTUD when ICMPv6 messages are blocked.
 - o Clarified in Section 4. that nodes should validate the payload of ICMPv6 PTB messages per RFC4443.
 - o Removed text in Section 5.2 about the number of paths to a destination.
 - o Changed title of Section 5.4 to "Packetization layer actions".
 - o Clarified first paragraph in Section 5.4 to to cover all packetization layers, not just TCP.
 - o Clarified text in Section 5.4 to use normal retransmission methods.
 - o Add clarification to Note in Section 5.4 about retransmissions.
 - o Removed text in Section 5.4 that described 4.2BSD as it is now obsolete.
 - o Removed reference to TP4 in Section 5.5.

- o Updated text in Section 5.5 about NFS including adding a current reference to NFS and removing obsolete text.
 - o Revised text in Section 6 to clarify first attack response.
 - o Added new text in Section 6 to clarify the effect of ICMPv6 filtering on PMTUD.
 - o Aligned terminology for the packetization layer terminology.
 - o Editorial changes.
- 04) Changes based on AD Evaluation including removing details about RFC4821 algorithm in Section 1, remove text about decrementing hop limit from Section 3, and removed text about obsolete security classifications from Section 5.2.
- 04) Editorial changes and clarification in Section 5.2 based on IP Directorate review by Donald Eastlake
- 03) Remove text in Section 5.3 regarding RH0 since it was deprecated by RFC5095
- 02) Clarified in Section 3 that ICMPv6 Packet Too Big should be sent even if the node doesn't decrement the hop limit
- 01) Revised the text about PLPMTUD to use the word "path".
- 01) Editorial changes.
- 00) Added text to discard an ICMPv6 Packet Too Big message containing an MTU less than the IPv6 minimum link MTU.
- 00) Revision of text regarding RFC4821.
- 00) Added R. Hinden as Editor to facilitate ID submission.
- 00) Editorial changes.

Individual Internet Drafts

- 01) Remove Note about a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU. This was removed from [I-D.ietf-6man-rfc2460bis].

- 01) Include a link to RFC4821 along with a short summary of what it does.
- 01) Assigned references to informative and normative.
- 01) Editorial changes.
- 00) Establish a baseline from RFC1981. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, updating references, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Jack McCann
Digital Equipment Corporation

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Jeffrey Mogul
Digital Equipment Corporation

Robert M. Hinden (editor)
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Network Working Group
Internet-Draft
Obsoletes: 2460 (if approved)
Intended status: Standards Track
Expires: November 20, 2017

S. Deering
Retired
R. Hinden
Check Point Software
May 19, 2017

Internet Protocol, Version 6 (IPv6) Specification
draft-ietf-6man-rfc2460bis-13

Abstract

This document specifies version 6 of the Internet Protocol (IPv6).
It obsoletes RFC2460

Status of This Memo

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

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

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

This Internet-Draft will expire on November 20, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the
document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. IPv6 Header Format	5
4. IPv6 Extension Headers	6
4.1. Extension Header Order	8
4.2. Options	9
4.3. Hop-by-Hop Options Header	12
4.4. Routing Header	12
4.5. Fragment Header	14
4.6. Destination Options Header	21
4.7. No Next Header	22
4.8. Defining New Extension Headers and Options	22
5. Packet Size Issues	23
6. Flow Labels	24
7. Traffic Classes	24
8. Upper-Layer Protocol Issues	24
8.1. Upper-Layer Checksums	25
8.2. Maximum Packet Lifetime	26
8.3. Maximum Upper-Layer Payload Size	27
8.4. Responding to Packets Carrying Routing Headers	27
9. IANA Considerations	27
10. Security Considerations	28
11. Acknowledgments	30
12. References	30
12.1. Normative References	30
12.2. Informative References	31
Appendix A. Formatting Guidelines for Options	33
Appendix B. Changes Since RFC2460	36
B.1. Change History Since RFC2460	39
Authors' Addresses	45

1. Introduction

IP version 6 (IPv6) is a new version of the Internet Protocol (IP), designed as the successor to IP version 4 (IPv4) [RFC0791]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- o Expanded Addressing Capabilities

IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.

- o Header Format Simplification

Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.

- o Improved Support for Extensions and Options

Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.

- o Flow Labeling Capability

A new capability is added to enable the labeling of sequences of packets that the sender requests to be treated in the network as a single flow.

- o Authentication and Privacy Capabilities

Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

This document specifies the basic IPv6 header and the initially-defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in [RFC4291].

The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [RFC4443]

The data transmission order for IPv6 is the same as for IPv4 as defined in Appendix B of [RFC0791].

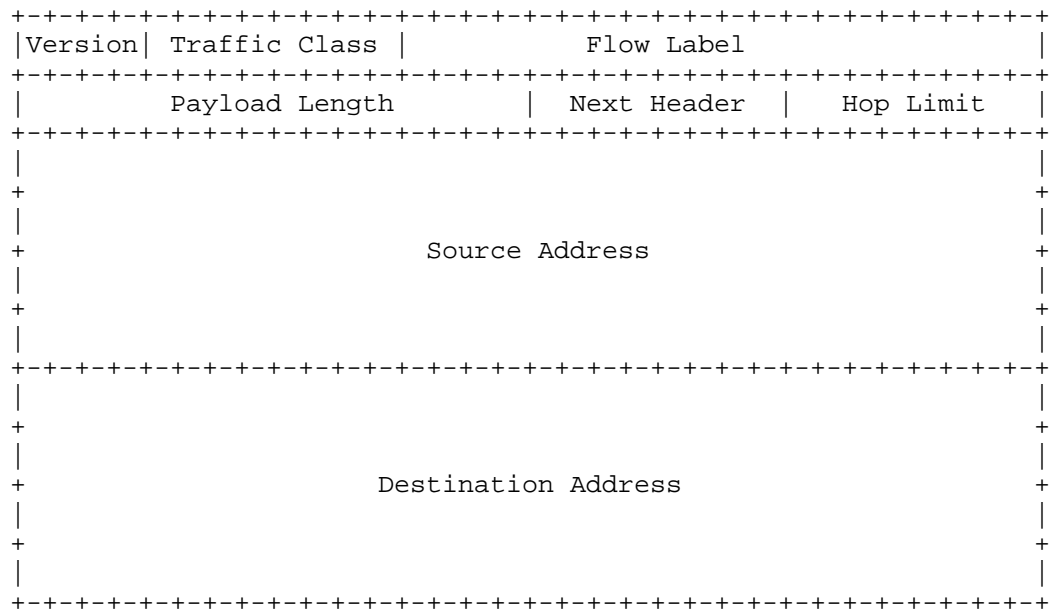
Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

2. Terminology

node	a device that implements IPv6.
router	a node that forwards IPv6 packets not explicitly addressed to itself. [See Note below].
host	any node that is not a router. [See Note below].
upper layer	a protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.
link	a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernet (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.
neighbors	nodes attached to the same link.
interface	a node's attachment to a link.
address	an IPv6-layer identifier for an interface or a set of interfaces.
packet	an IPv6 header plus payload.
link MTU	the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a link.
path MTU	the minimum link MTU of all the links in a path between a source node and a destination node.

Note: it is possible for a device with multiple interfaces to be configured to forward non-self-destined packets arriving from some set (fewer than all) of its interfaces, and to discard non-self-destined packets arriving from its other interfaces. Such a device must obey the protocol requirements for routers when receiving packets from, and interacting with neighbors over, the former (forwarding) interfaces. It must obey the protocol requirements for hosts when receiving packets from, and interacting with neighbors over, the latter (non-forwarding) interfaces.

3. IPv6 Header Format



Version	4-bit Internet Protocol version number = 6.
Traffic Class	8-bit traffic class field. See section 7.
Flow Label	20-bit flow label. See section 6.
Payload Length	16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. (Note that any extension headers [Section 4] present are considered part of the payload, i.e., included in the length count.)

Next Header	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hop Limit	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. When forwarding, the packet is discarded if Hop Limit was zero when received or is decremented to zero. A node that is the destination of a packet should not discard a packet with hop limit equal to zero, it should process the packet normally.
Source Address	128-bit address of the originator of the packet. See [RFC4291].
Destination Address	128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present). See [RFC4291] and section 4.4.

4. IPv6 Extension Headers

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. There is a small number of such extension headers, each one identified by a distinct Next Header value.

Extension Headers are numbered from IANA IP Protocol Numbers [IANA-PN], the same values used for IPv4 and IPv6. When processing a sequence of Next Header values in a packet, the first one that is not an Extension Header [IANA-EH] indicates that the next item in the packet is the corresponding upper-layer header. A special "No Next Header" value is used if there is no upper-layer header.

As illustrated in these examples, an IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header:

IPv6 header	TCP header + data
Next Header = TCP	

IPv6 header	Routing header	TCP header + data
Next Header = Routing	Next Header = TCP	

IPv6 header	Routing header	Fragment header	fragment of TCP header + data
Next Header = Routing	Next Header = Fragment	Next Header = TCP	

Extension headers (except for the Hop-by-Hop Options header) are not processed, inserted, or deleted 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.

The Hop-by-Hop Options header is not inserted or deleted, but may be 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. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

NOTE: While [RFC2460] required that all nodes must examine and process the Hop-by-Hop Options header, it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so.

At the Destination node, normal demultiplexing on the Next Header field of the IPv6 header invokes the module to process the first extension header, or the upper-layer header if no extension header is present. The contents and semantics of each extension header determine whether or not to proceed to the next header. Therefore, extension headers must be processed strictly in the order they appear in the packet; a receiver must not, for example, scan through a

packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

If, as a result of processing a header, the destination node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet. The same action should be taken if a node encounters a Next Header value of zero in any header other than an IPv6 header.

Each extension header is an integer multiple of 8 octets long, in order to retain 8-octet alignment for subsequent headers. Multi-octet fields within each extension header are aligned on their natural boundaries, i.e., fields of width *n* octets are placed at an integer multiple of *n* octets from the start of the header, for *n* = 1, 2, 4, or 8.

A full implementation of IPv6 includes implementation of the following extension headers:

- Hop-by-Hop Options
- Fragment
- Destination Options
- Routing
- Authentication
- Encapsulating Security Payload

The first four are specified in this document; the last two are specified in [RFC4302] and [RFC4303], respectively. The current list of IPv6 extension headers can be found at [IANA-EH].

4.1. Extension Header Order

When more than one extension header is used in the same packet, it is recommended that those headers appear in the following order:

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header (note 1)
- Routing header
- Fragment header
- Authentication header (note 2)
- Encapsulating Security Payload header (note 2)
- Destination Options header (note 3)
- upper-layer header

note 1: for options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.

note 2: additional recommendations regarding the relative order of the Authentication and Encapsulating Security Payload headers are given in [RFC4303].

note 3: for options to be processed only by the final destination of the packet.

Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header).

If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

If and when other extension headers are defined, their ordering constraints relative to the above listed headers must be specified.

IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the above recommended order until and unless subsequent specifications revise that recommendation.

4.2. Options

Two of the currently-defined extension headers defined in this document -- the Hop-by-Hop Options header and the Destination Options header -- carry a variable number of type-length-value (TLV) encoded "options", of the following format:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| Option Type | Opt Data Len | Option Data |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Option Type 8-bit identifier of the type of option.

Opt Data Len 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data Variable-length field. Option-Type-specific data.

The sequence of options within a header must be processed strictly in the order they appear in the header; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type:

- 00 - skip over this option and continue processing the header.
- 01 - discard the packet.
- 10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
- 11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination. When an Authentication header is present in the packet, for any option whose data may change en-route, its entire Option Data field must be treated as zero-valued octets when computing or verifying the packet's authenticating value.

- 0 - Option Data does not change en-route
- 1 - Option Data may change en-route

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

The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers.

Individual options may have specific alignment requirements, to ensure that multi-octet values within Option Data fields fall on natural boundaries. The alignment requirement of an option is specified using the notation $xn+y$, meaning the Option Type must appear at an integer multiple of x octets from the start of the header, plus y octets. For example:

2n means any 2-octet offset from the start of the header.
8n+2 means any 8-octet offset from the start of the header, plus 2 octets.

There are two padding options which are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. These padding options must be recognized by all IPv6 implementations:

Pad1 option (alignment requirement: none)

$$\begin{array}{ccccccccc} + & - & + & - & + & - & + & - & + \\ | & & & & & & 0 & & & & | \\ + & - & + & - & + & - & + & - & + \end{array}$$

NOTE! the format of the Padl option is a special case -- it does not have length and value fields.

The Pad1 option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

PadN option (alignment requirement: none)

1	Opt Data Len	Option Data
---	--------------	-------------

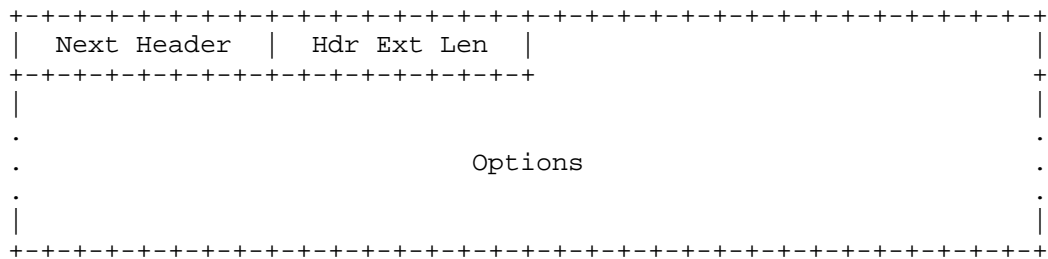
The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the

Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

Appendix A contains formatting guidelines for designing new options.

4.3. Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to carry optional information that may be examined and processed by every node along a packet's delivery path. The Hop-by-Hop Options header is identified by a Next Header value of 0 in the IPv6 header, and has the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Options	Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only hop-by-hop options defined in this document are the Pad1 and PadN options specified in section 4.2.

4.4. Routing Header

The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is very similar to IPv4's Loose Source

and Record Route option. The Routing header is identified by a Next Header value of 43 in the immediately preceding header, and has the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len | Routing Type | Segments Left |
+-----+-----+-----+-----+-----+-----+-----+-----+
|
.
.
.
.
.
+-----+-----+-----+-----+-----+-----+-----+-----+

```

type-specific data

Next Header	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets.
Routing Type	8-bit identifier of a particular Routing header variant.
Segments Left	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
type-specific data	Variable-length field, of format determined by the Routing Type, and of length such that the complete Routing header is an integer multiple of 8 octets long.

If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field, as follows:

If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing header.

If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

If, after processing a Routing header of a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node must discard the packet and send an ICMP Packet Too Big message to the packet's Source Address.

The currently defined IPv6 Routing Headers and their status can be found at [IANA-RH]. Allocation guidelines for IPv6 Routing Headers can be found in [RFC5871].

4.5. Fragment Header

The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.) The Fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Reserved | Fragment Offset | Res | M |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Identification                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Next Header	8-bit selector. Identifies the initial header type of the Fragmentable Part of the original packet (defined below). Uses the same values as the IPv4 Protocol field [IANA-PN].
Reserved	8-bit reserved field. Initialized to zero for transmission; ignored on reception.
Fragment Offset	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet.
Res	2-bit reserved field. Initialized to zero for transmission; ignored on reception.
M flag	1 = more fragments; 0 = last fragment.

Identification 32 bits. See description below.

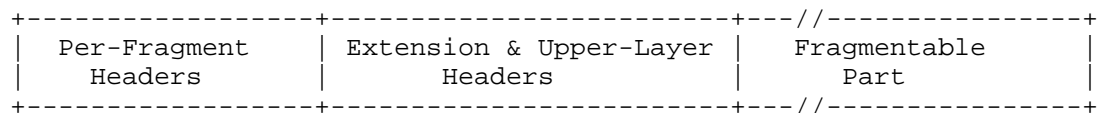
In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

For every packet that is to be fragmented, the source node generates an Identification value. The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address. If a Routing header is present, the Destination Address of concern is that of the final destination.

- * "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node knows the maximum packet lifetime. Rather, it is assumed that the requirement can be met by implementing an algorithm that results in a low identification reuse frequency. Examples of algorithms that can meet this requirement are described in [RFC7739].

The initial, large, unfragmented packet is referred to as the "original packet", and it is considered to consist of three parts, as illustrated:

original packet:



The Per-Fragment Headers must consist of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers.

The Extension Headers are all other extension headers that are not included in the Per-Fragment headers part of the packet. For this purpose, the Encapsulating Security Payload (ESP) is not considered an extension header. The Upper-Layer Header is the first upper-layer header that is not an IPv6 extension header.

Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted ESP.

The Fragmentable Part consists of the rest of the packet after the upper-layer header or after any header (i.e., initial IPv6 header or extension header) that contains a Next Header value of No Next Header.

The Fragmentable Part of the original packet is divided into fragments. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s). Each complete fragment, except possibly the last ("rightmost") one, being an integer multiple of 8 octets long.

The fragments are transmitted in separate "fragment packets" as illustrated:

original packet:

Per-Fragment Headers	Ext & Upper-Layer Headers	first fragment	second fragment	...	last fragment
-------------------------	------------------------------	-------------------	--------------------	-----	------------------

fragment packets:

Per-Fragment Headers	Fragment Header	Ext & Upper-Layer Headers	first fragment
-------------------------	--------------------	------------------------------	-------------------

Per-Fragment Headers	Fragment Header	second fragment
-------------------------	--------------------	--------------------

o
o
o

Per-Fragment Headers	Fragment Header	last fragment
-------------------------	--------------------	------------------

The first fragment packet is composed of:

- (1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the

IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable Part of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0.

An M flag value of 1 as this is the first fragment.

The Identification value generated for the original packet.

(3) Extension Headers, if any, and the Upper-Layer header. These headers must be in the first fragment. Note: This restricts the size of the headers through the Upper-Layer header to the MTU of the path to the packets' destinations(s).

(4) The first fragment.

The subsequent fragment packets are composed of:

(1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable part of the original packet.

An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1.

The Identification value generated for the original packet.

(3) The fragment itself.

Fragments must not be created that overlap with any other fragments created from the original packet.

At the destination, fragment packets are reassembled into their original, unfragmented form, as illustrated:

reassembled original packet:

```
+-----+-----+-----+-----+//+-----+
| Per-Fragment | Ext & Upper-Layer | first | second |      | last |
|   Headers    |   Headers    | frag data | fragment | .... | fragment |
+-----+-----+-----+-----+//+-----+
```

The following rules govern reassembly:

An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification.

The Per-Fragment Headers of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following two changes:

The Next Header field of the last header of the Per-Fragment Headers is obtained from the Next Header field of the first fragment's Fragment header.

The Payload Length of the reassembled packet is computed from the length of the Per-Fragment Headers and the length and offset of the last fragment. For example, a formula for computing the Payload Length of the reassembled original packet is:

$$PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last$$

where

PL.orig = Payload Length field of reassembled packet.

PL.first = Payload Length field of first fragment packet.

FL.first = length of fragment following Fragment header of first fragment packet.
FO.last = Fragment Offset field of Fragment header of last fragment packet.
FL.last = length of fragment following Fragment header of last fragment packet.

The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets. The length of each fragment is computed by subtracting from the packet's Payload Length the length of the headers between the IPv6 header and fragment itself; its relative position in Fragmentable Part is computed from its Fragment Offset value.

The Fragment header is not present in the final, reassembled packet.

If the fragment is a whole datagram (that is, both the Fragment Offset field and the M flag are zero), then it does not need any further reassembly and should be processed as a fully reassembled packet (i.e., updating Next Header, adjust Payload Length, removing the Fragmentation Header, etc.). Any other fragments that match this packet (i.e., the same IPv6 Source Address, IPv6 Destination Address, and Fragment Identification) should be processed independently.

The following error conditions may arise when reassembling fragmented packets:

- o If insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded. If the first fragment (i.e., the one with a Fragment Offset of zero) has been received, an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message should be sent to the source of that fragment.
- o If the length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet.

- o If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet.
- o If the first fragment does not include all headers through an Upper-Layer header, then that fragment should be discarded and an ICMP Parameter Problem, Code 3, message should be sent to the source of the fragment, with the Pointer field set to zero.
- o If any of the fragments being reassembled overlaps with any other fragments being reassembled for the same packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded and no ICMP error messages should be sent.

It should be noted that fragments may be duplicated in the network. Instead of treating these exact duplicate fragments as overlapping fragments, an implementation may choose to detect this case and drop exact duplicate fragments while keeping the other fragments belonging to the same packet.

The following conditions are not expected to occur frequently, but are not considered errors if they do:

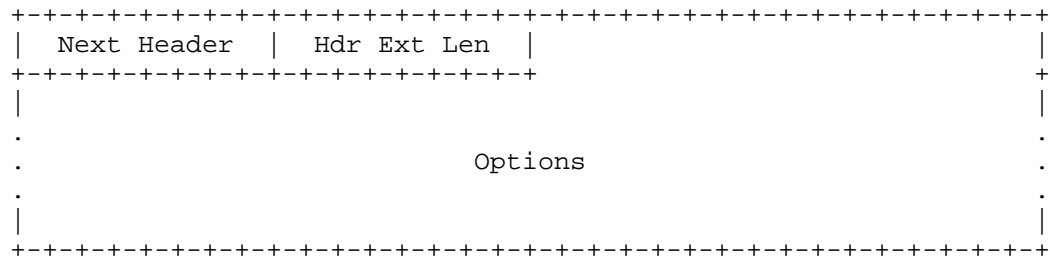
The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly. Only those headers in the Offset zero fragment packet are retained in the reassembled packet.

The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly.

Other fields in the IPv6 header may also vary across the fragments being reassembled. Specifications that use these fields may provide additional instructions if the basic mechanism of using the values from the Offset zero fragment is not sufficient. For example, Section 5.3 of [RFC3168] describes how to combine the Explicit Congestion Notification (ECN) bits from different fragments to derive the ECN bits of the reassembled packet.

4.6. Destination Options Header

The Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header, and has the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the Destination Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Options	Variable-length field, of length such that the complete Destination Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only destination options defined in this document are the Pad1 and PadN options specified in section 4.2.

Note that there are two possible ways to encode optional destination information in an IPv6 packet: either as an option in the Destination Options header, or as a separate extension header. The Fragment header and the Authentication header are examples of the latter approach. Which approach can be used depends on what action is desired of a destination node that does not understand the optional information:

- o If the desired action is for the destination node to discard the packet and, only if the packet's Destination Address is not a multicast address, send an ICMP Unrecognized Type message to the packet's Source Address, then the information may be encoded either as a separate header or as an option in the Destination Options header whose Option Type has the value 11 in its highest-order two bits. The choice may depend on such factors as which takes fewer octets, or which yields better alignment or more efficient parsing.
- o If any other action is desired, the information must be encoded as an option in the Destination Options header whose Option Type has the value 00, 01, or 10 in its highest-order two bits, specifying the desired action (see section 4.2).

4.7. No Next Header

The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header. If the Payload Length field of the IPv6 header indicates the presence of octets past the end of a header whose Next Header field contains 59, those octets must be ignored, and passed on unchanged if the packet is forwarded.

4.8. Defining New Extension Headers and Options

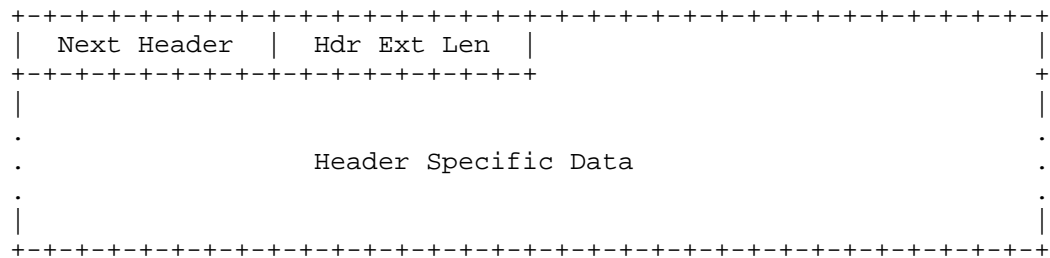
Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6 extension header can not be used for the desired new function. See [RFC6564] for additional background information.

Note: New extension headers that require hop-by-hop behavior must not be defined because, as specified in Section 4 of this document, the only Extension Header that has hop-by-hop behavior is the Hop-by-Hop Options header.

New hop-by-hop options are not recommended because nodes may be configured to ignore the Hop-by-Hop Option header, drop packets containing a hop-by-hop header, or assign packets containing a hop-by-hop header to a slow processing path. Designers considering defining new hop-by-hop options need to be aware of this likely behaviour. There has to be a very clear justification why any new hop-by-hop option is needed before it is standardized.

Instead of defining new Extension Headers, it is recommended that the Destination Options header is used to carry optional information that must be examined only by a packet's destination node(s), because they provide better handling and backward compatibility.

If new Extension Headers are defined, they need to use the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the extension header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Header Specific Data	Variable-length field. Fields specific to the extension header.

5. Packet Size Issues

IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. This is known as the IPv6 minimum link MTU. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6.

Links that have a configurable MTU (for example, PPP links [RFC1661]) must be configured to have an MTU of at least 1280 octets; it is recommended that they be configured with an MTU of 1500 octets or greater, to accommodate possible encapsulations (i.e., tunneling) without incurring IPv6-layer fragmentation.

From each link to which a node is directly attached, the node must be able to accept packets as large as that link's MTU.

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

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

A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets. A node is permitted to accept fragmented packets that reassemble to more than 1500 octets. An upper-layer protocol or application that depends on IPv6 fragmentation to send packets larger than the MTU of a path should not send packets larger than 1500 octets unless it has assurance that the destination is capable of reassembling packets of that larger size.

6. Flow Labels

The 20-bit Flow Label field in the IPv6 header is used by a source to label sequences of packets to be treated in the network as a single flow.

The current definition of the IPv6 Flow Label can be found in [RFC6437].

7. Traffic Classes

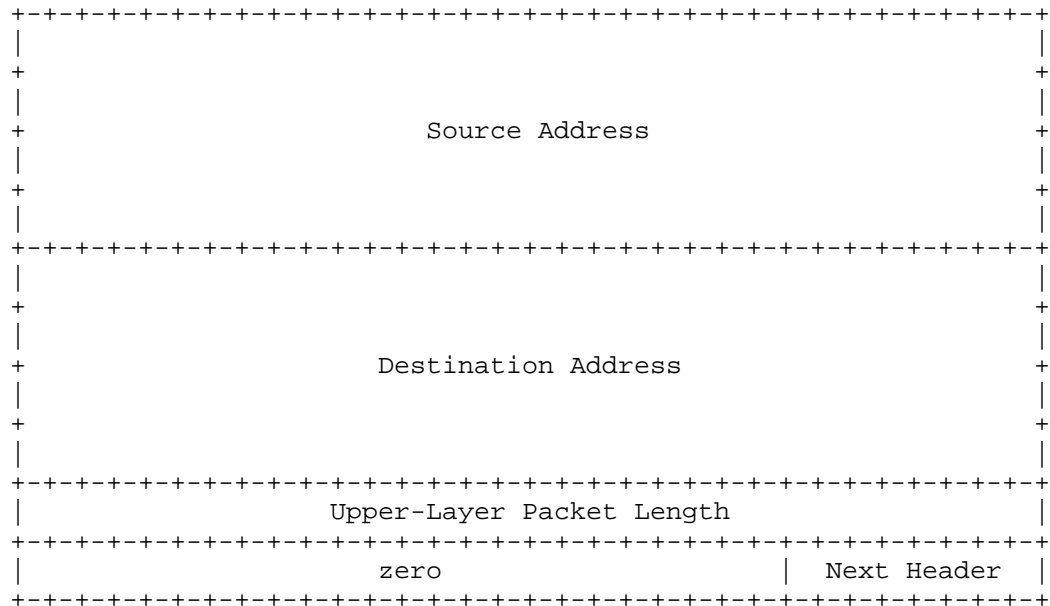
The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet or fragment might be different from the value sent by the packet's source.

The current use of the Traffic Class field for Differentiated Services and Explicit Congestion Notification is specified in [RFC2474] and [RFC3168].

8. Upper-Layer Protocol Issues

8.1. Upper-Layer Checksums

Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:



- o If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header.
- o The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.
- o The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own

length information (e.g., the Length field in the UDP header); for such protocols, that is the length used in the pseudo-header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.

- o Unlike IPv4, the default behavior when UDP packets are originated by an IPv6 node is that the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.
- o As an exception to the default behaviour, protocols that use UDP as a tunnel encapsulation may enable zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing zero-checksum mode must follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

The IPv6 version of ICMP [RFC4443] includes the above pseudo-header in its checksum computation; this is a change from the IPv4 version of ICMP, which does not include a pseudo-header in its checksum. The reason for the change is to protect ICMP from misdelivery or corruption of those fields of the IPv6 header on which it depends, which, unlike IPv4, are not covered by an internet-layer checksum. The Next Header field in the pseudo-header for ICMP contains the value 58, which identifies the IPv6 version of ICMP.

8.2. Maximum Packet Lifetime

Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime. That is the reason the IPv4 "Time to Live" field was renamed "Hop Limit" in IPv6. In practice, very few, if any, IPv4 implementations conform to the requirement that they limit packet lifetime, so this is not a change in practice. Any upper-layer protocol that relies on the internet layer (whether IPv4 or IPv6) to limit packet lifetime ought to be upgraded to provide its own mechanisms for detecting and discarding obsolete packets.

8.3. Maximum Upper-Layer Payload Size

When computing the maximum payload size available for upper-layer data, an upper-layer protocol must take into account the larger size of the IPv6 header relative to the IPv4 header. For example, in IPv4, TCP's MSS option is computed as the maximum packet size (a default value or a value learned through Path MTU Discovery) minus 40 octets (20 octets for the minimum-length IPv4 header and 20 octets for the minimum-length TCP header). When using TCP over IPv6, the MSS must be computed as the maximum packet size minus 60 octets, because the minimum-length IPv6 header (i.e., an IPv6 header with no extension headers) is 20 octets longer than a minimum-length IPv4 header.

8.4. Responding to Packets Carrying Routing Headers

When an upper-layer protocol sends one or more packets in response to a received packet that included a Routing header, the response packet(s) must not include a Routing header that was automatically derived by "reversing" the received Routing header UNLESS the integrity and authenticity of the received Source Address and Routing header have been verified (e.g., via the use of an Authentication header in the received packet). In other words, only the following kinds of packets are permitted in response to a received packet bearing a Routing header:

- o Response packets that do not carry Routing headers.
- o Response packets that carry Routing headers that were NOT derived by reversing the Routing header of the received packet (for example, a Routing header supplied by local configuration).
- o Response packets that carry Routing headers that were derived by reversing the Routing header of the received packet IF AND ONLY IF the integrity and authenticity of the Source Address and Routing header from the received packet have been verified by the responder.

9. IANA Considerations

RFC2460 is referenced in a number of IANA registries. These include:

- o Internet Protocol Version 6 (IPv6) Parameters [IANA-6P]

- o Assigned Internet Protocol Numbers [IANA-PN]
- o ONC RPC Network Identifiers (netids) [IANA-NI]
- o Technical requirements for authoritative name servers [IANA-NS]
- o Network Layer Protocol Identifiers (NLPIDs) of Interest [IANA-NL]
- o Protocol Registries [IANA-PR]
- o Structure of Management Information (SMI) Numbers (MIB Module Registrations) [IANA-MI]

The IANA should update these references to point to this document.

10. Security Considerations

IPv6, from the viewpoint of the basic format and transmission of packets, has security properties that are similar to IPv4. These security issues include:

- o Eavesdropping, On-path elements can observe the whole packet (including both contents and metadata) of each IPv6 datagram.
- o Replay, where attacker records a sequence of packets off of the wire and plays them back to the party which originally received them.
- o Packet insertion, where the attacker forges a packet with some chosen set of properties and injects it into the network.
- o Packet deletion, where the attacker remove a packet from the wire.
- o Packet modification, where the attacker removes a packet from the wire, modifies it, and re-injects it into the network.
- o Man in the Middle attacks, where the attacker subverts the communication stream in order to pose as the sender to receiver and the receiver to the sender.
- o Denial of Service Attacks, where the attacker sends large amounts of legitimate traffic to a destination to overwhelm it.

IPv6 packets can be protected from eavesdropping, replay, packet insertion, packet modification, and man in the middle attacks by use of the "Security Architecture for the Internet Protocol" [RFC4301]. In addition, upper-layer protocols such as TLS or SSH can be used to protect the application layer traffic running on top of IPv6.

There is not any mechanism to protect against "denial of service attacks". Defending against these type of attacks is outside the scope of this specification.

IPv6 addresses are significantly larger than IPv4 address making it much harder to scan the address space across the Internet and even on a single network link (e.g., Local Area Network). See [RFC7707] for more information.

IPv6 addresses of nodes are expected to be more visible on the Internet as compared with IPv4 since the use of address translation technology is reduced. This creates some additional privacy issues such as making it easier to distinguish endpoints. See [RFC7721] for more information.

The design of IPv6 extension headers architecture, while adding a lot of flexibility, also creates new security challenges. As noted below, issues relating the fragment extension header have been resolved, but it's clear that for any new extension header designed in the future, the security implications need to be examined thoroughly, and this needs to include how the new extension header works with existing extension headers. See [RFC7045] for more information.

This version of the IPv6 specification resolves a number of security issues that were found with the previous version [RFC2460] of the IPv6 specification. These include:

- o Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero). See [RFC6946] and [RFC8021] for more information.
- o Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes clarification that no ICMP error message should be sent if overlapping fragments are received. See [RFC5722] for more information.

- 0 Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. See [RFC6946] for more information.
- o Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. See [RFC7112] for more information.
- o Incorporated the updates from [RFC5095] and [RFC5871] to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.

Security issues relating to other parts of IPv6 including addressing, ICMPv6, Path MTU Discovery, etc., are discussed in the appropriate specifications.

11. Acknowledgments

The authors gratefully acknowledge the many helpful suggestions of the members of the IPng working group, the End-to-End Protocols research group, and the Internet Community At Large.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move the IPv6 specification to Internet Standard. They are Joe Abley, Shane Amante, Jari Arkko, Manav Bhatia, Ronald P. Bonica, Scott Bradner, Brian Carpenter, P.F. Chimento, Marshall Eubanks, Fernando Gont, James Hoagland, Sheng Jiang, Erik Kline, Suresh Krishnan, Vishwas Manral, George Neville-Neil, Jarno Rajahalme, Pekka Savola, Magnus Westerlund, and James Woodyatt.

12. References

12.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<http://www.rfc-editor.org/info/rfc2474>>.

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<http://www.rfc-editor.org/info/rfc6437>>.

12.2. Informative References

- [IANA-6P] "Internet Protocol Version 6 (IPv6) Parameters", <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>>.
- [IANA-EH] "IPv6 Extension Header Types", <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header>>.
- [IANA-MI] "Structure of Management Information (SMI) Numbers (MIB Module Registrations)", <<http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml>>.
- [IANA-NI] "ONC RPC Network Identifiers (netids)", <<http://www.iana.org/assignments/rpc-netids/rpc-netids.xhtml>>.
- [IANA-NL] "Network Layer Protocol Identifiers (NLPIDs) of Interest", <<http://www.iana.org/assignments/nlpids/nlpids.xhtml>>.
- [IANA-NS] "Technical requirements for authoritative name servers", <<https://www.iana.org/help/nameserver-requirements>>.
- [IANA-PN] "Assigned Internet Protocol Numbers", <<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>>.

- [IANA-PR] "Protocol Registries", <<https://www.iana.org/protocols>>.
- [IANA-RH] "IANA Routing Types Parameter Registry",
<<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-3>>.
- [RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994,
<<http://www.rfc-editor.org/info/rfc1661>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<http://www.rfc-editor.org/info/rfc1981>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <<http://www.rfc-editor.org/info/rfc5095>>.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <<http://www.rfc-editor.org/info/rfc5722>>.
- [RFC5871] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the IPv6 Routing Header", RFC 5871, DOI 10.17487/RFC5871, May 2010, <<http://www.rfc-editor.org/info/rfc5871>>.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <<http://www.rfc-editor.org/info/rfc6564>>.

- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<http://www.rfc-editor.org/info/rfc6936>>.
- [RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments", RFC 6946, DOI 10.17487/RFC6946, May 2013, <<http://www.rfc-editor.org/info/rfc6946>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <<http://www.rfc-editor.org/info/rfc7112>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<http://www.rfc-editor.org/info/rfc7707>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<http://www.rfc-editor.org/info/rfc7721>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<http://www.rfc-editor.org/info/rfc7739>>.
- [RFC8021] Gont, F., Liu, W., and T. Anderson, "Generation of IPv6 Atomic Fragments Considered Harmful", RFC 8021, DOI 10.17487/RFC8021, January 2017, <<http://www.rfc-editor.org/info/rfc8021>>.

Appendix A. Formatting Guidelines for Options

This appendix gives some advice on how to lay out the fields when designing new options to be used in the Hop-by-Hop Options header or the Destination Options header, as described in section 4.2. These guidelines are based on the following assumptions:

- o One desirable feature is that any multi-octet fields within the Option Data area of an option be aligned on their natural

boundaries, i.e., fields of width n octets should be placed at an integer multiple of n octets from the start of the Hop-by-Hop or Destination Options header, for $n = 1, 2, 4$, or 8 .

- o Another desirable feature is that the Hop-by-Hop or Destination Options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.
- o It may be assumed that, when either of the option-bearing headers are present, they carry a very small number of options, usually only one.

These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of 8 octets). This approach is illustrated in the following examples:

Example 1

If an option X required two data fields, one of length 8 octets and one of length 4 octets, it would be laid out as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     | Option Type=X |Opt Data Len=12|
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     | 4-octet field |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     | 8-octet field |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Its alignment requirement is $8n+2$, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=1 | Option Type=X | Opt Data Len=12 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     8-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Example 2

If an option Y required three data fields, one of length 4 octets, one of length 2 octets, and one of length 1 octet, it would be laid out as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Option Type=Y                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |           2-octet field           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Its alignment requirement is $4n+3$, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=1 | Pad1 Option=0 | Option Type=Y |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |           2-octet field           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| PadN Option=1 | Opt Data Len=2 |           0           |           0           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Example 3

A Hop-by-Hop or Destination Options header containing both options X and Y from Examples 1 and 2 would have one of the two following formats, depending on which option appeared first:

```

+-----+
| Next Header | Hdr Ext Len=3 | Option Type=X | Opt Data Len=12 |
+-----+
|                                     4-octet field                                     |
+-----+
|                                     8-octet field                                     |
+-----+
| PadN Option=1 | Opt Data Len=1 |          0          | Option Type=Y |
+-----+
| Opt Data Len=7 | 1-octet field |          2-octet field          |
+-----+
|                                     4-octet field                                     |
+-----+
| PadN Option=1 | Opt Data Len=2 |          0          |          0          |
+-----+

+-----+
| Next Header | Hdr Ext Len=3 | Pad1 Option=0 | Option Type=Y |
+-----+
| Opt Data Len=7 | 1-octet field |          2-octet field          |
+-----+
|                                     4-octet field                                     |
+-----+
| PadN Option=1 | Opt Data Len=4 |          0          |          0          |
+-----+
|          0          |          0          | Option Type=X | Opt Data Len=12 |
+-----+
|                                     4-octet field                                     |
+-----+
|                                     8-octet field                                     |
+-----+

```

Appendix B. Changes Since RFC2460

This memo has the following changes from RFC2460.

- o Removed IP Next Generation from the Abstract.
- o Added text in Section 1 that the Data Transmission Order is the same as IPv4 as defined in RFC791.
- o Clarified the text in Section 3 about decrementing the hop limit.

- o Clarification that extension headers (except for the hop-by-hop options header) are not processed, inserted, or deleted by any node along a packet's delivery path.
- o Changed requirement for the Hop-by-Hop Options header to a may, and added a note to indicate what is expected regarding the Hop-by-Hop Options header.
- o Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.
- o Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.
- o Incorporate the updates from RFC5095 and RFC5871 to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.
- o Revised Section 4.5 on IPv6 Fragmentation based on updates from RFC5722, RFC6946 RFC7112, and RFC8021. This include:
 - Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The revised Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero).
 - Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes a clarification that no ICMP error message should be sent if overlapping fragments are received.
 - Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled, and added a new error case.
 - Added text to Fragment Header process on handling exact duplicate fragments.

- Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.
 - Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".
 - Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280.
 - Changed the text to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.
- o In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].
 - o Incorporated the update from RFC6564 to add a new Section 4.8 that describes recommendations for defining new Extension headers and options.
 - o Added text to Section 5 to define "IPv6 minimum link MTU".
 - o Simplify the text in Section 6 about Flow Labels and remove Appendix A, and instead point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].
 - o Incorporate the update in made by RFC6935 "UDP Checksums for Tunneled Packets" in Section 8. Added an exception to the default behaviour for the handling of handling UDP packets with zero checksums for tunnels.
 - o Add instruction to Section 9 "IANA Considerations" to change references to RFC2460 to this document
 - o Revised and expanded Section 10 "Security Considerations".
 - o Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents
 - o Update references to current versions and assign references to normative and informative.
 - o Changes to resolve the open Errata on RFC2460. These are:

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft.

Errata ID: 2843: This errata is marked rejected. No change was made.

B.1. Change History Since RFC2460

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

- 13) Added link to reference to RFC6564 in Section 4.8.
- 13) Added text to Section 5 to define "IPv6 minimum link MTU".
- 13) Editorial changes.
- 12) Editorial changes (remove old duplicate paragraph).
- 11) In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].
- 11) In Section 4 restructured text including separated behaviors of extension headers and the hop-by-hop option header, removed "examine" from first paragraph about extension headers, and removed reference to RFC7045 because "examine" was removed (RFC7045 is referenced in Security Considerations). Also removed "including the source and

destination nodes" from paragraph about the hop-by-hop options header.

- 11) Revised Section 4.8 to make it closer to the update done by RFC6554 that updated it and reordered the paragraphs.
- 11) Reordered items in Appendix B "Changes Since RFC2460" to match the order of the document.
- 11) Editorial changes.
- 10) Revised and expanded Security Consideration Section based on IESG Discuss comments.
- 10) Editorial changes.
- 09) Based on results of IETF last call, changed text in Section 4 to add clarification that extension headers are not examined, processed, inserted, or deleted by any node along a packet's delivery path.
- 09) Changed reference from draft-ietf-6man-rfc4291bis to RFC4291 because the bis draft won't be advanced as the same time.
- 09) Revised "Changes since RFC2460" Section to have a summary of changes since RFC2460 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.
- 09) Editorial changes.
- 08) Revised header insertion text in Section 4 based on the results of w.g. survey that concluded to describe the problems with header insertion.
- 08) Editorial changes.
- 07) Expanded Security Considerations section to include both IPsec and encryption at higher levels in the protocol stack as ways to mitigate IP level security issues.
- 07) Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.
- 07) Moved the text regarding network duplicated fragments to the received fragment error section.

- 07) Added clarification that no ICMP error message should be sent if overlapping fragments are received.
- 07) Revised the text in Section 4.8 regarding new hop-by-hop options and new Extension headers to be closer to the -05 version.
- 07) Added additional registries to the IANA Considerations section that IANA needs to update.
- 07) Editorial changes.
- 06) Added the Routing Header to the list required extension headers that a full implementation includes.
- 06) Moved the text in Section 4.5 regarding the handling of received overlapping fragments to the list of error conditions
- 06) Rewrote the text in Section 4.8 "Defining New Extension Headers and Options" to be clearer and remove redundant text.
- 06) Editorial changes.
- 05) Changed requirement for the Hop-by-Hop Options header from a should to a may, and added a note to indicate what is expected.
- 05) Corrected reference to point to draft-ietf-6man-rfc4291bis instead of draft-hinden-6man-rfc4291bis.
- 05) Change to text regarding not inserting extension headers to cite using encapsulation as an example.
- 04) Changed text discussing Fragment ID selection to refer to RFC7739 for example algorithms.
- 04) Editorial changes.
- 03) Clarified the text about decrementing the hop limit.
- 03) Removed IP Next Generation from the Abstract.
- 03) Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.
- 03) Editorial changes.

- 02) Added text to Section 4.8 "Defining New Extension Headers and Options" clarifying why no new hop by hop extension headers should be defined.
- 02) Added text to Fragment Header process on handling exact duplicate fragments.
- 02) Editorial changes.
- 01) Added text that Extension headers must never be inserted by any node other than the source of the packet.
- 01) Change "must" to "should" in Section 4.3 on the Hop-by-Hop header.
- 01) Added text that the Data Transmission Order is the same as IPv4 as defined in RFC791.
- 01) Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.
- 01) Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".
- 01) Removed paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. This is based on the update in RFC8021.
- 01) Changed to Fragmentation Header section to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.
- 01) Editorial changes.
- 00) Add instruction to the IANA to change references to RFC2460 to this document
- 00) Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents
- 00) Remove old paragraph in Section 4 that should have been removed when incorporating the update from RFC7045.
- 00) Editorial changes.

Individual Internet Drafts

- 07) Update references to current versions and assign references to normative and informative.

- 07) Editorial changes.

- 06) The purpose of this draft is to incorporate the updates dealing with Extension headers as defined in RFC6564, RFC7045, and RFC7112. The changes include:

RFC6564: Added new Section 4.8 that describe recommendations for defining new Extension headers and options

RFC7045: The changes were to add a reference to RFC7045, change the requirement for processing the hop-by-hop option to a should, and added a note that due to performance restrictions some nodes won't process the Hop-by-Hop Option header.

RFC7112: The changes were to revise the Fragmentation Section (Section 4.5) to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled and added a new error case.

- 06) Editorial changes.

- 05) The purpose of this draft is to incorporate the updates dealing with fragmentation as defined in RFC5722 and RFC6946. Note: The issue relating to the handling of exact duplicate fragments identified on the mailing list is left open.

- 05) Fix text in the end of Section 4 to correct the number of extension headers defined in this document.

- 05) Editorial changes.

- 04) The purpose of this draft is to update the document to incorporate the update made by RFC6935 "UDP Checksums for Tunneled Packets".

- 04) Remove Routing (Type 0) header from the list of required extension headers.
- 04) Editorial changes.
- 03) The purpose of this draft is to update the document for the deprecation of the RH0 Routing Header as specified in RFC5095 and the allocations guidelines for routing headers as specified in RFC5871. Both of these RFCs updated RFC2460.
- 02) The purpose of this version of the draft is to update the document to resolve the open Errata on RFC2460.

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft. Note: No change was made regarding host behaviour.

Errata ID: 2843: This errata is marked rejected. No change is required.

- 02) Editorial changes to the Flow Label and Traffic Class text.
- 01) The purpose of this version of the draft is to update the document to point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].
- 00) The purpose of this version is to establish a baseline from RFC2460. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC

and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Network Working Group
Internet-Draft
Obsoletes: 4291 (if approved)
Intended status: Standards Track
Expires: January 4, 2018

R. Hinden
Check Point Software
S. Deering
Retired
July 3, 2017

IP Version 6 Addressing Architecture
draft-ietf-6man-rfc4291bis-09

Abstract

This specification defines the addressing architecture of the IP Version 6 (IPv6) protocol. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses.

This document obsoletes RFC 4291, "IP Version 6 Addressing Architecture".

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

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

Table of Contents

1. Introduction	3
2. IPv6 Addressing	3
2.1. Addressing Model	4
2.2. Text Representation of IPv6 Addresses	4
2.2.1. Text Representation of Addresses	4
2.2.2. Text Representation of Address Prefixes	6
2.2.3. Recommendation for outputting IPv6 addresses	7
2.3. Address Type Identification	9
2.4. Unicast Addresses	10
2.4.1. Interface Identifiers	11
2.4.2. The Unspecified Address	12
2.4.3. The Loopback Address	12
2.4.4. Global Unicast Addresses	13
2.4.5. IPv6 Addresses with Embedded IPv4 Addresses	13
2.4.5.1. IPv4-Compatible IPv6 Address	13
2.4.5.2. IPv4-Mapped IPv6 Address	14
2.4.6. Link-Local IPv6 Unicast Addresses	14
2.4.7. Other Local Unicast IPv6 Addresses	14
2.5. Anycast Addresses	15
2.5.1. Required Anycast Address	15
2.6. Multicast Addresses	16
2.6.1. Pre-Defined Multicast Addresses	19
2.7. A Node's Required Addresses	20
3. IANA Considerations	21
4. Security Considerations	22
5. Acknowledgments	22
6. References	23
6.1. Normative References	23
6.2. Informative References	23

Appendix A. Modified EUI-64 Format Interface Identifiers	26
A.1. Creating Modified EUI-64 Format Interface Identifiers . .	27
Appendix B. CHANGES SINCE RFC 4291	29
B.1. Change History Since RFC4291	31
Authors' Addresses	35

1. Introduction

This specification defines the addressing architecture of the IP Version 6 protocol. It includes the basic formats for the various types of IPv6 addresses (unicast, anycast, and multicast).

2. IPv6 Addressing

IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces (where "interface" is as defined in Section 2 of [I-D.ietf-6man-rfc2460bis]). There are three types of addresses:

- Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).
- Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6, their function being superseded by multicast addresses.

In this document, fields in addresses are given a specific name, for example, "subnet". When this name is used with the term "ID" for identifier after the name (e.g., "subnet ID"), it refers to the contents of the named field. When it is used with the term "prefix" (e.g., "subnet prefix"), it refers to all of the address from the left up to and including this field.

Note: The term "prefix" is used in several different contexts for IPv6: a prefix used by a routing protocol, a prefix used by a node

to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.

In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded. Specifically, prefixes may contain, or end with, zero-valued fields.

2.1. Addressing Model

IPv6 addresses of all types are assigned to interfaces, not nodes. An IPv6 unicast address refers to a single interface. Since each interface belongs to a single node, any of that node's interfaces' unicast addresses may be used as an identifier for the node.

All interfaces are required to have at least one Link-Local unicast address (see Section 2.7 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. Unicast addresses with a scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces. There is one exception to this addressing model:

A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces.

Currently, IPv6 continues the IPv4 model in that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link. The relationship between links and IPv6 subnet prefixes differs from the IPv4 model in that all nodes automatically configure an address from the link-local prefix. A host is by definition on-link with its default router, and that unicast addresses are not automatically associated with an on-link prefix. See [RFC5942] "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes" for more details.

2.2. Text Representation of IPv6 Addresses

2.2.1. Text Representation of Addresses

There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is x:x:x:x:x:x:x:x, where the 'x's are one to four hexadecimal digits of the eight 16-bit pieces of the address. Examples:

```
abcd:ef01:2345:6789:abcd:ef01:2345:6789
2001:db8:0:0:8:800:200c:417a
```

Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.).

2. Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address.

For example, the following addresses

2001:db8:0:0:8:800:200c:417a	a unicast address
ff01:0:0:0:0:0:0:101	a multicast address
0:0:0:0:0:0:0:1	the loopback address
0:0:0:0:0:0:0:0	the unspecified address

may be represented as

2001:db8::8:800:200c:417a	a unicast address
ff01::101	a multicast address
::1	the loopback address
::	the unspecified address

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Examples:

```
0:0:0:0:0:0:13.1.68.3
0:0:0:0:0:ffff:129.144.52.38
```

or in compressed form:

```
::13.1.68.3
::ffff:129.144.52.38
```

2.2.2. Text Representation of Address Prefixes

The text representation of IPv6 address prefixes is similar to the way IPv4 address prefixes are written in Classless Inter-Domain Routing (CIDR) notation [RFC4632]. An IPv6 address prefix is represented by the notation:

```
ipv6-address/prefix-length
```

where

`ipv6-address` is an IPv6 address in any of the notations listed in Section 2.2.

`prefix-length` is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

For example, the following are legal representations of the 60-bit prefix 20010db80000cd3 (hexadecimal):

```
2001:0db8:0000:cd30:0000:0000:0000:0000/60
```

```
2001:0db8::cd30:0:0:0:0/60
```

```
2001:0db8:0:cd30::/60
```

The following are NOT legal representations of the above prefix:

```
2001:0db8:0:cd3/60    may drop leading zeros, but not trailing
                        zeros, within any 16-bit chunk of the address
```

```
2001:0db8::cd30/60    address to left of "/" expands to
                        2001:0db8:0000:0000:0000:0000:0000:cd30
```

```
2001:0db8::cd3/60     address to left of "/" expands to
                        2001:0db8:0000:0000:0000:0000:0000:0cd3
```

When writing both a node address and a prefix of that node address (e.g., the node's subnet prefix), the two can be combined as follows:

the node address	2001:0db8:0:cd30:123:4567:89ab:cdef
and its subnet prefix	2001:0db8:0:cd30::/60

can be abbreviated as 2001:0db8:0:cd30:123:4567:89ab:cdef/60

2.2.3. Recommendation for outputting IPv6 addresses

This section provides a recommendation for systems generating and outputting IPv6 addresses as text. Note, all implementations must accept and process all addresses in the formats defined in the previous two sections of this document. Background on this recommendation can be found in [RFC5952].

The recommendations are as follows:

1. The hexadecimal digits "a", "b", "c", "d", "e", and "f" in an IPv6 address must be represented in lowercase.
2. Leading zeros in a 16-Bit Field must be suppressed. For example,

2001:0db8::0001

is not correct and must be represented as

2001:db8::1

3. A single 16-bit 0000 field must be represented as 0.

The use of the symbol "::" must be used to its maximum capability. For example:

2001:db8:0:0:0:0:2:1

must be shortened to

2001:db8::2:1

Likewise,

2001:db8::0:1

is not correct, because the symbol "::" could have been used to produce a shorter representation

2001:db8::1.

4. When there is an alternative choice in the placement of a "::", the longest run of consecutive 16-bit 0 fields must be shortened, that is, in

2001:0:0:1:0:0:0:1

the sequence with three consecutive zero fields is shortened to

2001:0:0:1::1

5. When the length of the consecutive 16-bit 0 fields are equal, for example

2001:db8:0:0:1:0:0:1

the first sequence of zero bits must be shortened. For example

2001:db8::1:0:0:1

is the correct representation.

6. The symbol "::" must not be used to shorten just one 16-bit 0 field. For example, the representation

2001:db8:0:1:1:1:1:1

is correct, but

2001:db8::1:1:1:1:1

is not correct.

7. The text representation method describe in this section should also be use for text Representation of IPv6 Address Prefixes. For example

2001:0db8:0000:cd30:0000:0000:0000/60

should be shown as

2001:0db8:0:cd30::/60

8. The text representation method describe in this section should be applied for IPv6 addresses with embedded IPv4 address. For example

0:0:0:0:0:ffff:192.0.2.1

should be shown as

::ffff:192.0.2.1

2.3. Address Type Identification

The type of an IPv6 address is identified by the high-order bits of the address, as follows:

Address type	Binary prefix	IPv6 notation	Section
-----	-----	-----	-----
Unspecified	00...0 (128 bits)	::/128	2.4.2
Loopback	00...1 (128 bits)	::1/128	2.4.3
Multicast	11111111	ff00::/8	2.6
Link-Local unicast	1111111010	fe80::/10	2.4.6
Global Unicast	(everything else)		

Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses.

The general format of Global Unicast addresses is described in Section 2.4.4. Some special-purpose subtypes of Global Unicast addresses that contain embedded IPv4 addresses (for the purposes of IPv4-IPv6 interoperation) are described in Section 2.4.5.

Future specifications may redefine one or more sub-ranges of the Global Unicast space for other purposes, but unless and until that happens, implementations must treat all addresses that do not start with any of the above-listed prefixes as Global Unicast addresses.

The current assigned IPv6 prefixes and references to their usage can be found in the IANA Internet Protocol Version 6 Address Space registry [IANA-AD] and the IANA IPv6 Special-Purpose Address Registry [IANA-SP].

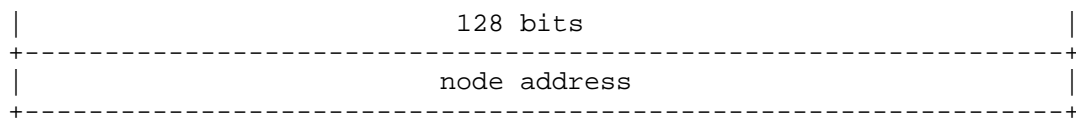
2.4. Unicast Addresses

IPv6 unicast addresses are aggregatable with prefixes of arbitrary bit-length, similar to IPv4 addresses under Classless Inter-Domain Routing.

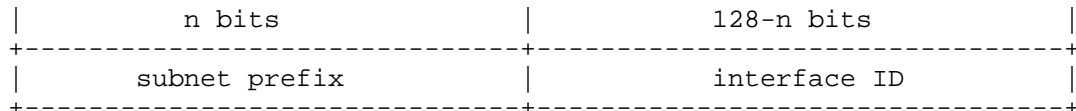
IPv6 unicast routing is based on prefixes of any valid length up to 128 [BCP198].

There are several types of unicast addresses in IPv6, in particular, Global Unicast, Local unicast, and Link-Local unicast. There are also some special-purpose subtypes of Global Unicast, such as IPv6 addresses with embedded IPv4 addresses. Additional address types or subtypes can be defined in the future.

IPv6 nodes may have considerable or little knowledge of the internal structure of the IPv6 address, depending on the role the node plays (for instance, host versus router). At a minimum, a node may consider that unicast addresses (including its own) have no internal structure:



A slightly more complex node may additionally be aware of subnet prefix(es) for the link(s) it is attached to, where different addresses may have different values for n:



Though a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy.

Except for the knowledge of the subnet boundary discussed in the previous paragraphs, nodes should not make any assumptions about the structure of an IPv6 address.

2.4.1. Interface Identifiers

Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link. They are required to be unique within a subnet prefix. It is recommended that the same interface identifier not be assigned to different nodes on a link. They may also be unique over a broader scope. The same interface identifier may be used on multiple interfaces on a single node, as long as they are attached to different subnets.

Interface IDs must be viewed outside of the node that created Interface ID as an opaque bit string without any internal structure.

Note that the uniqueness of interface identifiers is independent of the uniqueness of IPv6 addresses. For example, a Global Unicast address may be created with an interface identifier that is only unique on a single subnet, and a Link-Local address may be created with interface identifier that is unique over multiple subnets.

Interface Identifiers are 64 bit long except if the first three bits of the address are 000, or when the addresses are manually configured, or by exceptions defined in standards track documents. The rationale for using 64 bit Interface Identifiers can be found in

[RFC7421]. An example of a standards track exception is [RFC6164] that standardises 127 bit prefixes on inter-router point-to-point links.

The details of forming interface identifiers are defined in other specifications, such as "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941] or "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)" [RFC7217]. Specific cases are described in appropriate "IPv6 over <link>" specifications, such as "IPv6 over Ethernet" [RFC2464] and "Transmission of IPv6 Packets over ITU-T G.9959 Networks" [RFC7428]. The security and privacy considerations for IPv6 address generation is described in [RFC7721].

Earlier versions of this document described a method of forming interface identifiers derived from IEEE MAC-layer addresses called Modified EUI-64 format. These are described in Appendix A and are no longer recommended.

2.4.2. The Unspecified Address

The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing headers. An IPv6 packet with a source address of unspecified must never be forwarded by an IPv6 router.

2.4.3. The Loopback Address

The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It must not be assigned to any physical interface. It is treated as having Link-Local scope, and may be thought of as the Link-Local unicast address of a virtual interface (typically called the "loopback interface") to an imaginary link that goes nowhere.

The loopback address must not be used as the source address in IPv6 packets that are sent outside of a single node. An IPv6 packet with a destination address of loopback must never be sent outside of a single node and must never be forwarded by an IPv6 router. A packet received on an interface with a destination address of loopback must be dropped.

2.4.4. Global Unicast Addresses

The general format for IPv6 Global Unicast addresses is as follows:

	n bits		m bits		128-n-m bits	
+	-----	+	-----	+	-----	+
	global routing prefix		subnet ID		interface ID	
+	-----	+	-----	+	-----	+

where the global routing prefix is a (typically hierarchically-structured) value assigned to a site (a cluster of subnets/links), the subnet ID is an identifier of a link within the site, and the interface ID is as defined in Section 2.4.1.

Examples of Global Unicast addresses that start with binary 000 are the IPv6 address with embedded IPv4 addresses described in Section 2.4.5. An example of global addresses starting with a binary value other than 000 (and therefore having a 64-bit interface ID field) can be found in [RFC3587].

2.4.5. IPv6 Addresses with Embedded IPv4 Addresses

Two types of IPv6 addresses are defined that carry an IPv4 address in the low-order 32 bits of the address. These are the "IPv4-Compatible IPv6 address" and the "IPv4-mapped IPv6 address".

2.4.5.1. IPv4-Compatible IPv6 Address

The "IPv4-Compatible IPv6 address" was defined to assist in the IPv6 transition. The format of the "IPv4-Compatible IPv6 address" is as follows:

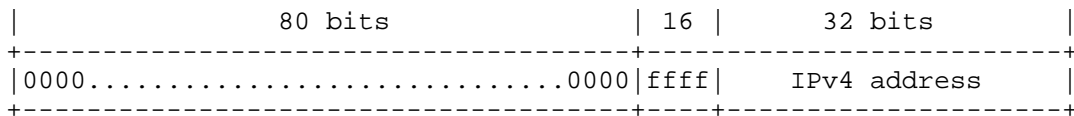
	80 bits		16		32 bits	
+	-----	+	-----	+	-----	+
	0000.....0000		0000		IPv4 address	
+	-----	+	-----	+	-----	+

Note: The IPv4 address used in the "IPv4-Compatible IPv6 address" must be a globally-unique IPv4 unicast address.

The "IPv4-Compatible IPv6 address" is now deprecated because the current IPv6 transition mechanisms no longer use these addresses. New or updated implementations are not required to support this address type.

2.4.5.2. IPv4-Mapped IPv6 Address

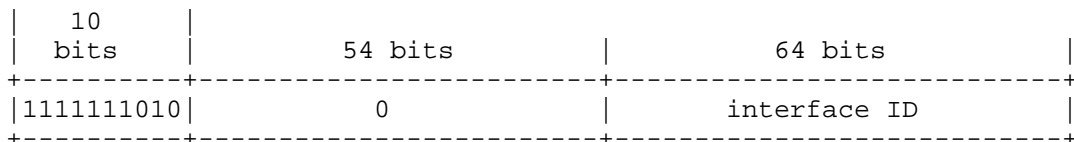
A second type of IPv6 address that holds an embedded IPv4 address is defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. The format of the "IPv4-mapped IPv6 address" is as follows:



See [RFC4038] for background on the usage of the "IPv4-mapped IPv6 address".

2.4.6. Link-Local IPv6 Unicast Addresses

Link-Local addresses are for use on a single link. Link-Local addresses have the following format:



Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present.

Routers must not forward any packets with Link-Local source or destination addresses to other links.

2.4.7. Other Local Unicast IPv6 Addresses

Unique Local Addresses (ULA) [RFC4193], the current form of Local IPv6 Addresses, are intended to be used for local communications, have global unicast scope, and are not expected to be routable on the global Internet.

Site-Local addresses, deprecated by [RFC3879], the previous form of Local IPv6 Addresses, were originally designed to be used for addressing inside of a site without the need for a global prefix.

The special behavior of Site-Local defined in [RFC3513] must no longer be supported in new implementations (i.e., new implementations must treat this prefix as Global Unicast). Existing implementations and deployments may continue to use this prefix.

2.5. Anycast Addresses

An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the "nearest" interface having that address, according to the routing protocols' measure of distance.

Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address.

For any assigned anycast address, there is a longest prefix P of that address that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, the anycast address must be maintained as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing entry for prefix P.

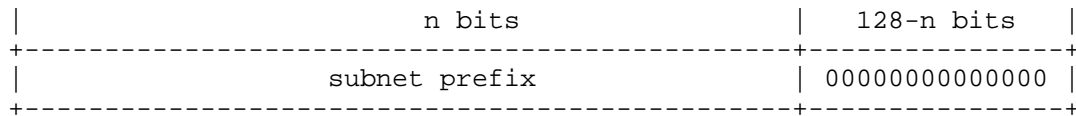
Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire Internet, which presents a severe scaling limit on how many such "global" anycast sets may be supported. Therefore, it is expected that support for global anycast sets may be unavailable or very restricted.

One expected use of anycast addresses is to identify the set of routers belonging to an organization providing Internet service. Such addresses could be used as intermediate addresses in an IPv6 Routing header, to cause a packet to be delivered via a particular service provider or sequence of service providers.

Some other possible uses are to identify the set of routers attached to a particular subnet, or the set of routers providing entry into a particular routing domain.

2.5.1. Required Anycast Address

The Subnet-Router anycast address is predefined. Its format is as follows:



The "subnet prefix" in an anycast address is the prefix that identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero.

Packets sent to the Subnet-Router anycast address will be delivered to one router on the subnet. All routers are required to support the Subnet-Router anycast addresses for the subnets to which they have interfaces.

The Subnet-Router anycast address is intended to be used for applications where a node needs to communicate with any one of the set of routers.

2.6. Multicast Addresses

An IPv6 multicast address is an identifier for a group of interfaces (typically on different nodes). An interface may belong to any number of multicast groups. Multicast addresses have the following format:



binary 11111111 at the start of the address identifies the address as being a multicast address.

flgs is a set of 4 flags:

+	+	+	+	+	+
	0		R		P
					T
+	+	+	+	+	+

The high-order flag is reserved, and must be initialized to 0.

T = 0 indicates a permanently-assigned ("well-known") multicast address, assigned by the Internet Assigned Numbers Authority (IANA).

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

The P flag's definition and usage can be found in [RFC3306].

The R flag's definition and usage can be found in [RFC3956].

scop is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are as follows:

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

Interface-Local scope spans only a single interface on a node and is useful only for loopback transmission of multicast. Packets with interface-local scope received from another node must be discarded.

Link-Local multicast scope spans the same topological region as the corresponding unicast scope.

Interface-Local, Link-Local, and Realm-Local scope boundaries are automatically derived from physical connectivity or other non-multicast-related configurations. Global scope has no boundary. The boundaries of all other non-reserved scopes of Admin-Local or larger are administratively configured. For reserved scopes, the way of configuring their boundaries will be defined when the semantics of the scope are defined.

According to [RFC4007], the zone of a Realm-Local scope must fall within zones of larger scope. Because the zone of a Realm-Local scope is configured automatically while the zones of larger scopes are configured manually, care must be taken in the definition of those larger scopes to ensure that the inclusion constraint is met.

Realm-Local scopes created by different network technologies are considered to be independent and will have different zone indices (see Section 6 of [RFC4007]). A router with interfaces on links using different network technologies does not forward traffic between the Realm-Local multicast scopes defined by those technologies.

Site-Local scope is intended to span a single site.

Organization-Local scope is intended to span multiple sites belonging to a single organization.

scopes labeled "(unassigned)" are available for administrators to define additional multicast regions.

group ID identifies the multicast group, either permanent or transient, within the given scope. Additional definitions of the multicast group ID field structure are provided in [RFC3306].

The "meaning" of a permanently-assigned multicast address is independent of the scope value. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 101 (hex), then

ff01:0:0:0:0:0:0:101 means all NTP servers on the same interface (i.e., the same node) as the sender.

ff02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender.

ff05:0:0:0:0:0:0:101 means all NTP servers in the same site as the sender.

ff0e:0:0:0:0:0:0:101 means all NTP servers in the Internet.

Non-permanently-assigned multicast addresses are meaningful only within a given scope. For example, a group identified by the non-permanent, site-local multicast address ff15:0:0:0:0:0:0:101 at one site bears no relationship to a group using the same address at a different site, nor to a non-permanent group using the same group ID with a different scope, nor to a permanent group with the same group ID.

Multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header.

Routers must not forward any multicast packets beyond the scope indicated by the scop field in the destination multicast address.

Nodes must not originate a packet to a multicast address whose scop field contains the reserved value 0; if such a packet is received, it must be silently dropped. Nodes should not originate a packet to a multicast address whose scop field contains the reserved value F; if such a packet is sent or received, it must be treated the same as packets destined to a global (scop E) multicast address.

2.6.1. Pre-Defined Multicast Addresses

The following well-known multicast addresses are pre-defined. The group IDs defined in this section are defined for explicit scope values.

Use of these group IDs for any other scope values, with the T flag equal to 0, is not allowed.

```
Reserved Multicast Addresses: ff00:0:0:0:0:0:0:0
                             ff01:0:0:0:0:0:0:0
                             ff02:0:0:0:0:0:0:0
                             ff03:0:0:0:0:0:0:0
                             ff04:0:0:0:0:0:0:0
                             ff05:0:0:0:0:0:0:0
                             ff06:0:0:0:0:0:0:0
                             ff07:0:0:0:0:0:0:0
                             ff08:0:0:0:0:0:0:0
                             ff09:0:0:0:0:0:0:0
                             ff0a:0:0:0:0:0:0:0
                             ff0b:0:0:0:0:0:0:0
                             ff0c:0:0:0:0:0:0:0
                             ff0d:0:0:0:0:0:0:0
                             ff0e:0:0:0:0:0:0:0
                             ff0f:0:0:0:0:0:0:0
```

The above multicast addresses are reserved and shall never be assigned to any multicast group.

```
All Nodes Addresses:      ff01:0:0:0:0:0:0:1
                          ff02:0:0:0:0:0:0:1
```

The above multicast addresses identify the group of all IPv6 nodes, within scope 1 (interface-local) or 2 (link-local).

All Routers Addresses: ff01:0:0:0:0:0:0:2
 ff02:0:0:0:0:0:0:2
 ff05:0:0:0:0:0:0:2

The above multicast addresses identify the group of all IPv6 routers, within scope 1 (interface-local), 2 (link-local), or 5 (site-local).

Solicited-Node Address: ff02:0:0:0:0:1:ffxx:xxxx

Solicited-Node multicast address are computed as a function of a node's unicast and anycast addresses. A Solicited-Node multicast address is formed by taking the low-order 24 bits of an address (unicast or anycast) and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range

ff02:0:0:0:0:1:ff00:0000

to

ff02:0:0:0:0:1:ffff:ffff

For example, the Solicited-Node multicast address corresponding to the IPv6 address 4037::01:800:200e:8c6c is ff02::1:ff0e:8c6c. IPv6 addresses that differ only in the high-order bits (e.g., due to multiple high-order prefixes associated with different aggregations) will map to the same Solicited-Node address, thereby reducing the number of multicast addresses a node must join.

A node is required to compute and join (on the appropriate interface) the associated Solicited-Node multicast addresses for all unicast and anycast addresses that have been configured for the node's interfaces (manually or automatically).

Additional defined multicast address can be found in the IANA IPv6 Multicast Address Allocation registry [IANA-MC]

2.7. A Node's Required Addresses

A host is required to recognize the following addresses as identifying itself:

- o Its required Link-Local address for each interface.

- o Any additional Unicast and Anycast addresses that have been configured for the node's interfaces (manually or automatically).
- o The loopback address.
- o The All-Nodes multicast addresses defined in Section 2.6.1.
- o The Solicited-Node multicast address for each of its unicast and anycast addresses.
- o Multicast addresses of all other groups to which the node belongs.

A router is required to recognize all addresses that a host is required to recognize, plus the following addresses as identifying itself:

- o The Subnet-Router Anycast addresses for all interfaces for which it is configured to act as a router.
- o All other Anycast addresses with which the router has been configured.
- o The All-Routers multicast addresses defined in Section 2.6.1.

3. IANA Considerations

RFC4291 is referenced in a number of IANA registries. These include:

- o Internet Protocol Version 6 Address Space [IANA-AD]
- o IPv6 Global Unicast Address Assignments [IANA-GU]
- o IPv6 Multicast Address Space Registry [IANA-MC]
- o Application for an IPv6 Multicast Address [IANA-MA]
- o Internet Protocol Version 6 (IPv6) Anycast Addresses [IANA-AC]
- o IANA IPv6 Special-Purpose Address Registry [IANA-SP]
- o Reserved IPv6 Interface Identifiers [IANA-ID]

- o Number Resources [IANA-NR]
- o Protocol Registries [IANA-PR]
- o Technical requirements for authoritative name servers [IANA-NS]
- o IP Flow Information Export (IPFIX) Entities [IANA-FE]

The IANA should update these references to point to this document.

There are also other references in IANA procedures documents that the IANA should investigate to see if they should be updated.

4. Security Considerations

IPv6 addressing documents do not have any direct impact on Internet infrastructure security. Authentication of IPv6 packets is defined in [RFC4302].

One area relevant to IPv6 addressing is privacy. IPv6 addresses can be created using interface identifiers constructed with unique stable tokens. The addresses created in this manner can be used to track the movement of devices across the Internet. Since earlier versions of this document were published, several approaches have been developed that mitigate these problems. These are described in "Security and Privacy Considerations for IPv6 Address Generation Mechanisms" [RFC7721], "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [RFC4941], and "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)" [RFC7217].

5. Acknowledgments

The authors would like to acknowledge the contributions of Bill Simpson, Bob Fink, Bob Gilligan, Brian Carpenter, Christian Huitema, Deborah Estrin, Dimitry Haskin, Erik Nordmark, Greg Minshall, James Woodyatt, Jim Bound, Jun-ichiro Itojun Hagino, Larry Masinter, Mahmood Ali, Markku Savela, Matt Crawford, Paul Francis, Peter Ford, Roger Fajman, Scott Bradner, Sue Thomson, Suresh Krishnan, Tatuya Jinmei, Thomas Narten, Tom Harsch, Tony Li, and Yakov Rekhter.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move IPv6 to Internet Standard. This includes Marcelo Bagnulo, Congxiao Bao, Mohamed Boucadair, Brian Carpenter, Ralph Droms, Christian Huitema, Sheng Jiang, Seiichi Kawamura, Masanobu Kawashima, Xing Li, and Stig Venaas.

6. References

6.1. Normative References

- [I-D.ietf-6man-rfc2460bis]
Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-13 (work in progress), May 2017.

6.2. Informative References

- [BCP198] Boucadair, M., Petrescu, A., and F. Baker, "IPv6 Prefix Length Recommendation for Forwarding", BCP 198, RFC 7608, DOI 10.17487/RFC7608, July 2015, <<http://www.rfc-editor.org/info/rfc7608>>.
- [EUI64] "IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority"", March 1997, <<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>>.
- [IANA-AC] "Internet Protocol Version 6 (IPv6) Anycast Addresses", <<http://www.iana.org/assignments/ipv6-anycast-addresses/ipv6-anycast-addresses.xhtml>>.
- [IANA-AD] "Internet Protocol Version 6 Address Space", <<https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml>>.
- [IANA-FE] "IP Flow Information Export (IPFIX) Entities", <<http://www.iana.org/assignments/ipfix/ipfix.xhtml>>.
- [IANA-GU] "IPv6 Global Unicast Address Assignments", <<http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>>.
- [IANA-ID] "IANA IPv6 Special-Purpose Address Registry", <<http://www.iana.org/assignments/ipv6-interface-ids/ipv6-interface-ids.xhtml>>.
- [IANA-MA] "Application for an IPv6 Multicast Address", <<https://www.iana.org/form/multicast-ipv6>>.
- [IANA-MC] "IPv6 Multicast Address Space Registry", <<http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>>.
- [IANA-NR] "Number Resources", <<http://https://www.iana.org/numbers>>.

- [IANA-NS] "Technical requirements for authoritative name servers",
<<https://www.iana.org/help/nameserver-requirements>>.
- [IANA-PR] "Protocol Registries", <<https://www.iana.org/protocols>>.
- [IANA-SP] "IANA IPv6 Special-Purpose Address Registry",
<<https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998,
<<http://www.rfc-editor.org/info/rfc2464>>.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306, August 2002, <<http://www.rfc-editor.org/info/rfc3306>>.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003,
<<http://www.rfc-editor.org/info/rfc3513>>.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <<http://www.rfc-editor.org/info/rfc3587>>.
- [RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <<http://www.rfc-editor.org/info/rfc3879>>.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004,
<<http://www.rfc-editor.org/info/rfc3956>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005,
<<http://www.rfc-editor.org/info/rfc4007>>.
- [RFC4038] Shin, M-K., Ed., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, DOI 10.17487/RFC4038, March 2005,
<<http://www.rfc-editor.org/info/rfc4038>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005,
<<http://www.rfc-editor.org/info/rfc4193>>.

- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<http://www.rfc-editor.org/info/rfc4632>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC5942] Singh, H., Beebe, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <<http://www.rfc-editor.org/info/rfc5942>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<http://www.rfc-editor.org/info/rfc5952>>.
- [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, DOI 10.17487/RFC6164, April 2011, <<http://www.rfc-editor.org/info/rfc6164>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<http://www.rfc-editor.org/info/rfc7421>>.
- [RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <<http://www.rfc-editor.org/info/rfc7428>>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<http://www.rfc-editor.org/info/rfc7721>>.

Appendix A. Modified EUI-64 Format Interface Identifiers

Modified EUI-64 format-based interface identifiers may have universal scope when derived from a universal token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]) or may have local scope where a global token is not being used (e.g., serial links, tunnel end-points) or where global tokens are undesirable (e.g., temporary tokens for privacy [RFC4941]).

Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format, the "u" bit is set to one (1) to indicate universal scope, and it is set to zero (0) to indicate local scope. The first three octets in binary of an IEEE EUI-64 identifier are as follows:

0	0 0	1 1	2
0	7 8	5 6	3
+-----+-----+-----+-----+-----+			
cccc	ccug	cccc	cccc cccc
+-----+-----+-----+-----+-----+			

written in Internet standard bit-order, where "u" is the universal/local bit, "g" is the individual/group bit, and "c" is the bits of the company_id. Appendix A, "Creating Modified EUI-64 Format Interface Identifiers", provides examples on the creation of Modified EUI-64 format-based interface identifiers.

The motivation for inverting the "u" bit when forming an interface identifier is to make it easy for system administrators to hand configure non-global identifiers when hardware tokens are not available. This is expected to be the case for serial links and tunnel end-points, for example. The alternative would have been for these to be of the form 0200:0:0:1, 0200:0:0:2, etc., instead of the much simpler 0:0:0:1, 0:0:0:2, etc.

IPv6 nodes are not required to validate that interface identifiers created with modified EUI-64 tokens with the "u" bit set to universal are unique.

A.1. Creating Modified EUI-64 Format Interface Identifiers

Depending on the characteristics of a specific link or node, there are a number of approaches for creating Modified EUI-64 format interface identifiers. This appendix describes some of these approaches.

Links or Nodes with IEEE EUI-64 Identifiers

The only change needed to transform an IEEE EUI-64 identifier to an interface identifier is to invert the "u" (universal/local) bit. An example is a globally unique IEEE EUI-64 identifier of the form:

0	1	3	4	6
0	5	1	7	3
+-----+				
cccccc0gcccccccc ccccccccmmmmmmmmmm mmmmmmmmmmmmmmmmmmmm mmmmmmmmmmmmmmmmmmmm				
+-----+				

where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate universal scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The IPv6 interface identifier would be of the form:

0	1	3	4	6
0	5	1	7	3
+-----+				
cccccc1gcccccccc ccccccccmmmmmmmmmm mmmmmmmmmmmmmmmmmmmm mmmmmmmmmmmmmmmmmmmm				
+-----+				

The only change is inverting the value of the universal/local bit.

Links or Nodes with IEEE 802 48-bit MACs

[EUI64] defines a method to create an IEEE EUI-64 identifier from an IEEE 48-bit MAC identifier. This is to insert two octets, with hexadecimal values of 0xFF and 0xFE (see the Note at the end of appendix), in the middle of the 48-bit MAC (between the company_id and vendor-supplied id). An example is the 48-bit IEEE MAC with Global scope:

0	1	3	4
0	5	1	7
+-----+			
cccccc0gcccccccc ccccccccmmmmmmmmmm mmmmmmmmmmmmmmmmmmmm			
+-----+			

where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate Global scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The interface identifier would be of the form:

0	1	3	4	6
0	5	1	7	3

ccccccclgcccccccc cccccccccl11111111 11111110mmmmmmmm mmmmmmmmmmmmmmmmmmmm				

When IEEE 802 48-bit MAC addresses are available (on an interface or a node), an implementation may use them to create interface identifiers due to their availability and uniqueness properties.

Links with Other Kinds of Identifiers

There are a number of types of links that have link-layer interface identifiers other than IEEE EUI-64 or IEEE 802 48-bit MACs. Examples include LocalTalk and Arcnet. The method to create a Modified EUI-64 format identifier is to take the link identifier (e.g., the LocalTalk 8-bit node identifier) and zero fill it to the left. For example, a LocalTalk 8-bit node identifier of hexadecimal value 0x4F results in the following interface identifier:

0	1	3	4	6
0	5	1	7	3

0000000000000000 0000000000000000 0000000000000000 0000000001001111				

Note that this results in the universal/local bit set to "0" to indicate local scope.

Links without Identifiers

There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers that are unique within a subnet prefix must be chosen.

When no built-in identifier is available on a link, the preferred approach is to use a universal interface identifier from another interface or one that is assigned to the node itself. When using this approach, no other interface connecting the same node to the same subnet prefix may use the same identifier.

If there is no universal interface identifier available for use on the link, the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include the following:

- Manual Configuration
- Node Serial Number
- Other Node-Specific Token

The subnet-prefix-unique interface identifier should be generated in a manner such that it does not change after a reboot of a node or if interfaces are added or deleted from the node.

The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

Note: [EUI64] actually defines 0xFF and 0xFE as the bits to be inserted to create an IEEE EUI-64 identifier from an IEEE MAC-48 identifier. The 0xFF and 0xFE values are used when starting with an IEEE EUI-48 identifier. The incorrect value was used in earlier versions of the specification due to a misunderstanding about the differences between IEEE MAC-48 and EUI-48 identifiers.

This document purposely continues the use of 0xFF and 0xFE because it meets the requirements for IPv6 interface identifiers (i.e., that they must be unique on the link), IEEE EUI-48 and MAC-48 identifiers are syntactically equivalent, and that it doesn't cause any problems in practice.

Appendix B. CHANGES SINCE RFC 4291

This document has the following changes from RFC4291, "IP Version 6 Addressing Architecture":

- o Added Note: to Section 2 that the term "prefix" is used in different contexts in IPv6: a prefix used by a routing protocol, a prefix used by a node to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.
- o Added text to the last paragraph in Section 2.1 to clarify the differences on how subnets are handled in IPv4 and IPv6, includes

a reference to RFC5942 "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes".

- o Incorporate the updates made by RFC5952 in Section 2.2.3 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case. This includes a reference to RFC5952.
- o Incorporate the updates made by RFC6052. The change was to add a text in Section 2.3 that points to the IANA registries that records the prefix defined in RFC6052 and a number of other special use prefixes.
- o Clarified text that 64 bit Interface IDs are used except when the first three bits of the address are 000, or addresses are manually configured, or when defined by a standard track document. Added text that Modified EUI-64 identifiers not recommended and moved the text describing the format to Appendix A. This text was moved from Section 2.4 and is now consolidated in Section 2.4.1. Also removed text in Section 2.4.4 relating to 64 bit Interface IDs.
- o Added text to Section 2.4 summarizing IPv6 unicast routing and referencing BCP198, citing RFC6164 as an example of longer prefixes, and that IIDs are required to be 64 bits long as described in RFC7421.
- o Incorporate the updates made by RFC7136 to deprecate the U and G bits in Modified EUI-64 format Internet IDs.
- o Rename Section 2.4.7 to "Other Local Unicast Addresses" and rewrote the text to point to ULAs and say that Site-Local addresses were deprecated by RFC3879. The format of Site-Local was removed.
- o Incorporate the updates made by RFC7346. The change was to add Realm-Local scope to the multicast scope table in Section 2.6, and add the updating text to the same section.
- o Added a reference to the IANA Multicast address registry in Section 2.6.1.
- o Added instructions in IANA Considerations to update references in the IANA registries that currently point to RFC4291 to point to this document.
- o Expanded Security Considerations Section to discuss privacy issues related to using stable interface identifiers to create IPv6

addresses, and reference solutions that mitigate these issues such as RFC7721, RFC4941, RFC7271.

- o Add note to Section 5 section acknowledging the authors of the updating documents.
- o Updates to resolve the open Errata on RFC4291. These are:

Errata ID: 3480: Corrects the definition of Interface-Local multicast scope to also state that packets with interface-local scope received from another node must be discarded.

Errata ID: 1627: Remove extraneous "of" in Section 2.7.

Errata ID: 2702: This errata is marked rejected. No change is required.

Errata ID: 2735: This errata is marked rejected. No change is required.

Errata ID: 4406: This errata is marked rejected. No change is required.

Errata ID: 2406: This errata is marked rejected. No change is required.

Errata ID: 863: This errata is marked rejected. No change is required.

Errata ID: 864: This errata is marked rejected. No change is required.

Errata ID: 866: This errata is marked rejected. No change is required.

- o Editorial changes.

B.1. Change History Since RFC4291

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

- 09) Added text to the last paragraph in Section 2.1 to clarify the differences on how subnets are handled in IPv4 and IPv6, includes a reference to RFC5942 "The IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes".
- 09) Removed short paragraph about manual configuration in Section 2.4.1 that was added in the -08 version.
- 09) Revised "Changes since RFC4291" Section to have a summary of changes since RFC4291 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.
- 09) Editorial changes.
- 08) Added Note: to Section 2 that the term "prefix" is used in different contexts in IPv6: a prefix used by a routing protocol, a prefix used by a node to determine if another node is connected to the same link, and a prefix used to construct the complete address of a node.
- 08) Based on results of IETF last call and extensive w.g. list discussion, revised text to clarify that 64 bit Interface IDs are used except when the first three bits of the address are 000, or addresses are manually configured, or when defined by a standard track document. This text was moved from Section 2.4 and is now consolidated in Section 2.4.1 Also removed text in Section 2.4.4 relating to 64 bit Interface IDs.
- 08) Removed instruction to IANA fix error in Port Number assignment. IANA fixed the error on 4 March 2017.
- 08) Editorial changes.
- 07) Added text to Section 2.4 summarizing IPv6 unicast routing and referencing BCP198, citing RFC6164 as an example of longer prefixes, and that IIDs are required to be 64 bits long as described in RFC7421.
- 07) Based on review by Brian Haberman added reference to RFC5952 in Section 2.2.3, corrected case errors in Section 2.6.1, and added a reference to the IANA Multicast address registry in Section 2.6.1.

- 07) Corrected errors in Section 2.2.3 where the examples in 7. and 8. were reversed.
- 07) Editorial changes.
- 06) Editorial changes.
- 05) Expanded Security Considerations Section to discuss privacy issues related to using stable interface identifiers to create IPv6 addresses, and reference solutions that mitigate these issues such as RFC7721, RFC4941, RFC7271.
- 05) Added instructions in IANA Considerations to update references in the IANA registries that currently point to RFC4291 to point to this document.
- 05) Rename Section 2.4.7 to "Other Local Unicast Addresses" and rewrote the text to point to ULAs and say that Site-Local addresses were deprecated by RFC3879. The format of Site-Local was removed.
- 05) Added to Section 2.4.1 a reference to RFC7421 regarding the background on the 64 bit boundary in Interface Identifiers.
- 05) Editorial changes.
- 04) Added text and a pointer to the ULA specification in Section 2.4.7
- 04) Removed old IANA Considerations text, this was left from the baseline text from RFC4291 and should have been removed earlier.
- 04) Editorial changes.
- 03) Changes references in Section 2.4.1 that describes the details of forming IIDs to RFC7271 and RFC7721.
- 02) Remove changes made by RFC7371 because there isn't any known implementation experience.
- 01) Revised Section 2.4.1 on Interface Identifiers to reflect current approach, this included saying Modified EUI-64 identifiers not recommended and moved the text describing the format to Appendix A.
- 01) Editorial changes.

00) Working Group Draft.

00) Editorial changes.

Individual Internet Drafts

06) Incorporate the updates made by RFC7371. The changes were to the flag bits and their definitions in Section 2.6.

05) Incorporate the updates made by RFC7346. The change was to add Realm-Local scope to the multicast scope table in Section 2.6, and add the updating text to the same section.

04) Incorporate the updates made by RFC6052. The change was to add a text in Section 2.3 that points to the IANA registries that records the prefix defined in RFC6052 and a number of other special use prefixes.

03) Incorporate the updates made by RFC7136 to deprecate the U and G bits in Modified EUI-64 format Internet IDs.

03) Add note to the reference section acknowledging the authors of the updating documents.

03) Editorial changes.

02) Updates to resolve the open Errata on RFC4291. These are:

Errata ID: 3480: Corrects the definition of Interface-Local multicast scope to also state that packets with interface-local scope received from another node must be discarded.

Errata ID: 1627: Remove extraneous "of" in Section 2.7.

Errata ID: 2702: This errata is marked rejected. No change is required.

Errata ID: 2735: This errata is marked rejected. No change is required.

Errata ID: 4406: This errata is marked rejected. No change is required.

Errata ID: 2406: This errata is marked rejected. No change is required.

Errata ID: 863: This errata is marked rejected. No change is required.

Errata ID: 864: This errata is marked rejected. No change is required.

Errata ID: 866: This errata is marked rejected. No change is required.

02) Update references to current versions.

02) Editorial changes.

01) Incorporate the updates made by RFC5952 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case.

01) Revise this Section to document to show the changes from RFC4291.

01) Editorial changes.

00) Establish a baseline from RFC4291. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2020

C. Filsfils, Ed.
D. Dukes, Ed.
Cisco Systems, Inc.
S. Previdi
Huawei
J. Leddy
Individual
S. Matsushima
Softbank
D. Voyer
Bell Canada
October 22, 2019

IPv6 Segment Routing Header (SRH)
draft-ietf-6man-segment-routing-header-26

Abstract

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

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 <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on April 24, 2020.

Copyright Notice

Copyright (c) 2019 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

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Segment Routing Header	4
2.1. SRH TLVs	6
2.1.1. Padding TLVs	8
2.1.2. HMAC TLV	9
3. SR Nodes	12
3.1. Source SR Node	12
3.2. Transit Node	12
3.3. SR Segment Endpoint Node	12
4. Packet Processing	13
4.1. Source SR Node	13
4.1.1. Reduced SRH	13
4.2. Transit Node	14
4.3. SR Segment Endpoint Node	14
4.3.1. FIB Entry Is Locally Instantiated SRv6 SID	14
4.3.2. FIB Entry Is A Local Interface	16
4.3.3. FIB Entry Is A Non-Local Route	17
4.3.4. FIB Entry Is A No Match	17
5. Intra SR Domain Deployment Model	17
5.1. Securing the SR Domain	17
5.2. SR Domain as A Single System with Delegation Among Components	18
5.3. MTU Considerations	19
5.4. ICMP Error Processing	19
5.5. Load Balancing and ECMP	19
5.6. Other Deployments	20
6. Illustrations	20
6.1. Abstract Representation of an SRH	20
6.2. Example Topology	21
6.3. Source SR Node	22
6.3.1. Intra SR Domain Packet	22
6.3.2. Inter SR Domain Packet - Transit	22
6.3.3. Inter SR Domain Packet - Internal to External	23
6.4. Transit Node	23
6.5. SR Segment Endpoint Node	23
6.6. Delegation of Function with HMAC Verification	23
6.6.1. SID List Verification	24

7.	Security Considerations	24
7.1.	Source Routing Attacks	25
7.2.	Service Theft	25
7.3.	Topology Disclosure	25
7.4.	ICMP Generation	26
7.5.	Applicability of AH	26
8.	IANA Considerations	26
8.1.	Segment Routing Header Flags Registry	27
8.2.	Segment Routing Header TLVs Registry	27
9.	Implementation Status	27
9.1.	Linux	28
9.2.	Cisco Systems	28
9.3.	FD.io	28
9.4.	Barefoot	28
9.5.	Juniper	28
9.6.	Huawei	29
10.	Contributors	29
11.	Acknowledgements	29
12.	References	29
12.1.	Normative References	29
12.2.	Informative References	31
	Authors' Addresses	32

1. Introduction

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

The Segment Routing Architecture [RFC8402] describes Segment Routing and its instantiation in two data planes; MPLS and IPv6.

The encoding of IPv6 segments in the Segment Routing Header is defined in this document.

This document uses the terms Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy as defined in [RFC8402].

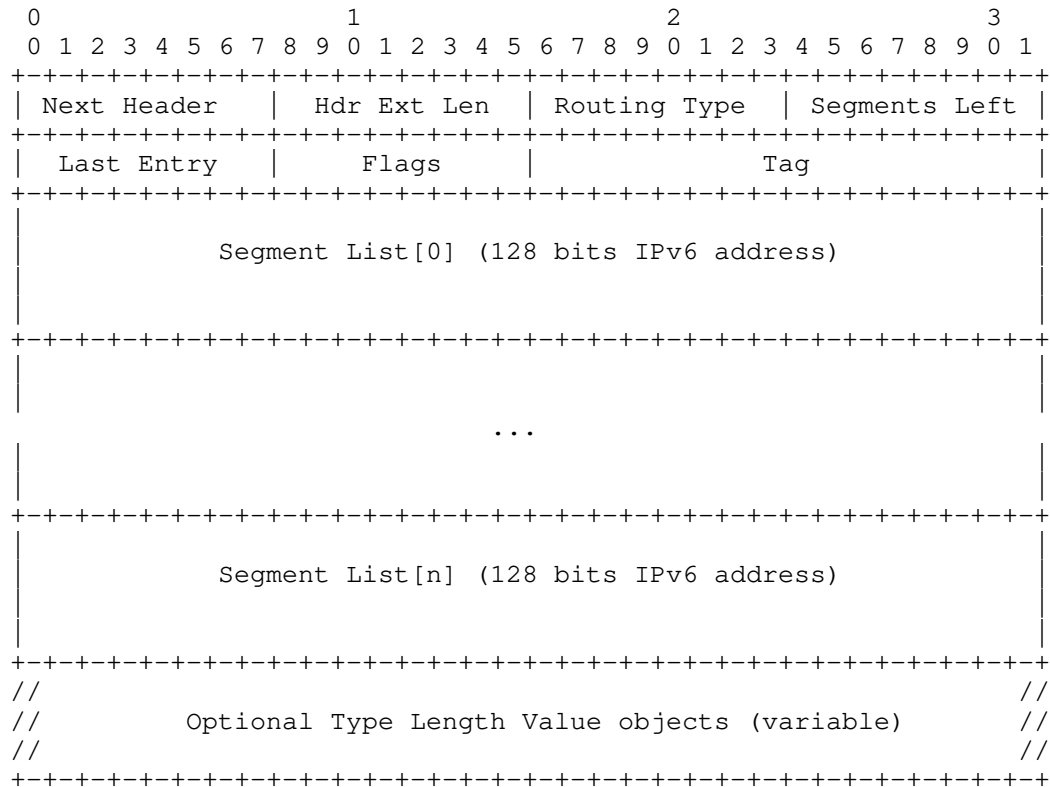
1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Segment Routing Header

Routing Headers are defined in [RFC8200]. The Segment Routing Header has a new Routing Type (suggested value 4) to be assigned by IANA.

The Segment Routing Header (SRH) is defined as follows:



where:

- o Next Header: Defined in [RFC8200] Section 4.4
- o Hdr Ext Len: Defined in [RFC8200] Section 4.4
- o Routing Type: TBD, to be assigned by IANA (suggested value: 4).
- o Segments Left: Defined in [RFC8200] Section 4.4
- o Last Entry: contains the index (zero based), in the Segment List, of the last element of the Segment List.

- o Flags: 8 bits of flags. Section 8.1 creates an IANA registry for new flags to be defined. The following flags are defined:

```

  0 1 2 3 4 5 6 7
+---+---+---+---+
|U U U U U U U U|
+---+---+---+---+

```

U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

- o Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties. When tag is not used at source it MUST be set to zero on transmission. When tag is not used during SRH Processing it SHOULD be ignored. Tag is not used when processing the SID defined in Section 4.3.1. It may be used when processing other SIDs that are not defined in this document. The allocation and use of tag is outside the scope of this document.
- o Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.
- o Type Length Value (TLV) are described in Section 2.1.

In the SRH, the Next Header, Hdr Ext Len, Routing Type, and Segments Left fields are defined in Section 4.4 of [RFC8200]. Based on the constraints in that section, Next Header, Header Ext Len, and Routing Type are not mutable while Segments Left is mutable.

The mutability of the TLV value is defined by the most significant bit in the type, as specified in Section 2.1.

Section 4.3 defines the mutability of the remaining fields in the SRH (Flags, Tag, Segment List) in the context of the SID defined in this document.

New SIDs defined in the future MUST specify the mutability properties of the Flags, Tag, and Segment List and indicate how the HMAC TLV (Section 2.1.2) verification works. Note, that in effect these fields are mutable.

Consistent with the source routing model, the source of the SRH always knows how to set the segment list, Flags, Tag and TLVs of the SRH for use within the SR Domain. How it achieves this is outside the scope of this document, but may be based on topology, available SIDs and their mutability properties, the SRH mutability requirements of the destination, or any other information.

2.1. SRH TLVs

This section defines TLVs of the Segment Routing Header.

A TLV provides meta-data for segment processing. The only TLVs defined in this document are the HMAC (Section 2.1.2) and PAD (Section 2.1.1) TLVs. While processing the SID defined in Section 4.3.1, all TLVs are ignored unless local configuration indicates otherwise (Section 4.3.1.1.1). Thus, TLV and HMAC support is optional for any implementation, however, an implementation adding or parsing TLVs MUST support PAD TLVs. Other documents may define additional TLVs and processing rules for them.

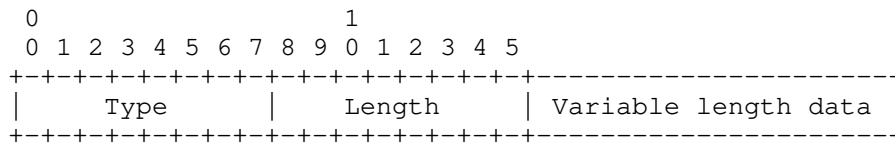
TLVs are present when the Hdr Ext Len is greater than (Last Entry+1)*2.

While processing TLVs at a segment endpoint, TLVs MUST be fully contained within the SRH as determined by the Hdr Ext Len. Detection of TLVs exceeding the boundary of the SRH Hdr Ext Len results in an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field of the SRH, and the packet being discarded.

An implementation MAY limit the number and/or length of TLVs it processes based on local configuration. It MAY:

- o Limit the number of consecutive Pad1 (Section 2.1.1.1) options to 1. If padding of more than one byte is required, then PadN (Section 2.1.1.2) should be used.
- o Limit the length in PadN to 5.
- o Limit the maximum number of non-Pad TLVs to be processed.
- o Limit the maximum length of all TLVs to be processed.

The implementation MAY stop processing additional TLVs in the SRH when these configured limits are exceeded.



Type: An 8 bit codepoint from Segment Routing Header TLVs Registry TBD IANA Reference. Unrecognized Types MUST be ignored on receipt.

Length: The length of the Variable length data in bytes.

Variable length data: Length bytes of data that is specific to the Type.

Type Length Value (TLV) entries contain OPTIONAL information that may be used by the node identified in the Destination Address (DA) of the packet.

Each TLV has its own length, format and semantic. The codepoint allocated (by IANA) to each TLV Type defines both the format and the semantic of the information carried in the TLV. Multiple TLVs may be encoded in the same SRH.

The highest-order bit of the TLV type (bit 0) specifies whether or not the TLV data of that type can change en route to the packet's final destination:

0: TLV data does not change en route

1: TLV data does change en route

All TLVs specify their alignment requirements using an $xn+y$ format. The $xn+y$ format is defined as per [RFC8200]. The SR Source nodes use the $xn+y$ alignment requirements of TLVs and Padding TLVs when constructing an SRH.

The "Length" field of the TLV is used to skip the TLV while inspecting the SRH in case the node doesn't support or recognize the Type. The "Length" defines the TLV length in octets, not including the "Type" and "Length" fields.

The following TLVs are defined in this document:

Padding TLVs

HMAC TLV

Additional TLVs may be defined in the future.

2.1.1.1. Padding TLVs

There are two types of Padding TLVs, pad1 and padN, the following applies to both:

Padding TLVs are used for meeting the alignment requirement of the subsequent TLVs.

Padding TLVs are used to pad the SRH to a multiple of 8 octets.

Padding TLVs are ignored by a node processing the SRH TLV.

Multiple Padding TLVs MAY be used in one SRH

2.1.1.1.1. PAD1

Alignment requirement: none

```

0 1 2 3 4 5 6 7
+---+---+---+---+
|           Type           |
+---+---+---+---+

```

Type: to be assigned by IANA (Suggested value 0)

A single Pad1 TLV MUST be used when a single byte of padding is required. A Pad1 TLV MUST NOT be used if more than one consecutive byte of padding is required.

2.1.1.1.2. PADN

Alignment requirement: none

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Type           |           Length           |           Padding (variable)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//                               Padding (variable)                               //
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type: to be assigned by IANA (suggested value 4).

Length: 0 to 5

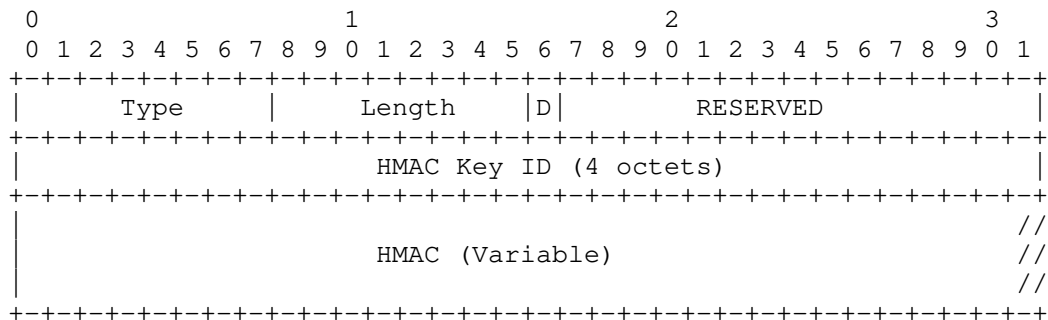
Padding: Length octets of padding. Padding bits have no semantic. They MUST be set to 0 on transmission and ignored on receipt.

The PadN TLV MUST be used when more than one byte of padding is required.

2.1.1.2. HMAC TLV

Alignment requirement: 8n

The keyed Hashed Message Authentication Code (HMAC) TLV is OPTIONAL and has the following format:



where:

- o Type: to be assigned by IANA (suggested value 5).
- o Length: The length of the variable length data in bytes.
- o D: 1 bit. 1 indicates the Destination Address verification is disabled due to use of reduced segment list, Section 4.1.1.
- o RESERVED: 15 bits. MUST be 0 on transmission.
- o HMAC Key ID: A 4 octet opaque number which uniquely identifies the pre-shared key and algorithm used to generate the HMAC.
- o HMAC: Keyed HMAC, in multiples of 8 octets, at most 32 octets.

The HMAC TLV is used to verify that the SRH applied to a packet was selected by an authorized party, and to ensure that the segment list is not modified after generation. This also allows for verification that the current segment (by virtue of being in the authorized segment list) is authorized for use. The SR Domain ensures the source node is permitted to use the source address in the packet via ingress filtering mechanisms as defined in BCP 84 [RFC3704], or other strategies as appropriate.

2.1.2.1. HMAC Generation and Verification

Local configuration determines when to check for an HMAC. This local configuration is outside the scope of this document. It may be based on the active segment at an SR Segment endpoint node, the result of an ACL that considers incoming interface, HMAC Key ID, or other packet fields.

An implementation that supports the generation and verification of the HMAC supports the following default behavior, as defined in the remainder of this section.

The HMAC verification begins by checking the current segment is equal to the destination address of the IPv6 header. The check is successful when either

- o HMAC D bit is 1 and Segments Left is greater than Last Entry.
- o HMAC Segments Left is less than or equal to Last Entry and destination address is equal to Segment List [Segments Left].

The HMAC field is the output of the HMAC computation as defined in [RFC2104], using:

- o key: the pre-shared key identified by HMAC Key ID
- o HMAC algorithm: identified by the HMAC Key ID
- o Text: a concatenation of the following fields from the IPv6 header and the SRH, as it would be received at the node verifying the HMAC:
 - * IPv6 header: source address (16 octets)
 - * SRH: Last Entry (1 octet)
 - * SRH: Flags (1 octet)
 - * SRH: HMAC 16 bits following Length
 - * SRH: HMAC Key ID (4 octets)
 - * SRH: all addresses in the Segment List (variable octets)

The HMAC digest is truncated to 32 octets and placed in the HMAC field of the HMAC TLV.

For HMAC algorithms producing digests less than 32 octets, the digest is placed in the lowest order octets of the HMAC field. Subsequent octets MUST be set to zero such that the HMAC length is a multiple of 8 octets.

If HMAC verification is successful, processing proceeds as normal.

If HMAC verification fails, an ICMP error message (parameter problem, error code 0, pointing to the HMAC TLV) SHOULD be generated (but rate limited) and SHOULD be logged and the packet discarded.

2.1.2.2. HMAC Pre-Shared Key Algorithm

The HMAC Key ID field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys.

The HMAC Key ID field is opaque, i.e., it has neither syntax nor semantic except as an identifier of the right combination of pre-shared key and hash algorithm.

At the HMAC TLV generating and verification nodes, the Key ID uniquely identifies the pre-shared key and HMAC algorithm.

At the HMAC TLV generating node, the Text for the HMAC computation is set to the IPv6 header fields and SRH fields as they would appear at the verification node(s), not necessarily the same as the source node sending a packet with the HMAC TLV.

Pre-shared key roll-over is supported by having two key IDs in use while the HMAC TLV generating node and verifying node converge to a new key.

The HMAC TLV generating node may need to revoke an SRH for which it previously generated an HMAC. Revocation is achieved by allocating a new key and key ID, then rolling over the key ID associated with the SRH to be revoked. The HMAC TLV verifying node drops packets with the revoked SRH.

An implementation supporting HMAC can support multiple hash functions. An implementation supporting HMAC MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

The selection of pre-shared key and algorithm, and their distribution is outside the scope of this document. Some options may include:

- o in the configuration of the HMAC generating or verifying nodes, either by static configuration or any SDN oriented approach

- o dynamically using a trusted key distribution protocol such as [RFC6407]

While key management is outside the scope of this document, the recommendations of BCP 107 [RFC4107] should be considered when choosing the key management system.

3. SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined to a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

3.1. Source SR Node

A Source SR Node is any node that originates an IPv6 packet with a segment (i.e. SRv6 SID) in the destination address of the IPv6 header. The packet leaving the source SR Node may or may not contain an SRH. This includes either:

A host originating an IPv6 packet.

An SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH.

The mechanism through which a segment in the destination address of the IPv6 header and the Segment List in the SRH, is derived is outside the scope of this document.

3.2. Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment nor a local interface. A transit node is not required to be capable of processing a segment nor SRH.

3.3. SR Segment Endpoint Node

A SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment or local interface.

4. Packet Processing

This section describes SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

4.1. Source SR Node

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to the SRH flag or to add TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

Next Header and Hdr Ext Len fields are set as specified in [RFC8200].

Routing Type field is set as TBD (to be allocated by IANA, suggested value 4).

The DA of the packet is set with the value of the first segment.

The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment, and so on.

The Segments Left field is set to $n-1$ where n is the number of elements in the SR Policy.

The Last Entry field is set to $n-1$ where n is the number of elements in the SR Policy.

TLVs (including HMAC) may be set according to their specification.

The packet is forwarded toward the packet's Destination Address (the first segment).

4.1.1. Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to $n-2$ where n is the number of elements in the SR Policy.

4.2. Transit Node

As specified in [RFC8200], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it's routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by means outside the scope of this document. For example, [RFC5308] or [RFC5340] may be used to advertise a prefix covering the SIDs on a node.

4.3. SR Segment Endpoint Node

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

- * A FIB entry that represents a locally instantiated SRv6 SID
- * A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- * A FIB entry that represents a non-local route
- * No Match

4.3.1. FIB Entry Is Locally Instantiated SRv6 SID

This document, and section, defines a single SRv6 SID. Future documents may define additional SRv6 SIDs. In which case, the entire content of this section will be defined in that document.

If the FIB entry represents a locally instantiated SRv6 SID, process the next header chain of the IPv6 header as defined in section 4 of [RFC8200]. Section 4.3.1.1 describes how to process an SRH, Section 4.3.1.2 describes how to process an upper layer header or no next header.

Processing this SID modifies the Segments Left and, if configured to process TLVs, it may modify the "variable length data" of TLV types that change en route. Therefore Segments Left is mutable and TLVs that change en route are mutable. The remainder of the SRH (Flags,

Tag, Segment List, and TLVs that do not change en route) are immutable while processing this SID.

4.3.1.1. SRH Processing

```
S01. When an SRH is processed {
S02.   If Segments Left is equal to zero {
S03.     Proceed to process the next header in the packet,
        whose type is identified by the Next Header field in
        the Routing header.
S04.   }
S05.   Else {
S06.     If local configuration requires TLV processing {
S07.       Perform TLV processing (see TLV Processing)
S08.     }
S09.     max_last_entry = ( Hdr Ext Len / 2 ) - 1
S10.     If ((Last Entry > max_last_entry) or
S11.        (Segments Left is greater than (Last Entry+1))) {
S12.       Send an ICMP Parameter Problem, Code 0, message to
        the Source Address, pointing to the Segments Left
        field, and discard the packet.
S13.     }
S14.     Else {
S15.       Decrement Segments Left by 1.
S16.       Copy Segment List[Segments Left] from the SRH to the
        destination address of the IPv6 header.
S17.       If the IPv6 Hop Limit is less than or equal to 1 {
S18.         Send an ICMP Time Exceeded -- Hop Limit Exceeded in
        Transit message to the Source Address and discard
        the packet.
S19.       }
S20.       Else {
S21.         Decrement the Hop Limit by 1
S22.         Resubmit the packet to the IPv6 module for transmission
        to the new destination.
S23.       }
S24.     }
S25.   }
S26. }
```

4.3.1.1.1. TLV Processing

Local configuration determines how TLVs are to be processed when the Active Segment is a local SID defined in this document. The definition of local configuration is outside the scope of this document.

For illustration purpose only, two example local configurations that may be associated with a SID are provided below.

Example 1:

For any packet received from interface I2
 Skip TLV processing

Example 2:

For any packet received from interface I1
 If first TLV is HMAC {
 Process the HMAC TLV
 }
 Else {
 Discard the packet
 }

4.3.1.2. Upper-layer Header or No Next Header

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 SID defined in this document.

```
IF (Upper-layer Header is IPv4 or IPv6) and
    local configuration permits {
    Perform IPv6 decapsulation
    Resubmit the decapsulated packet to the IPv4 or IPv6 module
}
ELSE {
    Send an ICMP parameter problem message to the Source Address and
    discard the packet.  Error code (TBD by IANA) "SR Upper-layer
    Header Error", pointer set to the offset of the upper-layer
    header.
}
```

A unique error code allows an SR Source node to recognize an error in SID processing at an endpoint.

4.3.2. FIB Entry Is A Local Interface

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

 If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header.

 If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

4.3.3. FIB Entry Is A Non-Local Route

Processing is not changed by this document.

4.3.4. FIB Entry Is A No Match

Processing is not changed by this document.

5. Intra SR Domain Deployment Model

The use of the SIDs exclusively within the SR Domain and solely for packets of the SR Domain is an important deployment model.

This enables the SR Domain to act as a single routing system.

This section covers:

- o securing the SR Domain from external attempt to use its SIDs
- o SR Domain as a single system with delegation between components
- o handling packets of the SR Domain

5.1. Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SIDs of the domain. This is enforced by two levels of access control lists:

1. Any packet entering the SR Domain and destined to a SID within the SR Domain is dropped. This may be realized with the following logic. Other methods with equivalent outcome are considered compliant:
 - * allocate all the SID's from a block S/s
 - * configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s
 - * Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described and referenced in [RFC5095]
2. The distributed protection in #1 is complemented with per node protection, dropping packets to SIDs from source addresses outside the SR Domain. This may be realized with the following

logic. Other methods with equivalent outcome are considered compliant:

- * assign all interface addresses from prefix A/a
- * at node k, all SIDs local to k are assigned from prefix Sk/sk
- * configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a.

5.2. SR Domain as A Single System with Delegation Among Components

All intra SR Domain packets are of the SR Domain. The IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

All inter domain packets are encapsulated for the part of the packet journey that is within the SR Domain. The outer IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

As a consequence, any packet within the SR Domain is of the SR Domain.

The SR Domain is a system in which the operator may want to distribute or delegate different operations of the outer most header to different nodes within the system.

An operator of an SR domain may choose to delegate SRH addition to a host node within the SR domain, and validation of the contents of any SRH to a more trusted router or switch attached to the host. Consider a top of rack switch (T) connected to host (H) via interface (I). H receives an SRH (SRH1) with a computed HMAC via some SDN method outside the scope of this document. H classifies traffic it sources and adds SRH1 to traffic requiring a specific SLA. T is configured with an IACL on I requiring verification of the SRH for any packet destined to the SID block of the SR Domain (S/s). T checks and verifies that SRH1 is valid, contains an HMAC TLV and verifies the HMAC.

An operator of the SR Domain may choose to have all segments in the SR Domain verify the HMAC. This mechanism would verify that the SRH segment list is not modified while traversing the SR Domain.

5.3. MTU Considerations

An SR Domain ingress edge node encapsulates packets traversing the SR Domain, and needs to consider the MTU of the SR Domain. Within the SR Domain, well known mitigation techniques are RECOMMENDED, such as deploying a greater MTU value within the SR Domain than at the ingress edges.

Encapsulation with an outer IPv6 header and SRH share the same MTU and fragmentation considerations as IPv6 tunnels described in [RFC2473]. Further investigation on the limitation of various tunneling methods (including IPv6 tunnels) are discussed in [I-D.ietf-intarea-tunnels] and SHOULD be considered by operators when considering MTU within the SR Domain.

5.4. ICMP Error Processing

ICMP error packets generated within the SR Domain are sent to source nodes within the SR Domain. The invoking packet in the ICMP error message may contain an SRH. Since the destination address of a packet with an SRH changes as each segment is processed, it may not be the destination used by the socket or application that generated the invoking packet.

For the source of an invoking packet to process the ICMP error message, the ultimate destination address of the IPv6 header may be required. The following logic is used to determine the destination address for use by protocol error handlers.

- o Walk all extension headers of the invoking IPv6 packet to the routing extension header preceding the upper layer header.
 - * If routing header is type TBD IANA (SRH)
 - + The SID at Segment List[0] may be used as the destination address of the invoking packet.

ICMP errors are then processed by upper layer transports as defined in [RFC4443].

For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [RFC2473].

5.5. Load Balancing and ECMP

For any inter domain packet, the SR Source node MUST impose a flow label computed based on the inner packet. The computation of the

flow label is as recommended in [RFC6438] for the sending Tunnel End Point.

For any intra domain packet, the SR Source node SHOULD impose a flow label computed as described in [RFC6437] to assist ECMP load balancing at transit nodes incapable of computing a 5-tuple beyond the SRH.

At any transit node within an SR domain, the flow label MUST be used as defined in [RFC6438] to calculate the ECMP hash toward the destination address. If flow label is not used, the transit node would likely hash all packets between a pair of SR Edge nodes to the same link.

At an SR segment endpoint node, the flow label MUST be used as defined in [RFC6438] to calculate any ECMP hash used to forward the processed packet to the next segment.

5.6. Other Deployments

Other deployment models and their implications on security, MTU, HMAC, ICMP error processing and interaction with other extension headers are outside the scope of this document.

6. Illustrations

This section provides illustrations of SRv6 packet processing at SR source, transit and SR segment endpoint nodes.

6.1. Abstract Representation of an SRH

For a node k , its IPv6 address is represented as A_k , its SRv6 SID is represented as S_k .

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A_1 and destination address A_2 is represented as (A_1, A_2) . The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as $\langle S_1, S_2, S_3 \rangle$ where S_1 is the first SID to visit, S_2 is the second SID to visit and S_3 is the last SID to visit.

$(SA, DA) (S_3, S_2, S_1; SL)$ represents an IPv6 packet with:

- o Source Address is SA, Destination Addresses is DA, and next-header is SRH.
- o SRH with SID list $\langle S_1, S_2, S_3 \rangle$ with SegmentsLeft = SL.

- o Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

```
Segments Left=2
Last Entry=2
Flags=0
Tag=0
Segment List[0]=S3
Segment List[1]=S2
Segment List[2]=S1
```

6.2. Example Topology

The following topology is used in examples below:

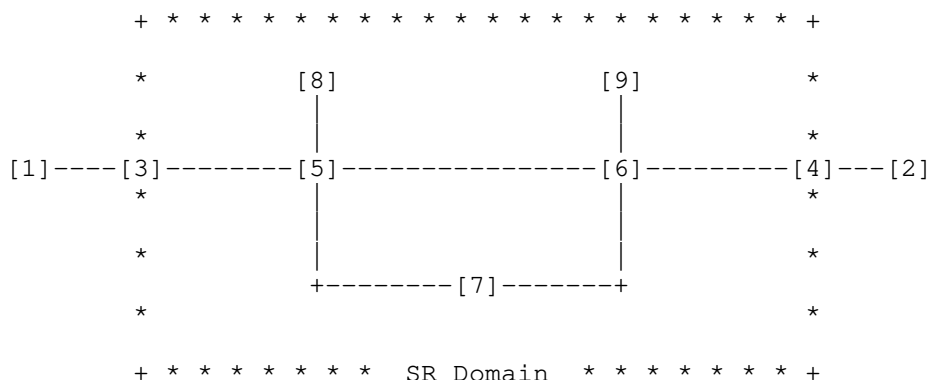


Figure 3

- o 3 and 4 are SR Domain edge routers
- o 5, 6, and 7 are all SR Domain routers
- o 8 and 9 are hosts within the SR Domain
- o 1 and 2 are hosts outside the SR Domain

- o The SR domain implements ingress filtering as per Section 5.1 and no external packet can enter the domain with a destination address equal to a segment of the domain.

6.3. Source SR Node

6.3.1. Intra SR Domain Packet

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> the packet is

P1: (A8,S7) (A9,S7; SL=1)

6.3.1.1. Reduced Variant

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> and it wants to use a reduced SRH, the packet is

P2: (A8,S7) (A9; SL=1)

6.3.2. Inter SR Domain Packet - Transit

When host 1 sends a packet to host 2, the packet is

P3: (A1,A2)

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. Router 3 encapsulates the received packet P3 in an outer header with an SRH. The packet is

P4: (A3, S7) (S4, S7; SL=1) (A1, A2)

If the SR Policy contains only one segment (the egress router 4), the ingress Router 3 encapsulates P3 into an outer header (A3, S4) without SRH. The packet is

P5: (A3, S4) (A1, A2)

6.3.2.1. Reduced Variant

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. If router 3 wants to use a reduced SRH, Router 3 encapsulates the received packet P3 in an outer header with a reduced SRH. The packet is

P6: (A3, S7) (S4; SL=1) (A1, A2)

6.3.3. Inter SR Domain Packet - Internal to External

When host 8 sends a packet to host 1, the packet is encapsulated for the portion of its journey within the SR Domain. From 8 to 3 the packet is

P7: (A8,S3) (A8,A1)

In the opposite direction, the packet generated from 1 to 8 is

P8: (A1,A8)

At node 3 P8 is encapsulated for the portion of its journey within the SR domain, with the outer header destined to segment S8. Resulting in

P9: (A3,S8) (A1,A8)

At node 8 the outer IPv6 header is removed by S8 processing, then processed again when received by A8.

6.4. Transit Node

Nodes 5 acts as transit nodes for packet P1, and sends packet

P1: (A8,S7) (A9,S7;SL=1)

on the interface toward node 7.

6.5. SR Segment Endpoint Node

Node 7 receives packet P1 and, using the logic in Section 4.3.1, sends packet

P7: (A8,A9) (A9,S7; SL=0)

on the interface toward router 6.

6.6. Delegation of Function with HMAC Verification

This section describes how a function may be delegated within the SR Domain. In the following sections consider a host 8 connected to a top of rack 5.

6.6.1. SID List Verification

An operator may prefer to apply the SRH at source 8, while 5 verifies the SID list is valid.

For illustration purpose, an SDN controller provides 8 an SRH terminating at node 9, with segment list <S5,S7,S6,A9>, and HMAC TLV computed for the SRH. The HMAC key ID and key associated with the HMAC TLV is shared with 5. Node 8 does not know the key. Node 5 is configured with an IACL applied to the interface connected to 8, requiring HMAC verification for any packet destined to S/s.

Node 8 originates packets with the received SRH including HMAC TLV.

P15: (A8,S5) (A9,S6,S7,S5;SL=3;HMAC)

Node 5 receives and verifies the HMAC for the SRH, then forwards the packet to the next segment

P16: (A8,S7) (A9,S6,S7,S5;SL=2;HMAC)

Node 6 receives

P17: (A8,S6) (A9,S6,S7,S5;SL=1;HMAC)

Node 9 receives

P18: (A8,A9) (A9,S6,S7,S5;SL=0;HMAC)

This use of an HMAC is particularly valuable within an enterprise based SR Domain [SRN].

7. Security Considerations

This section reviews security considerations related to the SRH, given the SRH processing and deployment models discussed in this document.

As described in Section 5, it is necessary to filter packets ingress to the SR Domain, destined to SIDs within the SR Domain (i.e., bearing a SID in the destination address). This ingress filtering is via an IACL at SR Domain ingress border nodes. Additional protection is applied via an IACL at each SR Segment Endpoint node, filtering packets not from within the SR Domain, destined to SIDs in the SR Domain. ACLs are easily supported for small numbers of prefixes, making summarization important, and when the prefixes requiring filtering is kept to a seldom changing set.

Additionally, ingress filtering of IPv6 source addresses as recommended in BCP38 [RFC2827] SHOULD be used.

7.1. Source Routing Attacks

An SR domain implements distributed and per node protection as described in section 5.1. Additionally, domains deny traffic with spoofed addresses by implementing the recommendations in BCP 84 [RFC3704].

Full implementation of the recommended protection blocks the attacks documented in [RFC5095] from outside the SR domain, including bypassing filtering devices, reaching otherwise unreachable Internet systems, network topology discovery, bandwidth exhaustion, and defeating anycast.

Failure to implement distributed and per node protection allows attackers to bypass filtering devices and exposes the SR Domain to these attacks.

Compromised nodes within the SR Domain may mount the attacks listed above along with other known attacks on IP networks (e.g. DOS/DDOS, topology discovery, man-in-the-middle, traffic interception/siphoning).

7.2. Service Theft

Service theft is defined as the use of a service offered by the SR Domain by a node not authorized to use the service.

Service theft is not a concern within the SR Domain as all SR Source nodes and SR segment endpoint nodes within the domain are able to utilize the services of the Domain. If a node outside the SR Domain learns of segments or a topological service within the SR domain, IACL filtering denies access to those segments.

7.3. Topology Disclosure

The SRH is unencrypted and may contain SIDs of some intermediate SR-nodes in the path towards the destination within the SR Domain. If packets can be snooped within the SR Domain, the SRH may reveal topology, traffic flows, and service usage.

This is applicable within an SR Domain, but the disclosure is less relevant as an attacker has other means of learning topology, flows, and service usage.

7.4. ICMP Generation

The generation of ICMPv6 error messages may be used to attempt denial-of-service attacks by sending an error-causing destination address or SRH in back-to-back packets. An implementation that correctly follows Section 2.4 of [RFC4443] would be protected by the ICMPv6 rate-limiting mechanism.

7.5. Applicability of AH

The SR Domain is a trusted domain, as defined in [RFC8402] Section 2 and Section 8.2. The SR Source is trusted to add an SRH (optionally verified as having been generated by a trusted source via the HMAC TLV in this document), and segments advertised within the domain are trusted to be accurate and advertised by trusted sources via a secure control plane. As such the SR Domain does not rely on the Authentication Header (AH) as defined in [RFC4302] to secure the SRH.

The use of SRH with AH by an SR source node, and processing at a SR segment endpoint node is not defined in this document. Future documents may define use of SRH with AH and its processing.

8. IANA Considerations

This document makes the following registrations in the Internet Protocol Version 6 (IPv6) Parameters "Routing Type" registry maintained by IANA:

Suggested Value	Description	Reference
4	Segment Routing Header (SRH)	This document

This document makes the following registrations in "Type 4 - Parameter Problem" message of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry maintained by IANA:

CODE	NAME/DESCRIPTION
TBD IANA	SR Upper-layer Header Error

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the SRH, in accordance with BCP 26, [RFC8126].

The following terms are used here with the meanings defined in BCP 26: "namespace", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26 [RFC8126]: "IETF Review".

8.1. Segment Routing Header Flags Registry

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header Flags". Flags is 8 bits.

8.2. Segment Routing Header TLVs Registry

This document requests the creation of a new IANA managed registry to identify SRH TLVs. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header TLVs". A TLV is identified through an unsigned 8 bit codepoint value, with assigned values 0-127 for TLVs that do not change en route, and 128-255 for TLVs that may change en route. The following codepoints are defined in this document:

Assigned Value	Description	Reference
0	Pad1 TLV	This document
1	Reserved	This document
2	Reserved	This document
3	Reserved	This document
4	PadN TLV	This document
5	HMAC TLV	This document
6	Reserved	This document
124-126	Experimentation and Test	This document
127	Reserved	This document
252-254	Experimentation and Test	This document
255	Reserved	This document

Values 1,2,3,6 were defined in draft versions of this specification and are Reserved for backwards compatibility with early implementations and should not be reassigned. Values 127 and 255 are Reserved to allow for expansion of the Type field in future specifications if needed.

9. Implementation Status

This section is to be removed prior to publishing as an RFC.

See [I-D.matsushima-spring-srv6-deployment-status] for updated deployment and interoperability reports.

9.1. Linux

Name: Linux Kernel v4.14

Status: Production

Implementation: adds SRH, performs END processing, supports HMAC TLV

Details: <https://irtf.org/anrw/2017/anrw17-final3.pdf> and
[I-D.filsfils-spring-srv6-interop]

9.2. Cisco Systems

Name: IOS XR and IOS XE

Status: Production (IOS XR), Pre-production (IOS XE)

Implementation: adds SRH, performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.3. FD.io

Name: VPP/Segment Routing for IPv6

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

Details: https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6 and
[I-D.filsfils-spring-srv6-interop]

9.4. Barefoot

Name: Barefoot Networks Tofino NPU

Status: Prototype

Implementation: performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.5. Juniper

Name: Juniper Networks Trio and vTrio NPU's

Status: Prototype & Experimental

Implementation: SRH insertion mode, Process SID where SID is an interface address, no TLV processing

9.6. Huawei

Name: Huawei Systems VRP Platform

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

10. Contributors

Kamran Raza, Zafar Ali, Brian Field, Daniel Bernier, Ida Leung, Jen Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, David Lebrun, Dirk Steinberg, Robert Raszuk, Dave Barach, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin contributed to the content of this document.

11. Acknowledgements

The authors would like to thank Ole Troan, Bob Hinden, Ron Bonica, Fred Baker, Brian Carpenter, Alexandru Petrescu, Punit Kumar Jaiswal, David Lebrun, Benjamin Kaduk, Frank Xialiang, Mirja Kuhlewind, Roman Danyliw, Joe Touch, and Magnus Westerlund for their comments to this document.

12. References

12.1. Normative References

[FIPS180-4]

National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.

[RFC2104]

Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

[RFC2119]

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

- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <<https://www.rfc-editor.org/info/rfc3704>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<https://www.rfc-editor.org/info/rfc4107>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <<https://www.rfc-editor.org/info/rfc5095>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

12.2. Informative References

- [I-D.filsfils-spring-srv6-interop]
Filsfils, C., Clad, F., Camarillo, P., Abdelsalam, A., Salsano, S., Bonaventure, O., Horn, J., and J. Liste, "SRv6 interoperability report", draft-filsfils-spring-srv6-interop-02 (work in progress), March 2019.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.
- [I-D.matsushima-spring-srv6-deployment-status]
Matsushima, S., Filsfils, C., Ali, Z., and Z. Li, "SRv6 Implementation and Deployment Status", draft-matsushima-spring-srv6-deployment-status-02 (work in progress), October 2019.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <<https://www.rfc-editor.org/info/rfc5308>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [SRN] Lebrun, D., Jadin, M., Clad, F., Filsfils, C., and O. Bonaventure, "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing", 2018, <<https://inl.info.ucl.ac.be/system/files/sosr18-final15-embedfonts.pdf>>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Darren Dukes (editor)
Cisco Systems, Inc.
Ottawa
CA

Email: ddukes@cisco.com

Stefano Previdi
Huawei
Italy

Email: stefano@previdi.net

John Leddy
Individual
US

Email: john@leddy.net

Satoru Matsushima
Softbank

Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer
Bell Canada

Email: daniel.voyer@bell.ca

IETF
Internet-Draft
Intended status: Informational
Expires: October 13, 2016

P. McCann, Ed.
J. Kaippallimalil, Ed.
Huawei
April 11, 2016

Communicating Prefix Cost to Mobile Nodes
draft-mccann-dmm-prefixcost-03

Abstract

In a network implementing Distributed Mobility Management, it has been agreed that Mobile Nodes (MNs) should exhibit agility in their use of IP addresses. For example, an MN might use an old address for ongoing socket connections but use a new, locally assigned address for new socket connections. Determining when to assign a new address, and when to release old addresses, is currently an open problem. Making an optimal decision about address assignment and release must involve a tradeoff in the amount of signaling used to allocate the new addresses, the amount of utility that applications are deriving from the use of a previously assigned address, and the cost of maintaining an address that was assigned at a previous point of attachment. As the MN moves farther and farther from the initial point where an address was assigned, more and more resources are used to redirect packets destined for that IP address to its current location. The MN currently does not know the amount of resources used as this depends on mobility path and internal routing topology of the network(s) which are known only to the network operator. This document provides a mechanism to communicate to the MN the cost of maintaining a given prefix at the MN's current point of attachment so that the MN can make better decisions about when to release old addresses and assign new ones.

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 October 13, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	4
1.2. Abbreviations	4
2. Motivation	4
3. Prefix Cost Sub-option	5
4. Host Considerations	6
5. Security Considerations	7
6. IANA Considerations	8
7. References	8
7.1. Normative References	8
7.2. Informative References	8
Authors' Addresses	9

1. Introduction

Previous discussions on address agility in distributed mobility management have focused on "coloring" prefixes with one of a small number of categories, such as Fixed, Sustained, or Nomadic. The assumption here is that the MN should use a permanent home address for sessions that need a persistent IP address, and a local, ephemeral address for short-lived sessions such as browsing. However, a small set of address categories lacks expressive power and leads to false promises being made to mobile nodes. For example, the concept that a home address can be maintained permanently and offered as an on-link prefix by any access router to which the MN may be attached in future is simply not attainable in the real world. There will always exist some access routers that do not have arrangements in place with the home network to re-route (via tunneling or other mechanisms) the home prefix to the current point of attachment.

Conversely, the assumption that a Nomadic prefix will never be available to an MN after it changes its current point of attachment is too limiting. There is no reason why an MN should not be able to keep a prefix that was assigned by a first network after it moves to a second network, provided that measures are put in place to re-route such prefixes to the new attachment point.

Rather, this document argues that there is in reality a continuum of cost associated with an address as the MN moves from one attachment point to another or from one network to another. The sources of the cost are the increased latency, network bandwidth, and network state being maintained by a network-based mobility management scheme to route packets destined to the prefix to the MN's current point of attachment. By communicating this cost to the MN every time its attachment point changes, the MN can make intelligent decisions about when to release old addresses and when to acquire new ones.

The cost should be communicated to the MN because of several constraints inherent in the problem:

- (1) The MN is the entity that must make decisions about allocating new addresses and releasing old ones. This is because only the MN has the information about which addresses are still in use by applications or have been registered with other entities such as DNS servers.
- (2) Only the network has information about the cost of maintaining the prefix in a network-based mobility management scheme, because the MN cannot know the network topology that gives rise to the inefficiencies.

If the cost of maintaining a prefix is not made available to the mobile node, it may attempt to infer the cost through heuristic mechanisms. For example, it can measure increased end-to-end latency after a mobility event, and attribute the increased latency to a longer end-to-end path. However, this method does not inform the MN about the network bandwidth being expended or network state being maintained on its behalf. Alternatively, a MN may attempt to count mobility events or run a timer in an attempt to guess at which older prefixes are more costly and in need of being released. However, these methods fail because the number of mobility events is not an indication of how far the MN has moved in a topological sense from its original attachment point which is what gives rise to the costs outlined above. Re-allocating an address upon expiration of a timer may introduce unnecessary and burdensome signaling load on the network and air interface.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

1.2. Abbreviations

ANDSF	Access Network Discovery and Selection Function
MN	Mobile Node
MPTCP	Multi-Path Transmission Control Protocol
ND	Neighbor Discovery
NGMN	Next Generation Mobile Networks
NUD	Neighbor Unreachability Detection
OMA-DM	Open Mobile Alliance - Device Management
PIO	Prefix Information Discovery
PGW	Packet data network Gateway
SeND	Secure Neighbor Discovery
SGW	Serving Gateway

2. Motivation

The Introduction speaks in general terms about the cost of a prefix. More specifically, we are talking about the aggregate amount of state being maintained in the network on behalf of the mobile node in addition to the transport resources being used (or wasted) to get packets to the MN's current point of attachment.

In a non-mobile network, the addresses can be assigned statically in a manner that is aligned with the topology of the network. This means that prefix aggregation can be used for maximum efficiency in the state being maintained in such a network. Nodes deep in the network need only concern themselves with a small number of short prefixes, and only nodes near the end host need to know longer more specific prefixes. In the best case, only the last-hop router(s) need to know the actual address assigned to the end host. Also, routing protocols ensure that packets follow the least-cost path to the end host in terms of number of routing hops or according to other policies defined by the service provider, and these routing paths can change dynamically as links fail or come back into service.

However, mobile nodes in a wide-area wireless network are often handled very differently. A mobile node is usually assigned a fixed gateway somewhere in the network, either in a fixed central location or (better) in a location near where the MN first attaches to the network. For example, in a 3GPP network this gateway is a PGW that can be allocated in the home or visited networks. Initially, the cost of such a prefix is the state entry in the fixed gateway plus

any state entries in intermediate tunneling nodes (like SGWs) plus whatever transport resources are being used to get the packet to the MN's initial point of attachment.

When an MN changes its point of attachment, but keeps a fixed address, the cost of the prefix changes (usually it increases). Even if the fixed gateway was initially allocated very close to the initial point of attachment, as the MN moves away from this point, additional state must be inserted into the network and additional transport resources must be provided to get the packets to the current point of attachment. For example, a new SGW might be allocated in a new network, and now the packets must traverse the network to which the MN first attached before being forwarded to their destination, even though there may be a better and more direct route to communication peers from the new network. Whatever aggregation was possible at the initial point of attachment is now lost and tunnels must be constructed or holes must be punched in routing tables to ensure continued connectivity of the fixed IP address at the new point of attachment. Over time, as the MN moves farther and farther from its initial point of attachment, these costs can become large. When summed over millions of mobile nodes, the costs can be quite large.

Obviously, the assignment of a new address at a current point of attachment and release of the older, more costly prefix will help to reduce costs and may be the only way to meet emerging more stringent latency requirements [8]. However, the MN does not in general know the current cost of a prefix because it depends on the network topology and the number of handovers that have taken place and whether these handovers have caused the MN to transition between different topological parts of the network. It is the purpose of the protocol extension defined in this document to communicate the current cost of a prefix to the MN so that it can make intelligent decisions about when to get a new address and when to release older addresses. Only the MN can make a decision about when to release an address, because it is the only entity that knows whether applications are still listening waiting to receive packets at the old address.

Section 4 describes MN behavior when Router Advertisements with Prefix Cost is received.

3. Prefix Cost Sub-option

This document defines a prefix cost option to be carried in router advertisements. It is a sub-option that carries meta-data as defined by Korhonen et al. [7]

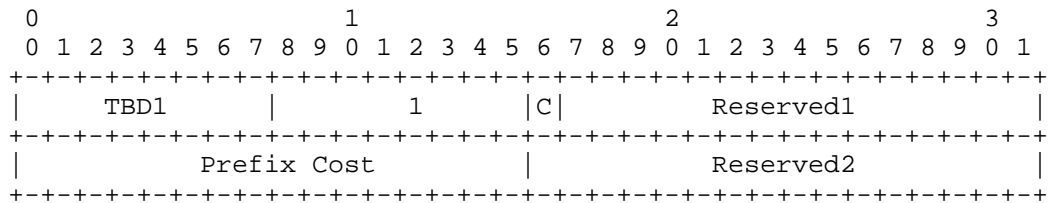


Figure 1: Prefix Cost suboption

The prefix cost is carried as a 16-bit, unsigned number in network byte order. An higher number indicates an increased cost.

This sub-option is appended in Router Advertisement messages that are sent on a periodic basis. No additional signaling cost is incurred to support this mechanism.

It should be noted that link layer events do not cause a change in the prefix cost.

The prefix cost is for a connection segment. No end-to-end congestion or flow control mechanisms are implied with this cost.

4. Host Considerations

Prefix Cost in a Router Advertisement PIO serves as a hint for the MN to use along with application knowledge, MN policy configuration on network cost and available alternative routes to determine the IP addresses and routes used. For example, if the application is downloading a large file, it may want to maintain an IP address and route until the download is complete. On the other hand, some applications may use multiple connections (e.g., with MPTCP) and may not want to maintain an IP address above a configured cost. It could also be the case that the MN maintains the IP address even at high cost if there is no alternative route/address. These decisions are made based on configured policy, and interaction with applications, all of which are decided by the MN.

When the MN is ready to release an IP address, it may send a DHCPv6 [5] Release message. The network may also monitor the status of a high cost connection with Neighbor Unreachability Detection (NUD) [2], [6], and determine that an address is not used after the NUD times out. The network should not continue to advertise this high cost route following the explicit release of the address or NUD timeout. It can initiate the release of network resources dedicated to providing the IP address to the MN.

The operator of the network or host's service provider can configure policy that determines how the host should handle the prefix cost values. In a 3GPP network, the subscription provider may configure policies in the host via OMA-DM or S14 (ANDSF). For example, the service provider may configure rules to state that prefix cost values below 500 indicate low cost and ideal access network conditions, values from 501 - 5000 indicate that the host should try to relocate connections, and values above 5000 indicate a risk and impending loss of connectivity. The policies themselves can be (re-)configured as needed by the operator. Prefix cost information with each Router Advertisement allows the host to interpret a simple number and associated policies to (re-)select optimal routes. For networks service providers, when this cost is associated with charging, it can be a valuable tool in dynamically managing the utilization of network resources.

This draft does not aim to provide definitive guidance on how an OS or application process receives indications as a result of prefix cost option being conveyed in Router Advertisements. Only high level design options are listed here. New socket options or other APIs can be used to communicate the cost of an address in use on a given connection. For example, a new "prefix-cost" socket option, if set, can indicate that the application is interested in being notified when there is a change in the prefix cost. The actual mechanisms used to either notify or other means of busy polling on this change of prefix cost information need to be specified in other drafts. An alternative to the application discovering the changed prefix cost is to use a model where a connection manager handles the interface between the network and the application (e.g., Android Telephony Manager [9]). In this case, the connection manager is responsible to select and manage addresses based on policies (configured via OMA-DM or S14) and prefix cost obtained from the Router Advertisements.

5. Security Considerations

Security of the prefix cost option in the PIO needs to be considered. Neighbor Discovery (ND) and Prefix Information Option (PIO) security are described in [2] and [3]. A malicious node on a shared link can advertise a low cost route in the prefix cost option and cause the MN to switch. Alternatively, an incorrect higher cost route in the prefix cost option can result in the suboptimal use of network resources. In order to avoid such on-link attacks, SeND [4] can be used to reject Router Advertisements from nodes whose identities are not validated.

6. IANA Considerations

This memo defines a new Prefix Information Option (PIO) sub-option in Section 3.

7. References

7.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [2] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [3] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [4] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<http://www.rfc-editor.org/info/rfc3971>>.
- [5] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.
- [6] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, DOI 10.17487/RFC7048, January 2014, <<http://www.rfc-editor.org/info/rfc7048>>.

7.2. Informative References

- [7] Korhonen, J., Gundavelli, S., Seite, P., and D. Liu, "IPv6 Prefix Properties", draft-korhonen-dmm-prefix-properties-05 (work in progress), February 2016.
- [8] NGMN Alliance, "NGMN 5G Whitepaper", February 2015.

- [9] Android Telephony Developer's Forum,
 <http://developer.android.com/reference/android/telephony/TelephonyManager.html>, "Android Telephony Manager".

Authors' Addresses

Peter J. McCann (editor)
Huawei
400 Crossing Blvd, 2nd Floor
Bridgewater, NJ 08807
USA

Phone: +1 908 541 3563
Email: peter.mccann@huawei.com

John Kaippallimalil (editor)
Huawei
5340 Legacy Dr., Suite 175
Plano, TX 75024
USA

Email: john.kaippallimalil@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 4, 2016

S. Previdi, Ed.
C. Filsfils
Cisco Systems, Inc.
B. Field
Comcast
I. Leung
Rogers Communications
J. Linkova
Google
E. Aries
Facebook
T. Kosugi
NTT
E. Vyncke
Cisco Systems, Inc.
D. Lebrun
Universite Catholique de Louvain
October 2, 2015

IPv6 Segment Routing Header (SRH)
draft-previdi-6man-segment-routing-header-08

Abstract

Segment Routing (SR) allows a node to steer a packet through a controlled set of instructions, called segments, by prepending a SR header to the packet. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any path (topological, or application/service based) while maintaining per-flow state only at the ingress node to the SR domain.

Segment Routing can be applied to the IPv6 data plane with the addition of a new type of Routing Extension Header. This draft describes the Segment Routing Extension Header Type and how it is used by SR capable nodes.

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 April 4, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Segment Routing Documents	3
2. Introduction	3
2.1. Data Planes supporting Segment Routing	4
2.2. Segment Routing (SR) Domain	4
2.2.1. SR Domain in a Service Provider Network	5
2.2.2. SR Domain in a Overlay Network	6
2.3. Illustration	8
3. IPv6 Instantiation of Segment Routing	10
3.1. Segment Identifiers (SIDs)	10
3.1.1. Node-SID	10
3.1.2. Adjacency-SID	11
3.2. Segment Routing Extension Header (SRH)	11
3.2.1. SRH and RFC2460 behavior	14
4. SRH Procedures	15
4.1. Segment Routing Node Functions	15
4.1.1. Source SR Node	16
4.1.2. SR Domain Ingress Node	17
4.1.3. Transit Node	17
4.1.4. SR Segment Endpoint Node	17

5. Security Considerations	18
5.1. Threat model	19
5.1.1. Source routing threats	19
5.1.2. Applicability of RFC 5095 to SRH	19
5.1.3. Service stealing threat	20
5.1.4. Topology disclosure	20
5.1.5. ICMP Generation	20
5.2. Security fields in SRH	21
5.2.1. Selecting a hash algorithm	22
5.2.2. Performance impact of HMAC	22
5.2.3. Pre-shared key management	23
5.3. Deployment Models	23
5.3.1. Nodes within the SR domain	23
5.3.2. Nodes outside of the SR domain	24
5.3.3. SR path exposure	24
5.3.4. Impact of BCP-38	25
6. IANA Considerations	25
7. Manageability Considerations	25
8. Contributors	25
9. Acknowledgements	26
10. References	26
10.1. Normative References	26
10.2. Informative References	26
Authors' Addresses	28

1. Segment Routing Documents

Segment Routing terminology is defined in
[I-D.ietf-spring-segment-routing].

Segment Routing use cases are described in
[I-D.ietf-spring-problem-statement] and
[I-D.ietf-spring-ipv6-use-cases].

Segment Routing protocol extensions are defined in
[I-D.ietf-isis-segment-routing-extensions], and
[I-D.ietf-ospf-ospfv3-segment-routing-extensions].

2. Introduction

Segment Routing (SR), defined in [I-D.ietf-spring-segment-routing], allows a node to steer a packet through a controlled set of instructions, called segments, by prepending a SR header to the packet. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any path (topological or service/application based) while maintaining per-flow state only at the ingress node to the SR domain. Segments can be derived from different components: IGP, BGP, Services, Contexts,

Locators, etc. The list of segment forming the path is called the Segment List and is encoded in the packet header.

SR allows the use of strict and loose source based routing paradigms without requiring any additional signaling protocols in the infrastructure hence delivering an excellent scalability property.

The source based routing model described in [I-D.ietf-spring-segment-routing] is inherited from the ones proposed by [RFC1940] and [RFC2460]. The source based routing model offers the support for explicit routing capability.

2.1. Data Planes supporting Segment Routing

Segment Routing (SR), can be instantiated over MPLS ([I-D.ietf-spring-segment-routing-mpls]) and IPv6. This document defines its instantiation over the IPv6 data-plane based on the use-cases defined in [I-D.ietf-spring-ipv6-use-cases].

This document defines a new type of Routing Header (originally defined in [RFC2460]) called the Segment Routing Header (SRH) in order to convey the Segment List in the packet header as defined in [I-D.ietf-spring-segment-routing]. Mechanisms through which segment are known and advertised are outside the scope of this document.

A segment is materialized by an IPv6 address. A segment identifies a topological instruction or a service instruction. A segment can be either:

- o global: a global segment represents an instruction supported by all nodes in the SR domain and it is instantiated through an IPv6 address globally known in the SR domain.
- o local: a local segment represents an instruction supported only by the node who originates it and it is instantiated through an IPv6 address that is known only by the local node.

2.2. Segment Routing (SR) Domain

We define the concept of the Segment Routing Domain (SR Domain) as the set of nodes participating into the source based routing model. These nodes may be connected to the same physical infrastructure (e.g.: a Service Provider's network) as well as nodes remotely connected to each other (e.g.: an enterprise VPN or an overlay).

A non-exhaustive list of examples of SR Domains is:

- o The network of an operator, service provider, content provider, enterprise including nodes, links and Autonomous Systems.
- o A set of nodes connected as an overlay over one or more transit providers. The overlay nodes exchange SR-enabled traffic with segments belonging solely to the overlay routers (the SR domain). None of the segments in the SR-enabled packets exchanged by the overlay belong to the transit networks

The source based routing model through its instantiation of the Segment Routing Header (SRH) defined in this document equally applies to all the above examples.

While the source routing model defined in [RFC2460] doesn't mandate which node is allowed to insert (or modify) the SRH, it is assumed in this document that the SRH is inserted in the packet by its source. For example:

- o At the node originating the packet (host, server).
- o At the ingress node of a SR domain where the ingress node receives an IPv6 packet and encapsulates it into an outer IPv6 header followed by a Segment Routing header.

2.2.1. SR Domain in a Service Provider Network

The following figure illustrates an SR domain consisting of an operator's network infrastructure.

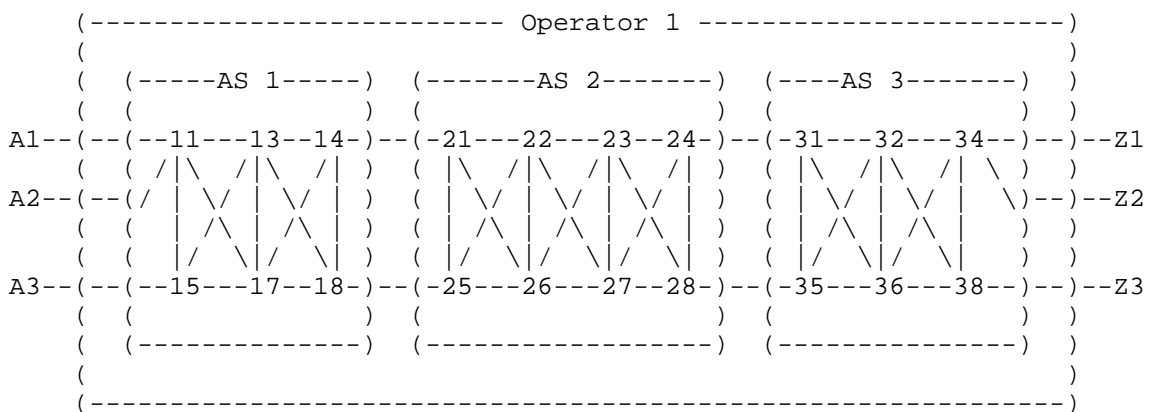


Figure 1: Service Provider SR Domain

Figure 1 describes an operator network including several ASes and delivering connectivity between endpoints. In this scenario, Segment

Routing is used within the operator networks and across the ASes boundaries (all being under the control of the same operator). In this case segment routing can be used in order to address use cases such as end-to-end traffic engineering, fast re-route, egress peer engineering, data-center traffic engineering as described in [I-D.ietf-spring-problem-statement], [I-D.ietf-spring-ipv6-use-cases] and [I-D.ietf-spring-resiliency-use-cases].

Typically, an IPv6 packet received at ingress (i.e.: from outside the SR domain), is classified according to network operator policies and such classification results into an outer header with an SRH applied to the incoming packet. The SRH contains the list of segment representing the path the packet must take inside the SR domain. Thus, the SA of the packet is the ingress node, the DA (due to SRH procedures described in Section 4) is set as the first segment of the path and the last segment of the path is the egress node of the SR domain.

The path may include intra-AS as well as inter-AS segments. It has to be noted that all nodes within the SR domain are under control of the same administration. When the packet reaches the egress point of the SR domain, the outer header and its SRH are removed so that the destination of the packet is unaware of the SR domain the packet has traversed.

The outer header with the SRH is no different from any other tunneling encapsulation mechanism and allows a network operator to implement traffic engineering mechanisms so to efficiently steer traffic across his infrastructure.

2.2.2. SR Domain in a Overlay Network

The following figure illustrates an SR domain consisting of an overlay network over multiple operator's networks.

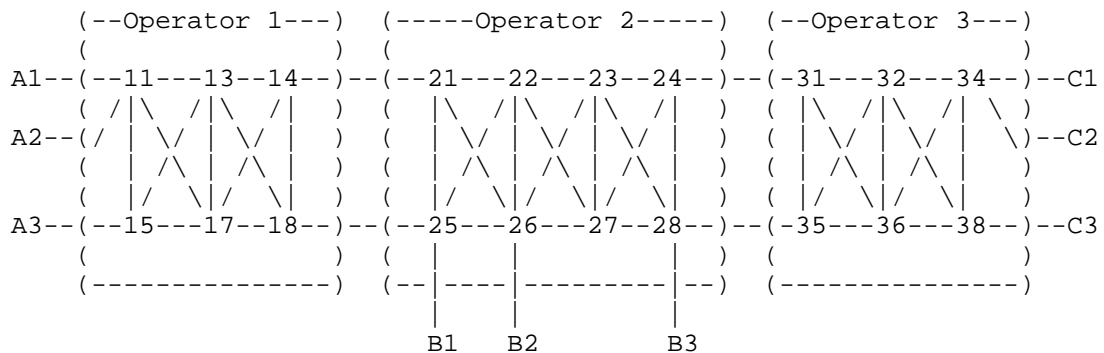


Figure 2: Overlay SR Domain

Figure 2 describes an overlay consisting of nodes connected to three different network operators and forming a single overlay network where Segment routing packets are exchanged.

The overlay consists of nodes A1, A2, A3, B1, B2, B3, C1, C2 and C3. These nodes are connected to their respective network operator and form an overlay network.

Each node may originate packets with an SRH which contains, in the segment list of the SRH or in the DA, segments identifying other overlay nodes. This implies that packets with an SRH may traverse operator's networks but, obviously, these SRHs cannot contain an address/segment of the transit operators 1, 2 and 3. The SRH originated by the overlay can only contain address/segment under the administration of the overlay (e.g. address/segments supported by A1, A2, A3, B1, B2, B3, C1, C2 or C3).

In this model, the operator network nodes are transit nodes and, according to [RFC2460], MUST NOT inspect the routing extension header since there are not the DA of the packet.

It is a common practice in operators networks to filter out, at ingress, any packet whose DA is the address of an internal node and it is also possible that an operator would filter out any packet destined to an internal address and having an extension header in it.

This common practice does not impact the SR-enabled traffic between the overlay nodes as the intermediate transit networks do never see a destination address belonging to their infrastructure. These SR-enabled overlay packets will thus never be filtered by the transit operators.

In all cases, transit packets (i.e.: packets whose DA is outside the domain of the operator's network) will be forwarded accordingly without introducing any security concern in the operator's network. This is similar to tunneled packets.

2.3. Illustration

In the context of Figure 3 we illustrate an example of how segment routing can be used within a SR domain in order to engineer traffic. Let's assume that the SR domain is configured as a single AS and the IGP (OSPF or IS-IS) is configured using the same cost on every link. Let's also assume that a packet P enters the SR domain at an ingress edge router I and that the operator requests the following requirements for packet P:

- o The local service S offered by node B must be applied to packet P.
- o The links AB and CE cannot be used to transport the packet P.
- o Any node N along the journey of the packet should be able to determine where the packet P entered the SR domain and where it will exit. The intermediate node should be able to determine the paths from the ingress edge router to itself, and from itself to the egress edge router.
- o Per-flow State for packet P should only be created at the ingress edge router.
- o The operator can forbid, for security reasons, anyone outside the operator domain to exploit its intra-domain SR capabilities.

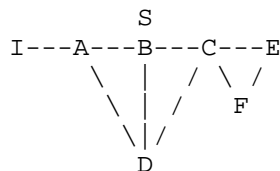


Figure 3: An illustration of SR properties

All these properties may be realized by instructing the ingress SR edge router I to create a SRH with the list of segments the packet must traverse: D, B, S, F, E. Therefore, the ingress router I creates an outer header where:

- o the SA is the IPv6 address of I

- o the final destination of the packet is the SR egress node E however, D being the first segment of the path, the DA is set to D IPv6 address.
- o the SRH is inserted with the segment list consisting of following IPv6 addresses: D, B, S, F, E

The SRH contains a source route encoded as a list of segments (D, B, S, F, E). The ingress and egress nodes are identified in the packet respectively by the SA and the last segment of the segment list.

The packet P reaches the ingress SR node I. Node I pushes the newly created outer header and SRH with the Segment List as illustrated above (D, B, S, F, E)

D is the IPv6 address of node D and it is recognized by all nodes in the SR domain as the forwarding instruction "forward to D according to D route in the IPv6 routing table". The routing table being built through IGPs (OSPF or IS-IS) it is equivalent to say "forward according to shortest path to D".

Once at D, the next segment is inspected and executed (segment B).

B is an instruction recognized by all the nodes in the SR domain which causes the packet to be forwarded along the shortest path to B.

Once at B, the next segment is executed (segment S).

S is an instruction only recognized by node B which causes the packet to receive service S.

Once the service S is applied, the next segment is executed (segment F) which causes the packet to be forwarded along the shortest path to F.

Once at F, the next segment is executed (segment E).

E is an instruction recognized by all the nodes in the SR domain which causes the packet to be forwarded along the shortest path to E.

E being the destination of the packet, removes the outer header and the SRH. Then, it inspects the inner packet header and forwards the packet accordingly.

All of the requirements are met:

- o First, the packet P has not used links AB and CE: the shortest-path from I to D is I-A-D, the shortest-path from D to B is D-B,

the shortest-path from B to F is B-C-F and the shortest-path from F to E is F-E, hence the packet path through the SR domain is I-A-D-B-C-F-E and the links AB and CE have been avoided.

- o Second, the service S supported by B has been applied on packet P.
- o Third, any node along the packet path is able to identify the service and topological journey of the packet within the SR domain by inspecting the SRH and SA/DA fields of the packet header.
- o Fourth, only node I maintains per-flow state for packet P. The entire program of topological and service instructions to be executed by the SR domain on packet P is encoded by the ingress edge router I in the SR header in the form of a list of segments where each segment identifies a specific instruction. No further per-flow state is required along the packet path. Intermediate nodes only hold states related to the global node segments and their local segments. These segments are not per-flow specific and hence scale very well. Typically, an intermediate node would maintain in the order of 100's to 1000's global node segments and in the order of 10's to 100 of local segments.
- o Fifth, the SR header (and its outer header) is inserted at the entrance to the domain and removed at the exit of the operator domain. For security reasons, the operator can forbid anyone outside its domain to use its intra-domain SR capability (e.g. configuring ACL that deny any packet with a DA towards its infrastructure segment).

3. IPv6 Instantiation of Segment Routing

3.1. Segment Identifiers (SIDs)

Segment Routing, as described in [I-D.ietf-spring-segment-routing], defines Node-SID and Adjacency-SID. When SR is used over IPv6 data-plane the following applies.

3.1.1. Node-SID

The Node-SID identifies a node. With SR-IPv6 the Node-SID is an IPv6 address that the operator configured on the node and that is used as the node identifier. Typically, in case of a router, this is the IPv6 address of the node loopback interface. Therefore, SR-IPv6 does not require any additional SID advertisement for the Node Segment. The Node-SID is in fact the IPv6 address of the node.

3.1.2. Adjacency-SID

Adjacency-SIDs can be either globally scoped IPv6 addresses or IPv6 addresses known locally by the node but not advertised in any control plane (in other words an Adjacency-SID may well be any 128-bit identifier). Obviously, in the latter case, the scope of the Adjacency-SID is local to the router and any packet with the a such Adjacency-SID would need first to reach the node through the node's Segment Identifier (i.e.: Node-SID) prior for the node to process the Adjacency-SID. In other words, two segments (SIDs) would then be required: the first is the node's Node-SID that brings the packet to the node and the second is the Adjacency-SID that will make the node to forward the packet through the interface the Adjacency-SID is allocated to.

In the SR architecture defined in [I-D.ietf-spring-segment-routing] a node may advertise one (or more) Adj-SIDs allocated to the same interface as well as a node can advertise the same Adj-SID for multiple interfaces. Use cases of Adj-SID advertisements are described in [I-D.ietf-spring-segment-routing] The semantic of the Adj-SID is:

Send out the packet to the interface this Adj-SID is allocated to.

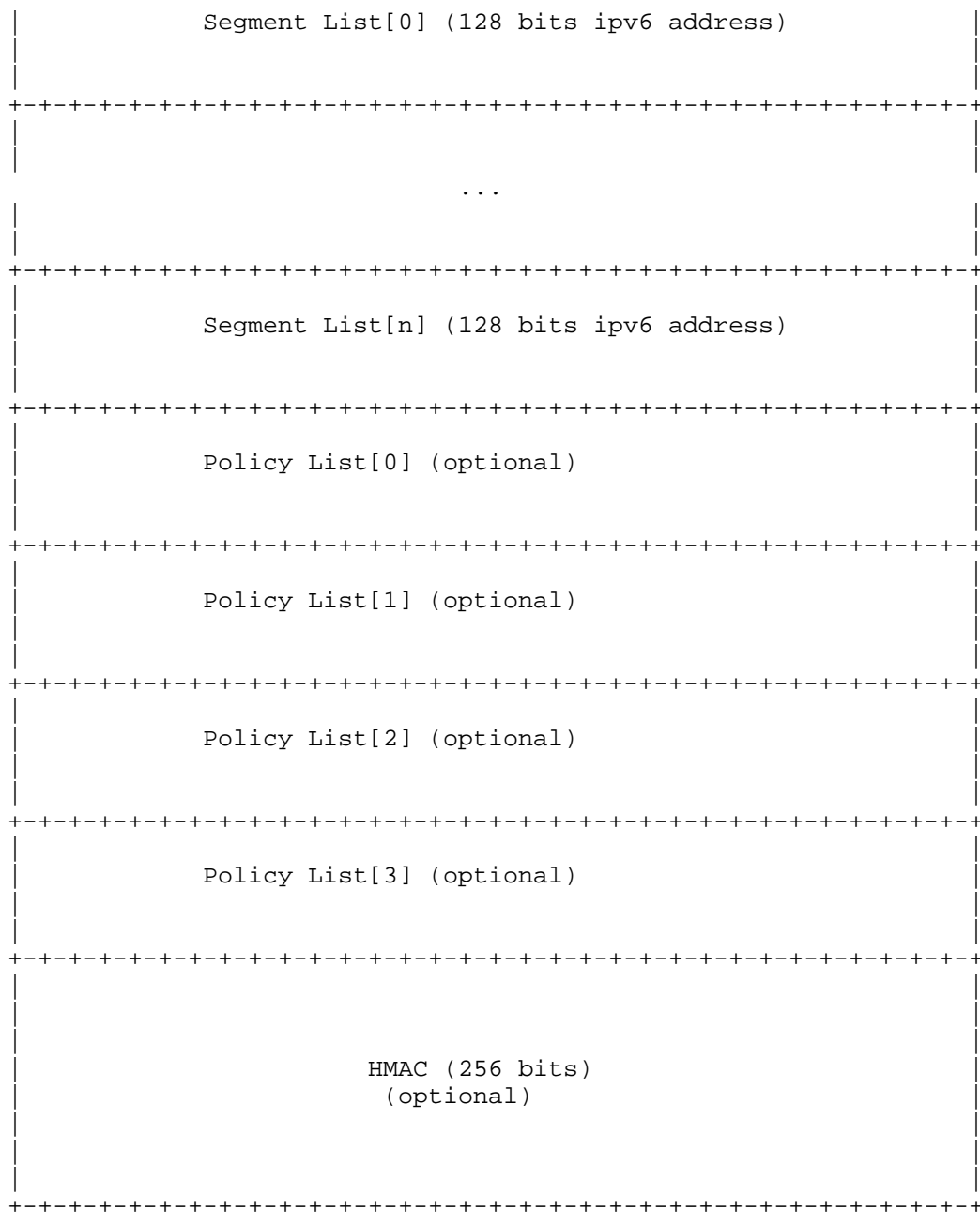
Advertisement of Adj-SID may be done using multiple mechanisms among which the ones described in ISIS and OSPF protocol extensions: [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions]. The distinction between local and global significance of the Adj-SID is given in the encoding of the Adj-SID advertisement.

3.2. Segment Routing Extension Header (SRH)

A new type of the Routing Header (originally defined in [RFC2460]) is defined: the Segment Routing Header (SRH) which has a new Routing Type, (suggested value 4) to be assigned by IANA.

The Segment Routing Header (SRH) is defined as follows:

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1
Next Header	Hdr Ext Len	Routing Type	Segments Left
First Segment	Flags	HMAC Key ID	



where:

- o Next Header: 8-bit selector. Identifies the type of header immediately following the SRH.
- o Hdr Ext Len: 8-bit unsigned integer, is the length of the SRH header in 8-octet units, not including the first 8 octets.
- o Routing Type: TBD, to be assigned by IANA (suggested value: 4).
- o Segments Left. Defined in [RFC2460], it contains the index, in the Segment List, of the next segment to inspect. Segments Left is decremented at each segment.
- o First Segment: contains the index, in the Segment List, of the first segment of the path which is in fact the last element of the Segment List.
- o Flags: 16 bits of flags. Following flags are defined:

```

                                1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|C|P|R|R|   Policy Flags   |
+---+---+---+---+---+---+---+---+

```

C-flag: Clean-up flag. Set when the SRH has to be removed from the packet when packet reaches the last segment.

P-flag: Protected flag. Set when the packet has been rerouted through FRR mechanism by a SR endpoint node.

R-flags. Reserved and for future use.

Policy Flags. Define the type of the IPv6 addresses encoded into the Policy List (see below). The following have been defined:

Bits 4-6: determine the type of the first element after the segment list.

Bits 7-9: determine the type of the second element.

Bits 10-12: determine the type of the third element.

Bits 13-15: determine the type of the fourth element.

The following values are used for the type:

0x0: Not present. If value is set to 0x0, it means the element represented by these bits is not present.

0x1: SR Ingress.

0x2: SR Egress.

0x3: Original Source Address.

0x4 to 0x7: currently unused and SHOULD be ignored on reception.

- o HMAC Key ID and HMAC field, and their use are defined in Section 5.
- o Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the path. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the path while the last segment of the Segment List (Segment List[n]) contains the first segment of the path. The index contained in "Segments Left" identifies the current active segment.
- o Policy List. Optional addresses representing specific nodes in the SR path such as:

SR Ingress: a 128 bit generic identifier representing the ingress in the SR domain (i.e.: it needs not to be a valid IPv6 address).

SR Egress: a 128 bit generic identifier representing the egress in the SR domain (i.e.: it needs not to be a valid IPv6 address).

Original Source Address: IPv6 address originally present in the SA field of the packet.

The segments in the Policy List are encoded after the segment list and they are optional. If none are in the SRH, all bits of the Policy List Flags MUST be set to 0x0.

3.2.1. SRH and RFC2460 behavior

The SRH being a new type of the Routing Header, it also has the same properties:

SHOULD only appear once in the packet.

Only the router whose address is in the DA field of the packet header MUST inspect the SRH.

Therefore, Segment Routing in IPv6 networks implies that the segment identifier (i.e.: the IPv6 address of the segment) is moved into the DA of the packet.

The DA of the packet changes at each segment termination/completion and therefore the original DA of the packet MUST be encoded as the last segment of the path.

As illustrated in Section 2.3, nodes that are within the path of a segment will forward packets based on the DA of the packet without inspecting the SRH. This ensures full interoperability between SR-capable and non-SR-capable nodes.

4. SRH Procedures

In this section we describe the different procedures on the SRH.

4.1. Segment Routing Node Functions

SR packets are forwarded to segments endpoints (i.e.: the segment endpoint is the node representing the segment and whose address is in the segment list and in the DA of the packet when traveling in the segment). The segment endpoint, when receiving a SR packet destined to itself, does:

- o Inspect the SRH.
- o Determine the next active segment.
- o Update the Segments Left field (or, if requested, remove the SRH from the packet).
- o Update the DA.
- o Forward the packet to the next segment.

The procedures applied to the SRH are related to the node function. Following nodes functions are defined:

Source SR Node.

SR Domain Ingress Node.

Transit Node.

SR Endpoint Node.

4.1.1.1. Source SR Node

A Source SR Node can be any node originating an IPv6 packet with its IPv6 and Segment Routing Headers. This include either:

A host originating an IPv6 packet

A SR domain ingress router encapsulating a received IPv6 packet into an outer IPv6 header followed by a SRH

The mechanism through which a Segment List is derived is outside of the scope of this document. As an example, the Segment List may be obtained through:

Local path computation.

Local configuration.

Interaction with a centralized controller delivering the path.

Any other mechanism.

The following are the steps of the creation of the SRH:

Next Header and Hdr Ext Len fields are set according to [RFC2460].

Routing Type field is set as TBD (SRH).

The Segment List is built with the FIRST segment of the path encoded in the LAST element of the Segment List. Subsequent segments are encoded on top of the first segment. Finally, the LAST segment of the path is encoded in the FIRST element of the Segment List. In other words, the Segment List is encoded in the reverse order of the path.

The original DA of the packet is encoded as the last segment of the path (encoded in the first element of the Segment List).

The DA of the packet is set with the value of the first segment (found in the last element of the segment list).

The Segments Left field is set to n-1 where n is the number of elements in the Segment List.

The First Segment field is set to n-1 where n is the number of elements in the Segment List.

The packet is sent out towards the first segment (i.e.: represented in the packet DA).

HMAC and HMAC Key ID may be set according to Section 5.

4.1.2. SR Domain Ingress Node

The SR Domain Ingress Node is the node where ingress policies are applied and where the packet path (and processing) is determined.

After policies are applied and packet classification is done, the result may be instantiated into a Segment List representing the path the packet should take. In such case, the SR Domain Ingress Node instantiate a new outer IPv6 header to which the SRH is appended (with the computed Segment List). The procedures for the creation and insertion of the new SRH are described in Section 4.1.1.

4.1.3. Transit Node

According to [RFC2460], the only node who is allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet towards the DA and according to the IPv6 routing table.

In the example case described in Section 2.2.2, when SR capable nodes are connected through an overlay spanning multiple third-party infrastructure, it is safe to send SRH packets (i.e.: packet having a Segment Routing Header) between each other overlay/SR-capable nodes as long as the segment list does not include any of the transit provider nodes. In addition, as a generic security measure, any service provider will block any packet destined to one of its internal routers, especially if these packets have an extended header in it.

4.1.4. SR Segment Endpoint Node

The SR segment endpoint node is the node whose address is in the DA. The segment endpoint node inspects the SRH and does:

1. IF DA = myself (segment endpoint)
2. IF Segments Left > 0 THEN
 decrement Segments Left
 update DA with Segment List[Segments Left]
3. IF Segments Left == 0 THEN
 IF Clean-up bit is set THEN remove the SRH
4. ELSE give the packet to next PID (application)
 End of processing.
5. Forward the packet out

5. Security Considerations

This section analyzes the security threat model, the security issues and mitigation techniques of SRH.

SRH is simply another type of the routing header as described in RFC 2460 [RFC2460] and is:

- o added to a new outer IP header by the ingress router when entering the SR domain or by the originating node itself. The source host can be outside the SR domain;
- o inspected and acted upon when reaching the destination address of the IP header per RFC 2460 [RFC2460].

Per RFC2460 [RFC2460], routers on the path that simply forward an IPv6 packet (i.e. the IPv6 destination address is none of theirs) will never inspect and process the content of any routing header (including SRH). Routers whose one interface IPv6 address equals the destination address field of the IPv6 packet MUST to parse the SRH and, if supported and if the local configuration allows it, MUST act accordingly to the SRH content.

According to RFC2460 [RFC2460], non SR-capable (or non SR-configured) router upon receipt of an IPv6 packet with SRH destined to an address of its:

- o must ignore the SRH completely if the Segment Left field is 0 and proceed to process the next header in the IPv6 packet;
- o must discard the IPv6 packet if Segment Left field is greater than 0 and send a Parameter Problem ICMP message back to the Source Address.

5.1. Threat model

5.1.1. Source routing threats

Using a SRH is a specific case of loose source routing, therefore it has some well-known security issues as described in RFC4942 [RFC4942] section 2.1.1 and RFC5095 [RFC5095]:

- o amplification attacks: where a packet could be forged in such a way to cause looping among a set of SR-enabled routers causing unnecessary traffic, hence a Denial of Service (DoS) against bandwidth;
- o reflection attack: where a hacker could force an intermediate node to appear as the immediate attacker, hence hiding the real attacker from naive forensic;
- o bypass attack: where an intermediate node could be used as a stepping stone (for example in a De-Militarized Zone) to attack another host (for example in the datacenter or any back-end server).

5.1.2. Applicability of RFC 5095 to SRH

First of all, the reader must remember this specific part of section 1 of RFC5095 [RFC5095], "A side effect is that this also eliminates benign RH0 use-cases; however, such applications may be facilitated by future Routing Header specifications.". In short, it is not forbidden to create new secure type of Routing Header; for example, RFC 6554 (RPL) [RFC6554] also creates a new Routing Header type for a specific application confined in a single network.

The main use case for SR consists of the single administrative domain (or cooperating administrative domains) where only trusted nodes with SR enabled and explicitly configured participate in SR: this is the same model as in RFC6554 [RFC6554]. All non-trusted nodes do not participate as either SR processing is not enabled by default or because they only process SRH from nodes within their domain.

Moreover, all SR routers SHOULD ignore SRH created by outsiders based on topology information (received on a peering or internal interface) or on presence and validity of the HMAC field. Therefore, if intermediate SR routers ONLY act on valid and authorized SRH (such as within a single administrative domain), then there is no security threat similar to RH-0. Hence, the RFC 5095 [RFC5095] attacks are not applicable.

5.1.3. Service stealing threat

Segment routing is used for added value services, there is also a need to prevent non-participating nodes to use those services; this is called 'service stealing prevention'.

5.1.4. Topology disclosure

The SRH may also contains IPv6 addresses of some intermediate SR routers in the path towards the destination, this obviously reveals those addresses to the potentially hostile attackers if those attackers are able to intercept packets containing SRH. On the other hand, if the attacker can do a traceroute whose probes will be forwarded along the SR path, then there is little learned by intercepting the SRH itself. The clean-bit of SRH can help by removing the SRH before forwarding the packet to potentially a non-trusted part of the network; if the attacker can force the generation of an ICMP message during the transit in the SR domain, then the ICMP will probably contain the SRH header (totally or partially) depending on the ICMP-generating router behavior.

5.1.5. ICMP Generation

Per section 4.4 of RFC2460 [RFC2460], when destination nodes (i.e. where the destination address is one of theirs) receive a Routing Header with unsupported Routing Type, the required behavior is:

- o If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet.
- o If Segments Left is non-zero, the node must discard the packet and SHOULD send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

This required behavior could be used by an attacker to force the generation of ICMP message by any node. The attacker could send packets with SRH (with Segment Left different than 0) destined to a node not supporting SRH. Per RFC2460 [RFC2460], the destination node must then generate an ICMP message per RFC 2460, causing a local CPU utilization and if the source of the offending packet with SRH was spoofed could lead to a reflection attack without any amplification.

It must be noted that this is a required behavior for any unsupported Routing Type and not limited to SRH packets. So, it is not specific to SRH and the usual rate limiting for ICMP generation is required anyway for any IPv6 implementation and has been implemented and deployed for many years.

5.2. Security fields in SRH

This section summarizes the use of specific fields in the SRH. They are based on a key-hashed message authentication code (HMAC).

The security-related fields in SRH are:

- o HMAC Key-id, 8 bits wide;
- o HMAC, 256 bits wide (optional, exists only if HMAC Key-id is not 0).

The HMAC field is the output of the HMAC computation (per RFC 2104 [RFC2104]) using a pre-shared key and hashing algorithm identified by HMAC Key-id and of the text which consists of the concatenation of:

- o the source IPv6 address;
- o First Segment field;
- o an octet whose bit-0 is the clean-up bit flag and others are 0;
- o HMAC Key-id;
- o all addresses in the Segment List.

The purpose of the HMAC field is to verify the validity, the integrity and the authorization of the SRH itself. If an outsider of the SR domain does not have access to a current pre-shared secret, then it cannot compute the right HMAC field and the first SR router on the path processing the SRH and configured to check the validity of the HMAC will simply reject the packet.

The HMAC field is located at the end of the SRH simply because only the router on the ingress of the SR domain needs to process it, then all other SR nodes can ignore it (based on local policy) because they trust the upstream router. This is to speed up forwarding operations because SR routers which do not validate the SRH do not need to parse the SRH until the end.

The HMAC Key-id field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys. This allows for pre-shared key roll-over when two pre-shared keys are supported for a while when all SR nodes converged to a fresher pre-shared key. The HMAC Key-id field is opaque, i.e., it has neither syntax nor semantic except as an index to the right combination of pre-shared key and hash algorithm and except that a value of 0 means that there is no HMAC field. It could

also allow for interoperation among different SR domains if allowed by local policy and assuming a collision-free Key Id allocation which is out of scope of this memo.

When a specific SRH is linked to a time-related service (such as turbo-QoS for a 1-hour period), then it is important to refresh the shared-secret frequently as the HMAC validity period expires only when the HMAC Key-id and its associated shared-secret expires.

5.2.1. Selecting a hash algorithm

The HMAC field in the SRH is 256 bits wide. Therefore, the HMAC MUST be based on a hash function whose output is at least 256 bits. If the output of the hash function is 256, then this output is simply inserted in the HMAC field. If the output of the hash function is larger than 256 bits, then the output value is truncated to 256 by taking the least-significant 256 bits and inserting them in the HMAC field.

SRH implementations can support multiple hash functions but MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

NOTE: SHA-1 is currently used by some early implementations used for quick interoperations testing, the 160-bit hash value must then be right-hand padded with 96 bits set to 0. The authors understand that this is not secure but is ok for limited tests.

5.2.2. Performance impact of HMAC

While adding a HMAC to each and every SR packet increases the security, it has a performance impact. Nevertheless, it must be noted that:

- o the HMAC field SHOULD be used only when SRH is inserted by a device (such as a home set-up box) which is outside of the segment routing domain. If the SRH is added by a router in the trusted segment routing domain, then, there is no need for a HMAC field, hence no performance impact.
- o when present, the HMAC field MUST be checked and validated only by the first router of the segment routing domain, this router is named 'validating SR router'. Downstream routers may not inspect the HMAC field.
- o this validating router can also have a cache of <IPv6 header + SRH, HMAC field value> to improve the performance. It is not the same use case as in IPsec where HMAC value was unique per packet, in SRH, the HMAC value is unique per flow.

- o Last point, hash functions such as SHA-2 have been optimized for security and performance and there are multiple implementations with good performance.

With the above points in mind, the performance impact of using HMAC is minimized.

5.2.3. Pre-shared key management

The field HMAC Key-id allows for:

- o key roll-over: when there is a need to change the key (the hash pre-shared secret), then multiple pre-shared keys can be used simultaneously. The validating routing can have a table of <HMAC Key-id, pre-shared secret, hash algorithm> for the currently active and future keys.
- o different algorithm: by extending the previous table to <HMAC Key-id, hash function, pre-shared secret>, the validating router can also support simultaneously several hash algorithms (see section Section 5.2.1)

The pre-shared secret distribution can be done:

- o in the configuration of the validating routers, either by static configuration or any SDN oriented approach;
- o dynamically using a trusted key distribution such as [RFC6407]

The intent of this document is NOT to define yet-another-key-distribution-protocol.

5.3. Deployment Models

5.3.1. Nodes within the SR domain

The routers inside a SR domain can be trusted to generate the outer IP header and the SRH and to process SRH received on interfaces that are part of the SR domain. These nodes MUST drop all SRH packets received on any interface that is not part of the SR domain and containing a SRH whose HMAC field cannot be validated by local policies. This includes obviously packet with a SRH generated by a non-cooperative SR domain.

If the validation fails, then these packets MUST be dropped, ICMP error messages (parameter problem) SHOULD be generated (but rate limited) and SHOULD be logged.

5.3.2. Nodes outside of the SR domain

Nodes outside of the SR domain cannot be trusted for physical security; hence, they need to obtain by some trusted means (outside of the scope of this document) a complete SRH for each new connection (i.e. new destination address). The received SRH MUST include a HMAC Key-id and HMAC field which has been computed correctly (see Section 5.2).

When a outside the SR domain sends a packet with a SRH and towards a SR domain ingress node, the packet MUST contain the HMAC Key-id and HMAC field and the destination address MUST be an address of a SR domain ingress node .

The ingress SR router, i.e., the router with an interface address equals to the destination address, MUST verify the HMAC field with respect to the HMAC Key-id.

If the validation is successful, then the packet is simply forwarded as usual for a SR packet. As long as the packet travels within the SR domain, no further HMAC check needs to be done. Subsequent routers in the SR domain MAY verify the HMAC field when they process the SRH (i.e. when they are the destination).

If the validation fails, then this packet MUST be dropped, an ICMP error message (parameter problem) SHOULD be generated (but rate limited) and SHOULD be logged.

5.3.3. SR path exposure

As the intermediate SR nodes addresses appears in the SRH, if this SRH is visible to an outsider then he/she could reuse this knowledge to launch an attack on the intermediate SR nodes or get some insider knowledge on the topology. This is especially applicable when the path between the source node and the first SR domain ingress router is on the public Internet.

The first remark is to state that 'security by obscurity' is never enough; in other words, the security policy of the SR domain SHOULD assume that the internal topology and addressing is known by the attacker.

IPsec Encapsulating Security Payload [RFC4303] cannot be use to protect the SRH as per RFC4303 the ESP header must appear after any routing header (including SRH).

When the SRH is not generated by the actual source node but by an SR domain ingress router, it is added after a new outer IP header, this

means that a normal traceroute will not reveal the routers in the SR domain (pretty much like in a MPLS network) and that if ICMP are generated by routers in the SR domain they will be sent to the ingress router of the SR domain without revealing anything to the outside of the SR domain.

To prevent a user to leverage the gained knowledge by intercepting SRH, it is recommended to apply an infrastructure Access Control List (iACL) at the edge of the SR domain. This iACL will drop all packets from outside the SR-domain whose destination is any address of any router inside the domain. This security policy should be tuned for local operations.

5.3.4. Impact of BCP-38

BCP-38 [RFC2827], also known as "Network Ingress Filtering", checks whether the source address of packets received on an interface is valid for this interface. The use of loose source routing such as SRH forces packets to follow a path which differs from the expected routing. Therefore, if BCP-38 was implemented in all routers inside the SR domain, then SR packets could be received by an interface which is not expected one and the packets could be dropped.

As a SR domain is usually a subset of one administrative domain, and as BCP-38 is only deployed at the ingress routers of this administrative domain and as packets arriving at those ingress routers have been normally forwarded using the normal routing information, then there is no reason why this ingress router should drop the SRH packet based on BCP-38. Routers inside the domain commonly do not apply BCP-38; so, this is not a problem.

6. IANA Considerations

TBD but should at least require a new type for routing header

7. Manageability Considerations

TBD should we talk about traceroute? about SRH in ICMP replies?

8. Contributors

The authors would like to thank Dave Barach, John Leddy, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin and Fred Baker for their contribution to this document.

9. Acknowledgements

TBD

10. References

10.1. Normative References

[FIPS180-4]

National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <<http://www.rfc-editor.org/info/rfc5095>>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<http://www.rfc-editor.org/info/rfc6407>>.

10.2. Informative References

[I-D.ietf-isis-segment-routing-extensions]

Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-05 (work in progress), June 2015.

- [I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,
Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3
Extensions for Segment Routing", draft-ietf-ospf-ospfv3-
segment-routing-extensions-03 (work in progress), June
2015.
- [I-D.ietf-spring-ipv6-use-cases]
Brzozowski, J., Leddy, J., Leung, I., Previdi, S.,
Townsend, W., Martin, C., Filsfils, C., and R. Maglione,
"IPv6 SPRING Use Cases", draft-ietf-spring-ipv6-use-
cases-05 (work in progress), September 2015.
- [I-D.ietf-spring-problem-statement]
Previdi, S., Filsfils, C., Decraene, B., Litkowski, S.,
Horneffer, M., and R. Shakir, "SPRING Problem Statement
and Requirements", draft-ietf-spring-problem-statement-04
(work in progress), April 2015.
- [I-D.ietf-spring-resiliency-use-cases]
Francois, P., Filsfils, C., Decraene, B., and R. Shakir,
"Use-cases for Resiliency in SPRING", draft-ietf-spring-
resiliency-use-cases-01 (work in progress), March 2015.
- [I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S.,
and r. rjs@rob.sh, "Segment Routing Architecture", draft-
ietf-spring-segment-routing-05 (work in progress),
September 2015.
- [I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J.,
and E. Crabbe, "Segment Routing with MPLS data plane",
draft-ietf-spring-segment-routing-mpls-01 (work in
progress), May 2015.
- [RFC1940] Estrin, D., Li, T., Rekhter, Y., Varadhan, K., and D.
Zappala, "Source Demand Routing: Packet Format and
Forwarding Specification (Version 1)", RFC 1940,
DOI 10.17487/RFC1940, May 1996,
<<http://www.rfc-editor.org/info/rfc1940>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
Hashing for Message Authentication", RFC 2104,
DOI 10.17487/RFC2104, February 1997,
<<http://www.rfc-editor.org/info/rfc2104>>.

- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<http://www.rfc-editor.org/info/rfc2827>>.
- [RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/ Co-existence Security Considerations", RFC 4942, DOI 10.17487/RFC4942, September 2007, <<http://www.rfc-editor.org/info/rfc4942>>.
- [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, DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.

Authors' Addresses

Stefano Previdi (editor)
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: sprevidi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Brian Field
Comcast
4100 East Dry Creek Road
Centennial, CO 80122
US

Email: Brian_Field@cable.comcast.com

Ida Leung
Rogers Communications
8200 Dixie Road
Brampton, ON L6T 0C1
CA

Email: Ida.Leung@rci.rogers.com

Jen Linkova
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: furry@google.com

Ebben Aries
Facebook
US

Email: exa@fb.com

Tomoya Kosugi
NTT
3-9-11, Midori-Cho Musashino-Shi,
Tokyo 180-8585
JP

Email: kosugi.tomoya@lab.ntt.co.jp

Eric Vyncke
Cisco Systems, Inc.
De Kleetlaann 6A
Diegem 1831
Belgium

Email: evyncke@cisco.com

David Lebrun
Universite Catholique de Louvain
Place Ste Barbe, 2
Louvain-la-Neuve, 1348
Belgium

Email: david.lebrun@uclouvain.be

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 22, 2016

I. van Beijnum
Institute IMDEA Networks
March 21, 2016

Extensions for Multi-MTU Subnets
draft-van-beijnum-multi-mtu-05

Abstract

In the early days of the internet, many different link types with many different maximum packet sizes were in use. For point-to-point or point-to-multipoint links, there are still some other link types (PPP, ATM, Packet over SONET), but multipoint subnets are now almost exclusively implemented as Ethernets. Even though the relevant standards mandate a 1500 byte maximum packet size for Ethernet, more and more Ethernet equipment is capable of handling packets bigger than 1500 bytes. However, since this capability isn't standardized, it is seldom used today, despite the potential performance benefits of using larger packets. This document specifies mechanisms to negotiate per-neighbor maximum packet sizes so that nodes on a multipoint subnet may use the maximum mutually supported packet size between them without being limited by nodes with smaller maximum sizes on the same 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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 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. Notational Conventions	4
3. Terminology	4
4. Overview of operation	5
5. The ND NODEMTU option	6
6. The MTUTEST packet format	7
7. Changes to the RA MTU option semantics	8
8. The TCP MSS option	9
9. Operation	9
9.1. Initialization	9
9.2. Probing	10
9.3. Monitoring	14
9.4. Neighbor MTU garbage collection	16
10. Applicability	16
11. IANA considerations	16
12. Security considerations	16
13. Acknowledgements	17
14. References	17
14.1. Normative References	17
14.2. Informative References	18
Appendix A. Document and discussion information	19
Appendix B. Advantages and disadvantages of larger packets	19
B.1. Clock skew	19
B.2. ECMP over paths with different MTUs	20
B.3. Delay and jitter	20
B.4. Path MTU Discovery problems	21
B.5. Packet loss through bit errors	21
B.6. Undetected bit errors	22
B.7. Interaction TCP congestion control	23
B.8. IEEE 802.3 compatibility	23
B.9. Conclusion	24
Author's Address	24

1. Introduction

Some protocols inherently generate small packets. Examples are VoIP, where it is necessary to send packets frequently before much data can be gathered to fill up the packet, and the DNS, where the queries are inherently small and the returned results also often do not fill up a full 1500-byte packet. However, most data that is transferred across the internet and private networks is part of long-lived sessions and requires segmentation by a transport protocol, which is almost always TCP. These types of data transfers can benefit from larger packets in several ways:

1. A higher data-to-header ratio makes for fewer overhead bytes
2. Fewer packets means fewer per-packet operations for the source and destination hosts
3. Fewer packets also means fewer per-packet operations in routers and middleboxes
4. TCP performance increases with larger packet sizes

Even though today, the capability to use larger packets (often called jumboframes) is present in a lot of Ethernet hardware, this capability typically isn't used because IP assumes a common MTU size for all nodes connected to a link or subnet. In practice, this means that using a larger MTU requires manual configuration of the non-standard MTU size on all hosts and routers and possibly on layer 2 switches connected to a subnet. Also, the MTU size for a subnet is limited to that of the least capable router, host or switch.

Perhaps in the future, when hosts support packetization layer path MTU discovery ([RFC4821], "Packetization Layer Path MTU Discovery") in all relevant transport protocols, it will be possible to simply ignore MTU limitations by sending at the maximum locally supported size and determining the maximum packet size towards a correspondent from acknowledgements that come back for packets of different sizes. However, [RFC4821] must be implemented in every transport protocol, and problems arise in the case where hosts implementing [RFC4821] interact with hosts that don't implement this mechanism, but do use a larger than standard MTU.

This document provides for a set of mechanisms that allow the use of larger packets between nodes that support them which interacts well with both manually configured non-standard MTUs and expected future [RFC4821] operation with larger MTUs. This is done using a new IPv6 Neighbor Discovery option and a new UDP-based protocol for exchanging

MTU information and testing whether jumboframes can be transmitted successfully.

Appendix B discusses several potential issues with larger packets, such as head-of-line blocking delays, path MTU discovery black holes and the strength of the CRC32 with increasing packet sizes.

2. Notational Conventions

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

Note that this specification is not standards track, and as such, can't overrule existing specifications. Whenever [RFC2119] language is used, this must be interpreted within the context of this specification: while the specification as a whole is optional and non-standard, whenever it is implemented, such an implementation can only function properly when all MUSTs are observed.

3. Terminology

Advertised MTU: The MTU size announced by a node to other nodes on the local subnet.

Confirmed MTU: The largest packet size successfully received from the neighbor or the largest packet size sent to the neighbor for which an acknowledgment was received; whichever size is greater.

Confirmed Time: When a packet the size of the confirmed MTU was last received or acknowledged.

Local MTU: The MTU configured on an interface. By default, this is the largest MTU size supported by the hardware, but the Local MTU may be lowered administratively or automatically based on policy. (For instance, the MTU may be set to the Standard MTU if the link speed is below 1000 Mbps.)

MRU: Maximum Receive Unit. The size of the largest IP packet that can be received on an interface. This document doesn't use the term MRU, and assumes that the MRU is equal to the MTU.

MTU: Maximum Transfer Unit. The size of the largest IP packet that can be transmitted on an interface, considering hardware (and administrative) limitations.

Neighbor: Another node on a connected subnet. Neighbors are identified by the combination of a link address and an IP version.

The MTU may be set to different values for IPv4 and IPv6 administratively, but it is assumed that if a node has multiple IPv4 or IPv6 addresses, the MTU for each set of addresses is the same.

Neighbor MTU: The currently used MTU towards a neighboring node on a subnet. The Neighbor MTU reflects the current best understanding of the maximum packet size that can successfully be transmitted towards that neighbor.

Safe MTU: The maximum packet size that is assumed to work without testing. Defaults to the Standard MTU, but may be set to a subnet-wide higher or lower value administratively, or to a lower value using the MTU option in IPv6 Router Advertisements.

Standard MTU: The MTU specified in the relevant IPv4-over-... or IPv6-over-... document, which is 1500 for Ethernet ([RFC0894] and [RFC2464]).

4. Overview of operation

The mechanisms described in this document come into play when a node is connected to a subnet using an interface that supports an MTU size larger than the standard MTU size for that link type.

For each remote node connected to such a subnet, the local node maintains a neighbor MTU setting. The length of packets transmitted to a neighbor is always limited to the neighbor MTU size.

When a node starts communicating with another node on the same subnet, it follows the following procedure:

1. Initialization: the neighbor MTU is set to local maximum MTU for the interface used to reach the neighbor.
2. Discovery: learning the other node's MTU.
3. Probing: determining the maximum packet size that can successfully be transmitted to and received from the other node, considering the (unknown) maximum packet size supported by the layer 2 infrastructure.
4. Monitoring: making sure that when large packets are transmitted, they are not silently discarded, for instance as the result of a layer 2 reconfiguration.

During the discovery and probing stages, the neighbor MTU is adjusted as new information becomes available. The monitoring stage is

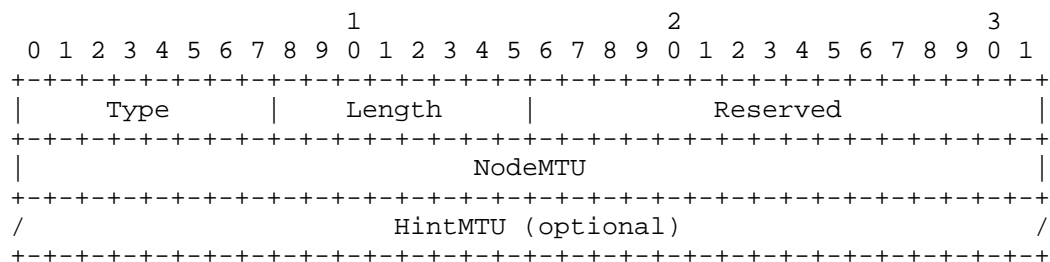
ongoing. If during the monitoring stage it is determined that large packets aren't successfully exchanged with the neighboring node, the neighbor MTU is set to the safe MTU and the node returns to the testing stage.

Unless administrative configuration or policy specifies otherwise, the link, IPv4 and IPv6 MTU sizes are set to the maximum supported by the hardware. This means that when TCP sessions are created, they carry a maximum segment size (MSS) option that reflects the larger-than-standard MTU.

5. The ND NODEMTU option

All MTU values are 32-bit unsigned integers in network byte order. All other values are also unsigned and in network byte order .

The MTU size and two flags are exchanged as an IPv6 Neighbor Discovery option. The new option, as well as the MTU value it advertises, are named "NODEMTU".



Type: TBD

Length: 1 or 2

Reserved: Set to 0 on transmission, ignored when received.

NodeMTU The maximum packet size the node wishes to receive on this interface.

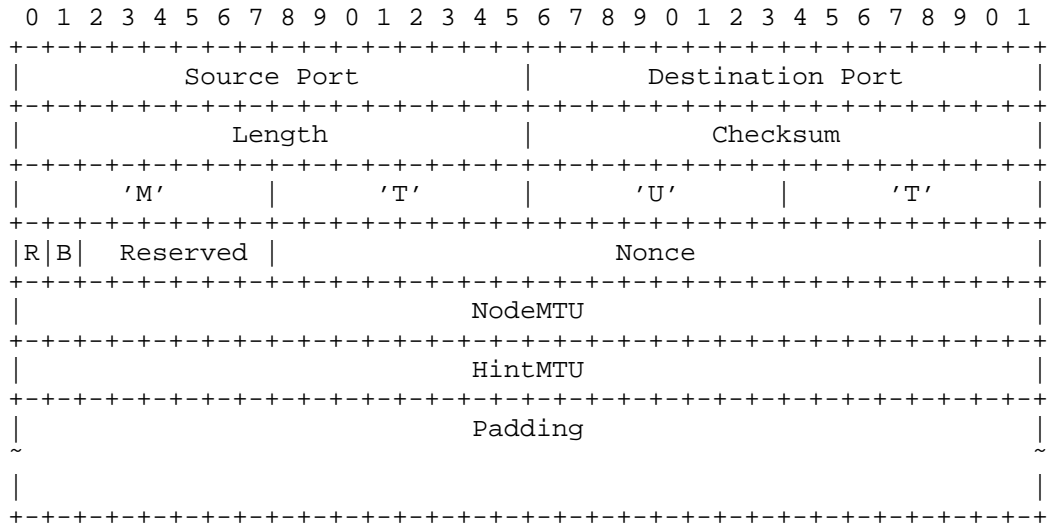
HintMTU The maximum packet size the node believes it can successfully receive on this interface at this time. If the HintMTU is equal to the NodeMTU or no value for HintMTU is known, this field may be omitted and the Length field is set to 1. If the HintMTU field is present, the Length field is set to 2.

When a node's interface speed changes, it MAY advertise a new MTU, but it SHOULD remain prepared to receive packets of the maximum size

advertised to neighbors previously (if the old maximum size is larger than the newly advertised one).

6. The MTUTEST packet format

The packets used to test whether large packets can be transmitted successfully and communicate status are sent using UDP ([RFC0768]). Their format is as follows:



Source port (UDP): For outgoing requests: an ephemeral port number.
For replies: 1022. (16 bits.)

Destination port (UDP): For outgoing requests: 1022. For replies:
the source port used in the request being replied to. (16 bits.)

Length (UDP): for IPv4 and IPv6 packets smaller than or equal to
65575 bytes, the length of the UDP segment. For IPv6 packets
larger than 65575 bytes, 0 (as per [RFC2675]). (16 bits.)

Checksum (UDP): the UDP checksum. (16 bits.)

R: reply request flag. If set to 0, no reply is sent. If set to 1,
the receiver is asked to send a reply. (1 bit.)

MTUT: The value corresponding to the ASCII string "MTUT", used to
differentiate MTUTEST packets from other UDP packets that use port
1022. Packets with a value other than "MTUT" at the beginning of
the UDP payload MUST be ignored. (32 bits.)

B: big reply request flag. If set to 0, replies are not padded. If set to 1, replies are padded to be the same size as the request. (1 bit.)

Reserved: set to 0 on transmission, ignored on reception. (6 bits.)

Nonce: a hard-to-guess value. (24 bits.)

NodeMTU: The maximum packet size that the sender is prepared to receive at this time. (32 bits.)

HintMTU: The maximum packet size that the sender believes it can successfully receive at this time. (32 bits.)

Padding: Filled with 0 or more all-zero bytes on transmission, ignored on reception.

In addition to the fields listed above, the following IP and link layer fields are taken into consideration:

Source link-layer address: On transmission: set automatically by the networking stack. On reception: used to identify a neighbor.

IP version: On transmission: set automatically by the networking stack. On reception: used to identify a neighbor. (The IP version may also be identified implicitly through the API without directly observing the version field.)

Time To Live / Hop Limit: On transmission: set to 255. On reception: if 255, the packet is processed. If other than 255, the packet is silently discarded. (To enforce that the protocol is only used within a local subnet.)

Source IP address: On transmission, for requests: set to the address the node intends to use to communicate with the neighbor. For replies: set to the destination IP address in the request being replied to. On reception: used to identify a neighbor.

Destination IP address: On transmission, for requests: set to the address the node intends to use to communicate with the neighbor. For replies: set to the source IP address in the request being replied to.

7. Changes to the RA MTU option semantics

Section 6.3.4 of [RFC4861] specifies:

"If the MTU option is present, hosts SHOULD copy the option's value into LinkMTU so long as the value is greater than or equal to the minimum link MTU and does not exceed the maximum LinkMTU value specified in the link-type-specific document"

This document changes the handling of the Router Advertisement MTU option such that it may also be used by routers to tell hosts that they SHOULD use an MTU larger than the LinkMTU and update their SafeMTU value. If multiple routers advertise different MTUs that are higher or lower than the standard MTU, behavior is undefined. MTU options containing the standard MTU SHOULD be ignored.

The ability to advertise a larger-than-standard MTU must be used with extreme care by network administrators, as advertising an MTU size that exceeds the capabilities of routers or the layer 2 infrastructure will lead to reachability problems.

If the advertised larger-than-standard MTU is ignored or not supported by some hosts connected to the subnet, TCP will presumably still work because the MSS option ([RFC0793]) limits the size of transmitted TCP segments to what the receiver supports. However, non-TCP protocols that use large packets will likely fail. The most prominent example of this is DNS over UDP with EDNS0 when requesting large records, such as those used for DNSSEC ([RFC6891]).

8. The TCP MSS option

Hosts SHOULD advertise the maximum MTU size they are prepared to use on a link in the TCP MSS value, even during times when probing has failed: should larger neighbor MTUs be established later, it will not be possible to adjust the MSS for ongoing sessions.

9. Operation

9.1. Initialization

When an interface is activated, an appropriate local MTU is determined, based on hardware limitations and administrative settings. Additionally, a policy may be in place to constrain packet sizes when operating at lower bandwidths, to avoid excessive delays as queues of large packets build up and cause significant head-of-line blocking for subsequent time-sensitive packets. Also, layer 2 devices operating at lower interface speeds are less likely to support non-standard MTUs.

In the absence of operational experience, this document RECOMMENDS limiting the use of larger than standard MTUs to interfaces operating at 400 Mbps or faster; and if a larger MTU is used for interfaces

operating at lower speeds, a "mini jumbo" size of 1982 bytes or less is used for Ethernets.

For IPv4, the local MTU is limited to 65535 bytes. For IPv6, if [RFC2675] jumbograms are not supported, the local MTU is limited to 65575 bytes. These limits apply even if the interface hardware supports a larger MTU. IPv6 nodes that implement [RFC2675] jumbograms MAY use MTU sizes larger than 65575 bytes.

When the interface speed changes, the local MTU MAY be changed to reflect the new speed. However, the node SHOULD remain prepared to receive packets of the size of a previously advertised MTU.

The local MTU MAY be different for IPv4 and IPv6. The local MTU is the size used to calculate the value of the TCP MSS option. The HintMTU is set to undefined.

When sending Neighbor Solicitations and Neighbor Advertisements, a node includes its local MTU in the NodeMTU field of the NODEMTU option. If the size of the HintMTU is known, it is also included.

9.2. Probing

When a node starts communicating with a new IPv4 or IPv6 neighbor, the probing procedure is started. This can happen when ARP [RFC0826] or Neighbor Discovery messages are exchanged, or when an incoming TCP SYN is received.

The node sends a MTUTEST packet to the new neighbor and sets the neighbor MTU to the safe MTU. The MTUTEST packet has the local MTU in the NodeMTU field. If a hint MTU is known, it is included in the HintMTU field. The R and B flags are set to 0. No padding is included.

Upon reception of a Neighbor Solicitation or a Neighbor Advertisement with the NODEMTU option or an MTUTEST packet, the node determines if the packet is received from a known neighbor IP address and a known neighbor link layer address. If the values match the values stored for a known neighbor, no action occurs.

If the values match the values for a known link layer address and IP version, but an unknown IP address, the IP address is added to the list of IP addresses for the neighbor in question and the known neighbor MTU for the neighbor is applied to the new address.

If the NodeMTU matches the NodeMTU previously sent by a known neighbor but the HintMTU as a different non-zero value, the HintMTU is updated.

If the HintMTU sent by a known neighbor is 0, the neighbor MTU is set to the safe MTU, the HintMTU for the neighbor is set to unknown and the probing procedure is started.

If the combination of link layer address and IP version is unknown, the neighbor MTU is set to the safe MTU, the HintMTU is set to the HintMTU value in the packet and the probing procedure is started.

Before starting the probing procedure, a node compares its link layer address to the neighbor's link layer address. If the node's link layer address is numerically larger than the neighbor's link layer address, the node applies a waiting period before starting the probing procedure. The waiting period SHOULD be at least 250 milliseconds and at most 1 second.

The following is pseudo-code for a probing procedure. Note that it differs from the one outlined in [RFC4821]. The latter favors conservative probing because lost probes can't easily be differentiated from congestion losses, so lost probes are expensive. For this specification, successful probes waste bandwidth and losses are less problematic, so more aggressive probing and failing quickly is more appropriate.

```
Neighbor.ConfirmedTime = UNDEFINED

if LocalMTU > Neighbor.AdvertisedMTU
  let Max = Neighbor.AdvertisedMTU
else
  let Max = LocalMTU

# test with maximum supported packet size first
# and finish probing upon success
test (Max)
if Success:
  Neighbor.MTU = Max
  return

# maximum size doesn't work, now find
# what does work
# assumption: 256 works for IPv4, 1280 for IPv6
let WorksNo = Max
if IPv6:
  let Neighbor.ConfirmedMTU = 1280
if IPv4:
  let Neighbor.ConfirmedMTU = 256

# test with the hinted size
# if successful, this becomes the minimum for further tests
```

```
# if unsuccessful, this becomes the maximum
test (HintMTU)
if Success:
    let Neighbor.ConfirmedMTU = HintMTU
else
    let WorksNo = HintMTU

# test the smallest usable size larger than
# the standard MTU (if that size is still
# in the range to be tested) so we avoid wasting
# time probing non-jumbo-capable nodes
if (StandardMTU + 8 > Neighbor.ConfirmedMTU and \
    StandardMTU + 8 < WorksNo)
    test (StandardMTU + 8)
    if Success:
        let Neighbor.ConfirmedMTU = StandardMTU + 8
    else
        let WorksNo = StandardMTU + 8

# to establish an upper bound quickly,
# test (320, 640, 1280, ) 2560, 5120, 10240, 20480, 40960, ...
let Current = 320
while (Current < WorksNo)
    if (Current > Neighbor.ConfirmedMTU)
        test (Current)
        if Success:
            let Neighbor.ConfirmedMTU = Current
        else
            let WorksNo = Current
    let Current = Current * 2

# we have now established that
# WorksNo =< Neighbor.ConfirmedMTU * 2

# further testing is based on a list of hints.
# there SHOULD be a mechanism for administrators
# to add hints.
#
# hint sources:
# 576: common PPP low delay
# 1492: PPP over Ethernet [RFC2516]
# 1500: Ethernet II
# 1982: IEEE Std 802.3as-2006
# 2304: IEEE 802.11
# 2482: Fibre Channel over Ethernet (FCoE)
# [CATALYST]:
# 9216, 8092, 1600, 1998, 2000, 1546, 1530, 17976, 2018
# sizes observed by the author:
```

```
# 576, 1982, 4070, 9000, 16384, 64000
let Hints = 576, 1492, 1530, 1982, 2304, 4070, 8092, 9000, \
            16384, 32000, 64000

foreach Size in Hints
  if Size > Neighbor.ConfirmedMTU and Size < WorksNo
    test (Size)
    if Success:
      let Neighbor.ConfirmedMTU = Size
    else
      let WorksNo = Size

# finished testing, maximum working packet size
# is now known to within about a factor 1.5,
# depending on the number of hints

if Neighbor.ConfirmedTime <> UNDEFINED
  # we got at least one probe back, use discovered MTU
  Neighbor.MTU = Neighbor.ConfirmedMTU
else
  # we never got any probes back, neighbor probably does
  # not implement MTUTEST protocol, so we use the safe MTU
  Neighbor.MTU = SafeMTU

# done!
return

# sending probes
function test (Size)

# wait 20 milliseconds between sending probes
let MsecSinceProbe = now () - ProbeTime

if (MsecSinceProbe < 20)
  sleep (20 - MsecSinceProbe)

# create probe, request reply (but not a big one)
let Probe.TTL = 255
let Probe.ReplyFlag = 1
let Probe.BigFlag = 0
let Nonce = rand ()
let Probe.Nonce = Nonce
let Probe.NodeMTU = LocalMTU
let Probe.HintMTU = HintMTU
let Probe.Padding = pad (Size - sizeof (Probe))
send (Probe)

let ProbeTime = now ()
```

```
# wait 2000 milliseconds for reply
# (this also avoids sending packets that are too large more
# than once every two seconds)
let Success = receive (Reply, 2000)

if not Success
    return false

if not (Reply.TTL = 255 and Reply.Nonce = Nonce
    and Reply.LinkAddress = Neighbor.LinkAddress)
    return false

# valid reply received
# note that Neighbor.MTU is not updated yet,
# this happens after probing has finished
Neighbor.ConfirmedMTU = Reply.NodeMTU
Neighbor.ConfirmedTime = now ()
Neighbor.HintMTU = Reply.HintMTU;
if HintMTU < Size
    HintMTU = Size
return true
```

If at any time an unsolicited packet arrives from the neighbor and the ConfirmedMTU of that neighbor is smaller than the size of the packet received, the HintMTU for the neighbor is set to the size of the received packet and a probe of that size may be sent. However, as the maximum size of incoming packets may be different than the maximum supported size of outgoing packets, reception of a large packet is not sufficient to update the ConfirmedMTU. The packets that update the HintMTU do not have to be MTUTEST protocol packets.

There are no retransmissions. Both nodes run the probing procedure, so there are two opportunities to succeed. However, if both fail to determine the maximum packet size that can be used because of lost packets, the hosts will have to use a smaller packet size.

It is assumed that the maximum packet size that A can send to B is the same as the maximum packet size that B can send to A. As such, the reception of a large packet is treated the same as receiving an acknowledgment for a sent large packet.

9.3. Monitoring

Once a working neighbor MTU is found, large packets can be exchanged. Presumably, this situation will persist indefinitely. However, it is possible that the network is reconfigured and then no longer supports the MTU used between two nodes. The aim of the monitoring phase is

to detect this when it happens and establish a working MTU value before sessions time out.

For each neighbor (as defined by a unique combination of link layer address and IP version) with a neighbor MTU larger than the safe MTU, the ability to successfully send or receive large packets is monitored. In the monitoring phase, a node tracks whether it sends any packets larger than the safe MTU to a neighbor and whether it receives either acknowledgments for those packets, or it receives packets of length neighbor MTU from that neighbor. (So acknowledged outgoing packets don't have to be the maximum size supported to/from the neighbor, but incoming packets do.)

The ability to track acknowledgment of non-MTUTEST packets is not required. However, it is expected that hosts will be able to do this for TCP packets because the TCP state is readily available.

Monitoring is happens in intervals. This document RECOMMENDS that this interval is between 25 and 35 seconds for hosts and between 35 and 45 seconds for routers. At the end of each monitoring interval, if acknowledgments or large packets were received, everything is fine and the neighbor confirmed time is updated.

At the end of a monitoring interval, if no large packets were sent, everything is fine and nothing happens.

At the end of a monitoring interval, if large packets were sent, but no acknowledgments or incoming maximum size packets were seen, there may have been a network reconfiguration that has made it impossible for large packets to be transmitted successfully between the two nodes. To determine whether this is the case, the node sends an MTUTEST packet with lenght neighbor MTU. The R flag is set to 1 and the B flag SHOULD be set to 0. A random nonce and the local MTU and the hint MTU are included.

The node waits 2 seconds for a reply. If there is no reply, the probe is retransmitted and the node waits 4 seconds for a reply. If after 4 seconds there is still no reply, the node sets the hint MTU to 0 and reinitializes all of the neighbor's MTU-related information to initial values. Most notably, this means that the neighbor MTU is set to the safe MTU.

If the node sets is own hint MTU to 0 or receives a hint MTU of 0 from a neighbor using an ND or MTUTEST packet, the node MAY start sending probes to other neighbors before the monitoring interval expires. However, nodes SHOULD limit the number of probes for all neighbors combined to no more than one every two seconds. If a node has many neighbors and sending probes at one every two seconds would

take too long, it MAY reset the neighbor MTUs of all of its neighbors to the safe MTU without sending probes if at least two neighbors appear to be affected by a reduction of the maximum working packet size.

9.4. Neighbor MTU garbage collection

The MTU size for a neighbor is garbage collected along with a neighbor's link address in accordance with regular ARP and neighbor discovery timeouts. Additionally, a neighbor's MTU size is reset to unknown after dead neighbor detection declares a neighbor "dead".

10. Applicability

As discussed in annex B, all larger packets, but especially very large packet sizes have the potential to be problematic in various ways. However, jumboframes of 9000 or 9216 bytes have been supported by various vendors for a long time. As such, larger MTUs of 9 kilobytes seem safe enough for larger scale experimentation at this time, but experiments with packet sizes larger than 11 kilobytes are best done in confined and closely monitored situations.

11. IANA considerations

IANA is requested to assign a neighbor discovery option type value.

[TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/icmpv6-parameters>

UDP port 1022 is used in accordance with [RFC4727]. Presumably, unlike an ND option type value, a UDP port would be relatively easy to change when experimentation makes way for production deployment.

12. Security considerations

Generating false neighbor discovery and MTUTEST packets with large MTUs may lead to a denial-of-serve condition, just like the advertisement of other false link parameters. Requests are large and replies typically short to avoid the MTUTEST protocol being used as an amplification vector. The nonce is used together with the ephemeral UDP port number to make sure that malicious nodes cannot generate a reply to a request in the blind. Enforcement of the value 255 for Hop Limit makes sure that off-link attackers can't use the protocol to influence packet sizes remotely.

A malicious node may negotiate the use of large packets and cause head-of-line blocking, especially on slower links. However, this can

only happen if the neighbor is prepared to use large packets in the first place.

13. Acknowledgements

This document benefited from feedback by Dave Thaler, Jari Arkko, Joe Touch, Pat Thaler, David Black, Brian Carpenter, Fred Templin, Jeffrey Hammond, Mikael Abrahamsson and others.

14. References

14.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<http://www.rfc-editor.org/info/rfc768>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<http://www.rfc-editor.org/info/rfc793>>.
- [RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <<http://www.rfc-editor.org/info/rfc826>>.
- [RFC0894] Hornig, C., "A Standard for the Transmission of IP Datagrams over Ethernet Networks", STD 41, RFC 894, DOI 10.17487/RFC0894, April 1984, <<http://www.rfc-editor.org/info/rfc894>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<http://www.rfc-editor.org/info/rfc2464>>.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<http://www.rfc-editor.org/info/rfc2675>>.
- [RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <<http://www.rfc-editor.org/info/rfc2992>>.

- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <<http://www.rfc-editor.org/info/rfc4727>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<http://www.rfc-editor.org/info/rfc4821>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<http://www.rfc-editor.org/info/rfc6891>>.
- [ETHERNETII]
Digital Equipment Corporation, Intel Corporation, Xerox Corporation, "The Ethernet - A Local Area Network", September 1980, <<http://research.microsoft.com/en-us/um/people/gbell/Digital/Ethernet%20Blue%20Book.pdf>>.

14.2. Informative References

- [RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, <<http://www.rfc-editor.org/info/rfc2516>>.
- [IEEE.802.3AS_2006]
IEEE, "IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 3: Frame Format Extensions", IEEE 802.3as-2006, DOI 10.1109/ieeestd.2006.248146, November 2006, <<http://ieeexplore.ieee.org/servlet/opac?punumber=4014413>>.

[IEEE.802.3_2012]

IEEE, "802.3-2012", IEEE 802.3-2012,
DOI 10.1109/ieeestd.2012.6419735, January 2013,
<<http://ieeexplore.ieee.org/servlet/opac?punumber=6419733>>.

[CRC]

Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.

[CATALYST]

Cisco, "Jumbo/Giant Frame Support on Catalyst Switches Configuration Example",
<<http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6000-series-switches/24048-148.html>>.

Appendix A. Document and discussion information

The latest version of this document will always be available at <http://www.muada.com/drafts/>. Please direct questions and comments to the int-area mailinglist or directly to the author.

Appendix B. Advantages and disadvantages of larger packets

Although often desirable, the use of larger packets isn't universally advantageous for the following reasons:

1. Clock skew
2. ECMP over paths with different MTUs
3. Increased delay and jitter
4. Increased reliance on path MTU discovery
5. Increased packet loss through bit errors
6. Increased risk of undetected bit errors

B.1. Clock skew

Ethernet hardware has to compensate between clocking differences between the sender and receiver through a FIFO buffer. As packets get larger, more buffer capacity is required. This places a limit on packet sizes.

As jumboframes have been widely supported since the introduction of Gigabit Ethernet, and in the absence of information to the contrary,

it seems safe to assume that the packet sizes that may be set administratively fall within the capabilities of the hardware. Administrators are encouraged to monitor the fraction of packets lost from different types of corruption and adjust MTU sizes accordingly.

B.2. ECMP over paths with different MTUs

Should Equal Cost Multipath [RFC2992] be in effect between two nodes implementing this specification, with the different paths having different MTUs, then there is a high risk that probing will detect the larger of the supported MTU sizes but some data packets will flow over the path with the smaller MTU size. In this situation, packets will be lost consistently and the protocol will not be able to recover.

As such, configuring paths used for ECMP with different MTU sizes MUST be avoided.

B.3. Delay and jitter

On low-bandwidth links, the additional time it takes to transmit larger packets may lead to unacceptable delays. For instance, transmitting a 9000-byte packet takes 7.23 milliseconds at 10 Mbps, while transmitting a 1500-byte packet takes only 1.23 ms. Once transmission of a packet has started, additional traffic must wait for the transmission to finish, so a larger maximum packet size immediately leads to a higher worst-case head-of-line blocking delay, and thus, to a bigger difference between the best and worst cases (jitter). The increase in average delay depends on the number of packets that are buffered, the average packet size and the queuing strategy in use. Buffer sizes vary greatly between implementations, from only a few buffers in some switches and on low-speed interfaces in routers, to hundreds of megabytes of buffer space on 10 Gbps interfaces in some routers.

If we assume that the delays involved with 1500-byte packets on 100 Mbps Ethernet are acceptable for most, if not all, applications, then the conclusion must be that 15000-byte packets on 1 Gbps Ethernet should also be acceptable, as the delay is the same. At 10 Gbps Ethernet, much larger packet sizes could be accommodated without adverse impact on delay-sensitive applications. At below 100 Mbps, larger packet sizes are probably not advisable.

When very tight QoS bounds are required, it may be appropriate to limit MTU sizes and forego larger MTUs. With IPv6 this can be accomplished by advertising a limited MTU size in Router Advertisements. With IPv4, it is necessary to configure each node to limit its MTU size.

B.4. Path MTU Discovery problems

PMTUD issues arise when routers can't fragment packets in transit because the DF bit is set or because the packet is IPv6, but the packet is too large to be forwarded over the next link, and the resulting "packet too big" ICMP messages from the router don't make it back to the sending host. If there is a PMTUD black hole, this will typically happen when there is an MTU bottleneck somewhere in the middle of the path. If the MTU bottleneck is located at either end, the TCP MSS (maximum segment size) option makes sure that TCP packets conform to the smallest MTU in the path. PMTUD problems are of course possible with non-TCP protocols, but this is rare in practice because non-TCP protocols are generally not capable of adjusting their packet size on the fly and therefore use more conservative packet sizes which won't trigger PMTUD issues.

Taking the delay and jitter issues to heart, maximum packet sizes should be larger for faster links and smaller for slower links. This means that in the majority of cases, the MTU bottleneck will tend to be at, or close to, one of the ends of a path, rather than somewhere in the middle, as in today's internet, the core of the network is quite fast, while users usually connect to the core at lower speeds.

A crucial difference between PMTUD problems that result from MTUs smaller than the de facto standard 1500 bytes and PMTUD problems that result from MTUs larger than 1500 bytes is that in the latter case, only the party that's actually using the non-standard MTU is affected. This puts potential problems, the potential benefits and the ability to solve any resulting problems in the same place: it's always possible to revert to a 1500-byte MTU if PMTUD problems can't be resolved otherwise.

Considering the above and the work that's going on in the IETF to resolve PMTUD issues as they exist today, increasing MTUs where desired doesn't seem to involve undue risks.

B.5. Packet loss through bit errors

All transmission media are subject to bit errors. In many cases, a bit error leads to a CRC failure, after which the packet is lost. In other cases, packets are retransmitted a number of times, but if error conditions are severe, packets may still be lost because an error occurred at every try. Using larger packets means that the chance of a packet being lost due to errors increases. And when a packet is lost, more data has to be retransmitted.

Both per-packet overhead and loss through errors reduce the amount of usable data transferred. The optimum tradeoff is reached when both

types of loss are equal. If we make the simplifying assumption that the relationship between the bit error rate of a medium and the resulting number of lost packets is linear with packet size for reasonable bit error rates, the optimum packet size is computed as follows:

$$\text{packet size} = \sqrt{\text{overhead bytes} / \text{bit error rate}}$$

According to this, the optimum packet size is one or more orders of magnitude larger than what's commonly used today. For instance, the maximum BER for 1000BASE-T is 10^{-10} , which implies an optimum packet size of 312250 bytes with Ethernet framing and IP overhead.

B.6. Undetected bit errors

Nearly all link layers employ some kind of checksum to detect bit errors so that packets with errors can be discarded. In the case of Ethernet, this is a frame check sequence in the form of a 32-bit CRC. Assuming a strong frame check sequence algorithm, a 32-bit checksum suggests that there is a 1 in 2^{32} chance that a packet with one or more bit errors in it has the same checksum as the original packet, so the bit errors go undetected and data is corrupted. However, according to [CRC] the CRC-32 that's used for FDDI and Ethernet has the property that packets between 375 and 11453 bytes long (including) have a Hamming distance of 4. (Smaller packets have a larger Hamming distance, larger packets a smaller Hamming distance.) As a result, all errors where only a single bit is flipped, two bits are flipped or three bits are flipped, will be detected, because they can't result in the same CRC as the original packet. The probability of a packet having undetected bit errors can be approximated as follows for a 32-bit CRC:

$$\text{PER} = (\text{PL} * \text{BER})^H / 2^{32}$$

Where PER is the packet error rate, BER is the bit error rate, PL is the packet length in bits and H is the Hamming distance. Another consideration is the impact of packet length on a multi-packet transmission of a given size. This would be:

$$\text{TER} = \text{transmission length} / \text{PL} * \text{PER}$$

So

$$\text{TER} = \text{transmission length} / (\text{PL}^{(H-1)} * \text{BER}^H) / 2^{32}$$

Where TER is the transmission error rate.

In the case of the Ethernet FCS and a Hamming distance of 4 for a large range of packet sizes, this means that the risk of undetected errors goes up with the cube of the packet length, but goes down with the fourth power of the bit error rate. This suggests that for a given acceptable risk of undetected errors, a maximum packet size can be calculated from the expected bit error rate. It also suggests that given the low BER rates mandated for Gigabit Ethernet, packet sizes of up to 11453 bytes should be acceptable.

Additionally, unlike properties such as the packet length, the frame check sequence can be made dependent on the physical media, so in the future it should be possible to define a stronger FCS in future Ethernet standards, or to negotiate a stronger FCS between two stations on a point-to-point Ethernet link (i.e., a host and a switch or a router and a switch).

B.7. Interaction TCP congestion control

TCP performance is based on the inverse of the square of the packet loss probability. Using larger and thus fewer packets is therefore a competitive advantage. Larger packets increase burstiness, which can be problematic in some circumstances. Larger packets also allow TCP to ramp up its transmission speed faster, which is helpful on fast links, where large packets will be more common. In general, it would seem advantageous for an individual user to use larger packets, but under some circumstances, users using smaller packets may be put at a slight disadvantage.

B.8. IEEE 802.3 compatibility

According to the IEEE 802.3 standard ([IEEE.802.3_2012]), the field following the Ethernet addresses is a length field. However, [RFC0894] uses this field as a type field. Ambiguity is largely avoided by numbering type codes above 2048. The mechanisms described in this memo only apply to the standard [RFC0894] and [RFC2464] encapsulation of IPv4 and IPv6 in Ethernet, not to possible encapsulations of IPv4 or IPv6 in IEEE 802.3/IEEE 802.2 frames, so there is no change to the current use of the Ethernet length/type field.

The 2006 revision of IEEE 802.3 ([IEEE.802.3AS_2006]) adds "frame expansion" to 2000 bytes (allowing for 1982-byte IP packets). As a result, layer 2 networks supporting MTUs of 1982 bytes are becoming more common. However, as [RFC0894] and [RFC2464] (encapsulation of IPv4 and IPv6 in Ethernet) are based on [ETHERNETII]), the IEEE 802.3 standard has little bearing on the problem at hand.

B.9. Conclusion

Larger packets aren't universally desirable. The factors that factor into the decision to use larger packets include:

- o A link's bit error rate
- o The number of bits per symbol on a link and hence the likelihood of multiple bit errors in a single packet
- o The strength of the frame check sequence
- o The link speed
- o The number of buffers
- o Queuing strategy
- o Number of sessions on shared links and paths

This means that choosing a good maximum packet size is, initially at least, the responsibility of hardware builders. A conservative approach may be called for, but even under conservative assumptions, 9000-byte jumboframes on Gigabit Ethernet links seem reasonable.

Author's Address

Iljitsch van Beijnum
Institute IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

Email: iljitsch@muada.com