Stream: Internet Engineering Task Force (IETF)
RFC: 8883
Category: Standards Track
Published: September 2020
ISSN: 2070-1721
Author: T. Herbert
Intel

RFC 8883
ICMPv6 Errors for Discarding Packets Due to Processing Limits

Abstract

Network nodes may discard packets if they are unable to process protocol headers of packets due to processing constraints or limits. When such packets are dropped, the sender receives no indication, so it cannot take action to address the cause of discarded packets. This specification defines several new ICMPv6 errors that can be sent by a node that discards packets because it is unable to process the protocol headers. A node that receives such an ICMPv6 error may use the information to diagnose packet loss and may modify what it sends in future packets to avoid subsequent packet discards.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8883.

Copyright Notice

Table of Contents

1. Introduction

This document specifies several new ICMPv6 errors that can be sent when a node discards a packet due to it being unable to process the necessary protocol headers because of processing constraints or limits. New ICMPv6 code points are defined to supplement those defined in [RFC4443]. Six of the errors are specific to the processing of extension headers; another error is used when the aggregate protocol headers in a packet exceed the processing limits of a node.

1.1. Extension Header Limits

In IPv6, optional internet-layer information is carried in one or more IPv6 extension headers [RFC8200]. Extension headers are placed between the IPv6 header and the upper-layer header in a packet. The term "header chain" refers collectively to the IPv6 header, extension headers, and upper-layer headers occurring in a packet. Individual extension headers may have a maximum length of 2048 octets and must fit into a single packet. Destination Options and Hop-by-Hop Options contain a list of options in type-length-value (TLV) format. Each option includes a length of the data field in octets: the minimum size of an option (non-pad type) is two octets, and the maximum size is 257 octets. The number of options in an extension header is only limited by the length of the extension header and the Path MTU from the source to the destination. Options may be skipped over by a receiver if they are unknown and the Option Type indicates to skip (first two high order bits are 00).

Per [RFC8200], except for Hop-by-Hop Options, extension headers are not examined or processed by intermediate nodes. However, in deployed networks, many intermediate nodes do examine extension headers for various purposes. For instance, a node may examine all extension headers to locate the transport header of a packet in order to implement transport-layer filtering or to track connections to implement a stateful firewall.

Destination hosts are expected to process all extension headers and options in Hop-by-Hop and Destination Options.

Due to the variable lengths, high maximum lengths, or potential for a denial-of-service attack of extension headers, many devices impose operational limits on extension headers in packets they process. [RFC7045] discusses the requirements of intermediate nodes that discard packets because of unrecognized extension headers. [RFC8504] discusses limits that may be applied to the number of options in Hop-by-Hop Options or Destination Options extension headers. Both intermediate nodes and end hosts may apply limits to extension header processing. When a limit is exceeded, the typical behavior is to silently discard the packet.

This specification defines six Parameter Problem codes that may be sent by a node that discards a packet due to the processing limits of extension headers being exceeded. The information in these ICMPv6 errors may be used for diagnostics to determine why packets are being dropped. Additionally, a source node that receives these ICMPv6 errors may be able to modify its use of extension headers in subsequent packets sent to the destination in order to avoid further occurrences of packets being discarded.

1.2. Aggregate Header Limits

Some hardware devices implement a parsing buffer of a fixed size to process packets. The parsing buffer is expected to contain all the headers (often up to a transport-layer header for filtering) that a device needs to examine. If the aggregate length of headers in a packet exceeds the size of the parsing buffer, a device will either discard the packet or defer processing to a software slow path. In any case, no indication of a problem is sent back to the sender.

This document defines one code for ICMPv6 Destination Unreachable that is sent by a node that is unable to process the headers of a packet due to the aggregate size of the packet headers exceeding a processing limit. The information in this ICMPv6 error may be used for diagnostics to determine why packets are being dropped. Additionally, a source node that receives this ICMPv6 error may be able to modify the headers used in subsequent packets to try to avoid further occurrences of packets being discarded.

1.3. Nonconformant Packet Discard

The ICMP errors defined in this specification may be applicable to scenarios in which a node is dropping packets outside the auspices of any standard specification. For instance, an intermediate node might send a "Headers too long" code in a case where it drops a packet because it is unable to parse deeply enough to extract the transport-layer information needed for packet filtering. Such behavior might be considered nonconformant (with respect to [RFC8200], for instance).

This specification does not advocate behaviors that might be considered nonconformant. However, packet discard does occur in real deployments, and the intent of this specification is to provide visibility as to why packets are being discarded. In the spirit that providing some reason is better than a silent drop, the sending of ICMP errors is **RECOMMENDED** even in cases where a node might be discarding packets per a nonconformant behavior.

1.4. Terminology

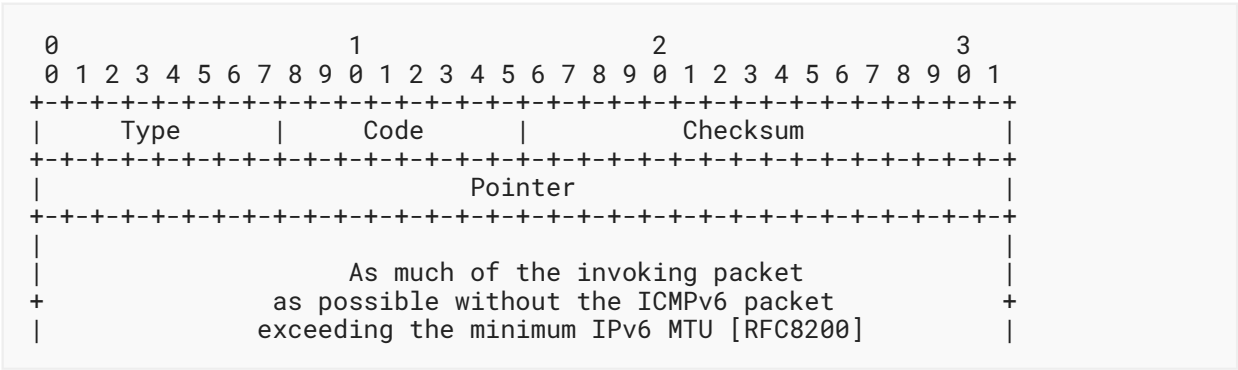
The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear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

2. ICMPv6 Errors for Extension Header Limits

Six new codes are defined for the Parameter Problem type.

2.1. Format

The format of the ICMPv6 Parameter Problem message [RFC4443] for an extension header limit exceeded error is:



IPv6 Header Fields:

Destination Address:

Copied from the Source Address field of the invoking packet.

ICMPv6 Fields:

Type:

4 (Parameter Problem type)

Code:

(pertinent to this specification)

5	Unrecognized Next Header type encountered by intermediate node
6	Extension header too big
7	Extension header chain too long
8	Too many extension headers

9	Too many options in extension header
10	Option too big

Table 1

Pointer:
Identifies the octet offset within the invoking packet where the problem occurred.

The pointer will point beyond the end of the IPv6 packet if the field exceeding the limit is beyond what can fit in the maximum size of an ICMPv6 error message. If the pointer is used as an offset to read the data in the invoking packet, then a node **MUST** first validate that the pointer value is less than the length of the invoking packet data.

2.2. Unrecognized Next Header Type Encountered by Intermediate Node (Code 5)

This code **SHOULD** be sent by an intermediate node that discards a packet because it encounters a Next Header type that is unknown in its examination. The ICMPv6 Pointer field is set to the offset of the unrecognized Next Header value within the original packet.

Note that this code is sent by intermediate nodes and **SHOULD NOT** be sent by a final destination. If a final destination node observes an unrecognized header, then it **SHOULD** send an ICMP Parameter Problem message with an ICMP Code value of 1 ("unrecognized Next Header type encountered") as specified in [RFC8200].

2.3. Extension Header Too Big (Code 6)

An ICMPv6 Parameter Problem with code for "Extension header too big" **SHOULD** be sent when a node discards a packet because the size of an extension header exceeds its processing limit. The ICMPv6 Pointer field is set to the offset of the first octet in the extension header that exceeds the limit.

2.4. Extension Header Chain Too Long (Code 7)

An ICMPv6 Parameter Problem with code for "Extension header chain too long" **SHOULD** be sent when a node discards a packet with an extension header chain that exceeds a limit on the total size in octets of the header chain. The ICMPv6 Pointer is set to the first octet beyond the limit.

2.5. Too Many Extension Headers (Code 8)

An ICMPv6 Parameter Problem with code for "Too many extension headers" **SHOULD** be sent when a node discards a packet with an extension header chain that exceeds a limit on the number of extension headers in the chain. The ICMPv6 Pointer is set to the offset of the first octet of the first extension header that is beyond the limit.

2.6. Too Many Options in Extension Header (Code 9)

An ICMPv6 Parameter Problem with code for "Too many options in extension header" **SHOULD** be sent when a node discards a packet with an extension header that has a number of options that exceeds the processing limits of the node. This code is applicable for Destination Options and Hop-by-Hop Options. The ICMPv6 Pointer field is set to the first octet of the first option that exceeds the limit.

2.7. Option Too Big (Code 10)

An ICMPv6 Parameter Problem with code for "Option too big" is sent in two different cases: when the length of an individual Hop-by-Hop or Destination Option exceeds a limit, or when the length or number of consecutive Hop-by-Hop or Destination padding options exceeds a limit. In a case where the length of an option exceeds a processing limit, the ICMPv6 Pointer field is set to the offset of the first octet of the option that exceeds the limit. In cases where the length or number of padding options exceeds a limit, the ICMPv6 Pointer field is set to the offset of the first octet of the padding option that exceeds the limit.

Possible limits related to padding include:

- The number of consecutive PAD1 options in Destination Options or Hop-by-Hop Options is limited to seven octets [RFC8504].
- The length of PADN options in Destination Options or Hop-by-Hop Options is limited seven octets [RFC8504].
- The aggregate length of a set of consecutive PAD1 or PADN options in Destination Options or Hop-by-Hop Options is limited to seven octets.

3. ICMPv6 Error for Aggregate Header Limits

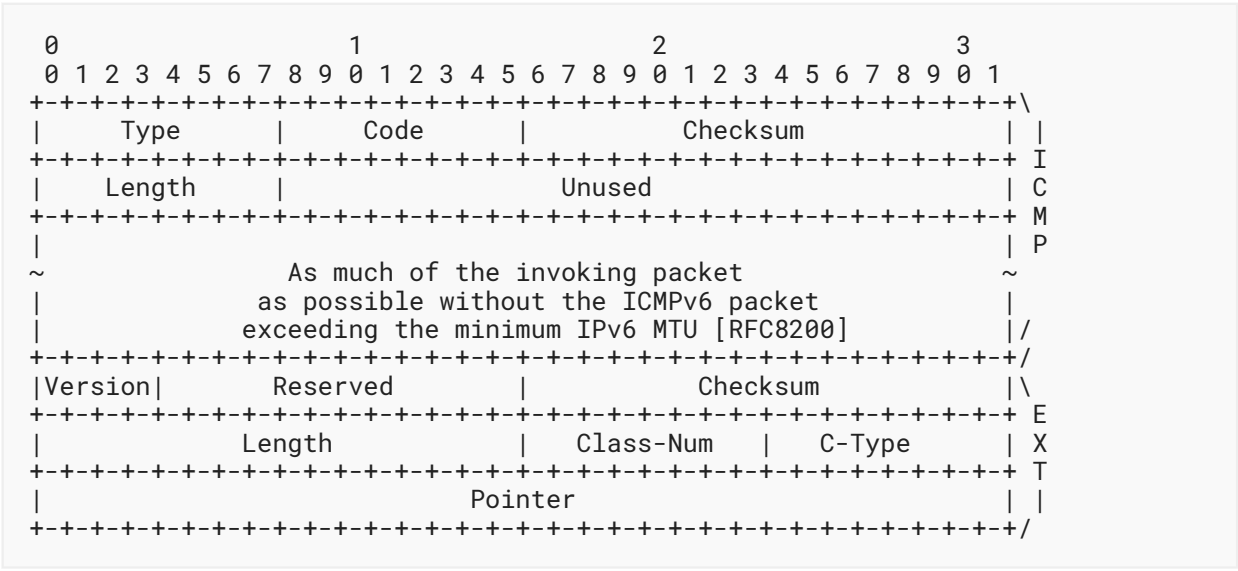
One code is defined for the Destination Unreachable type for aggregate header limits.

This ICMP error may be applied to other headers in a packet than just the IPv6 header or IPv6 extension headers. Therefore, a Destination Unreachable type with a multi-part ICMPv6 message format is used in lieu of the Parameter Problem type, which only indicates errors concerning IPv6 headers.

3.1. Format

The error for aggregate header limits employs a multi-part ICMPv6 message format as defined in [RFC4884]. The extension object class "Extended Information" is defined to contain objects for ancillary information pertaining to an ICMP Destination Unreachable error. Within this object class, the sub-type "Pointer" is defined, which contains a Pointer field with similar semantics to the Pointer field in ICMP Parameter Problem errors.

The format of the ICMPv6 message for an aggregate header limit exceeded is:



IPv6 Header Fields:

Destination Address:

Copied from the Source Address field of the invoking packet.

ICMPv6 Fields:

Type:

1 - Destination Unreachable

Code: (pertinent to this specification)

8 - Headers too long

Length:

Length of the padded invoking packet data measured in 64-bit words. The ICMP extension structure immediately follows the padded invoking packet data.

Invoking Packet:

Contains as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU. The invoking packet data **MUST** be zero padded to the nearest 64-bit boundary [RFC4884]. If the original invoking packet did not contain 128 octets, the invoking packet data **MUST** be zero padded to 128 octets.

ICMP Extension Fields:

Version:

2 - per [RFC4884]

Reserved:

0

Checksum:

The one's complement checksum of the ICMP extension [RFC4884]

Length:
8 - length of the object header and Pointer field

Class-Num:
4 - Extended Information

C-Type:
1 - Pointer

Pointer:
Identifies the octet offset within the invoking packet where a limit was exceeded.

The pointer will point beyond the end of the invoking packet data if the field exceeding the limit is beyond what can fit in the maximum size of an ICMPv6 error message with the ICMP extension. If the pointer is used as an offset to read the data in the invoking packet, then a node **MUST** first validate that the pointer value is less than the length of the invoking packet data.

3.2. Usage

An ICMPv6 Destination Unreachable error with code for "Headers too long" **SHOULD** be sent when a node discards a packet because the aggregate length of the headers in the packet exceeds the processing limits of the node. The Pointer in the extended ICMPv6 structure is set to the offset of the first octet that exceeds the limit.

This error is sent in response to a node dropping a packet because the aggregate header chain exceeds the processing limits of a node. The aggregate header chain may be composed of protocol headers other than an IPv6 header and IPv6 extension headers. For instance, in the case of a node parsing a UDP encapsulation protocol, the encapsulating UDP header would be considered to be in the aggregate header chain.

As noted in Section 4.1, the ICMPv6 Destination Unreachable error with code for "Headers too long" has the lowest precedence of the ICMP errors discussed in this specification. If a packet contains an error corresponding to a Parameter Problem code, then a node **SHOULD** send the Parameter Problem error instead of sending the ICMPv6 Destination Unreachable error with code for "Headers too long".

4. Operation

Nodes that send or receive ICMPv6 errors due to header processing limits **MUST** comply with ICMPv6 processing as specified in [RFC4443].

4.1. Priority of Reporting

More than one ICMPv6 error may be applicable to report for a packet. For instance, the number of extension headers in a packet might exceed a limit, and the aggregate length of protocol headers might also exceed a limit. Only one ICMPv6 error **SHOULD** be sent for a packet, so a priority is defined to determine which error to report.

The **RECOMMENDED** reporting priority of ICMPv6 errors for processing limits is listed from highest to lowest priority:

1. Existing ICMP errors defined in [RFC4443].
2. "Unrecognized Next Header type encountered by intermediate node"
3. "Extension header too big"
4. "Option too big" for length or number of consecutive padding options exceeding a limit.
5. "Option too big" for the length of an option exceeding a limit.
6. "Too many options in an extension header"
7. "Extension header chain too long" headers exceeding a limit.
8. "Too many extension headers"
9. "Headers too long"

4.2. Host Response

When a source host receives an ICMPv6 error for a processing limit being exceeded, it **SHOULD** verify the ICMPv6 error is valid and take appropriate action as suggested below.

The general validations for ICMP as described in [RFC4443] are applicable. The packet in the ICMP data **SHOULD** be validated to match the upper-layer process or connection that generated the original packet. Other validation checks that are specific to the upper layers may be performed and are out of the scope of this specification.

The ICMPv6 error **SHOULD** be logged with sufficient detail for debugging packet loss. The details of the error, including the addresses and the offending extension header or data, should be retained. This, for instance, would be useful for debugging when a node is misconfigured and unexpectedly discarding packets or when a new extension header is being deployed.

A host **MAY** modify its usage of protocol headers in subsequent packets to avoid repeated occurrences of the same error.

For ICMPv6 errors caused by extension header limits being exceeded:

- An error **SHOULD** be reported to an application if the application enabled extension headers for its traffic. In response, the application may terminate communications if extension headers are required, stop using extension headers in packets to the destination indicated by the ICMPv6 error, or attempt to modify its use of headers or extension headers to avoid further packet discards.
- A host system **SHOULD** take appropriate action if it is creating packets with extension headers on behalf of the application. If the offending extension header is not required for communication, the host may either stop sending it or otherwise modify its use in subsequent packets sent to the destination indicated in the ICMPv6 error.

5. Applicability and Use Cases

5.1. Reliability of ICMP

ICMP is fundamentally an unreliable protocol and, in real deployment, it may consistently fail over some paths. As with any other use of ICMP, it is assumed that the errors defined in this document are only the best effort to be delivered. No protocol should be implemented that relies on reliable delivery of ICMP messages. If necessary, alternative or additional mechanisms may be employed to augment the processes used to deduce the reason that packets are being discarded. For instance, ICMP error messages may be correlated with information attained through Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] or Happy Eyeballs for IPv6 [RFC8305]. Details of the interaction with alternative mechanisms are out of scope of this specification.

5.2. Processing Limits

This section discusses the trends and motivations of processing limits that warrant ICMP errors.

5.2.1. Long Headers and Header Chains

The trend towards longer and more complex headers and header chains needing to be processed by end nodes, as well as intermediate nodes, is driven by:

- Increasing prevalence of deep packet inspection in middleboxes. In particular, many intermediate nodes now parse network-layer encapsulation protocols or transport-layer protocols.
- Deployment of routing headers. For instance, [RFC8754] defines an extension header format that includes a list of IPv6 addresses which may consume a considerable number of bytes.
- Development of in situ OAM headers that allow a rich set of measurements to be gathered in the data path at the cost of additional header overhead, which may be significant [OAM-IPV6].
- Other emerging use cases of Hop-by-Hop and Destination Options.

5.2.2. At End Hosts

End hosts may implement limits on processing extension headers as described in [RFC8504]. Host implementations are usually software stacks that typically don't have inherent processing limitations. Limits imposed by a software stack are more likely to be for denial-of-service mitigation or performance.

5.2.3. At Intermediate Nodes

Hardware devices that process packet headers may have limits as to how many headers or bytes of headers they can process. For instance, a middlebox hardware implementation might have a parsing buffer that contains some number of bytes of packet headers to process. Parsing buffers typically have a fixed size such as 64, 128, or 256 bytes. In addition, hardware implementations

(and some software implementations) often don't have loop constructs. Processing of a TLV list might be implemented as an unrolled loop so that the number of TLVs that can be processed is limited.

6. Security Considerations

The security considerations for ICMPv6 described in [RFC4443] are applicable. The ICMP errors described in this document **MAY** be filtered by firewalls in accordance with [RFC4890].

In some circumstances, the sending of ICMP errors might conceptually be exploited as a means to covertly deduce the processing capabilities of nodes. Accordingly, an implementation **SHOULD** allow a configurable policy to withhold sending of the ICMP errors described in this specification in environments where the security of ICMP errors is a concern.

7. IANA Considerations

7.1. Parameter Problem Codes

IANA has assigned the following codes in the "Type 4 - Parameter Problem" registry within the ICMPv6 Parameters registry [IANA-ICMP]:

Code	Name
5	Unrecognized Next Header type encountered by intermediate node
6	Extension header too big
7	Extension header chain too long
8	Too many extension headers
9	Too many options in extension header
10	Option too big

Table 2

7.2. Destination Unreachable Codes

IANA has assigned the following code in the "Type 1 - Destination Unreachable" registry within the ICMPv6 Parameters registry [IANA-ICMP]:

Code	Name
8	Headers too long

Table 3

7.3. ICMP Extension Object Classes and Class Sub-types

IANA has assigned the following Class value in the "ICMP Extension Object Classes and Class Sub-types" registry within the ICMP Parameters registry [IANA-ICMPEXT]:

Class Value	Class Name
4	Extended Information

Table 4

IANA has created a sub-type registry for the "Extended Information" ICMP extension object class. The registration procedure for this registry is "Standards Action". The sub-type value of 0 is reserved; values greater than zero may be assigned.

IANA has assigned the following sub-type within the "Sub-types - Class 4 - Extended Information" registry within the ICMP Parameters registry:

Value	Description
1	Pointer

Table 5

8. References

8.1. Normative References

[IANA-ICMP] IANA, "Internet Control Message Protocol version 6 (ICMPv6) Parameters", <https://www.iana.org/assignments/icmpv6-parameters/>.

[IANA-ICMPEXT] IANA, "Internet Control Message Protocol (ICMP) Parameters", <https://www.iana.org/assignments/icmp-parameters/>.

[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>.

[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>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <https://www.rfc-editor.org/info/rfc4884>.

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

[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>.

8.2. Informative References

[OAM-IPV6] Bhandari, S., Brockners, F., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., Spiegel, M., Krishnan, S., Asati, R., and M. Smith, "In-situ OAM IPv6 Options", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-ipv6-options-03, 18 September 2020, <https://tools.ietf.org/html/draft-ietf-ippm-ioam-ipv6-options-03>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://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, <https://www.rfc-editor.org/info/rfc4890>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <https://www.rfc-editor.org/info/rfc8504>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

Acknowledgments

The author would like to thank Ron Bonica, Bob Hinden, Nick Hilliard, Michael Richardson, Mark Smith, Suresh Krishnan, and Ole Tran for their comments and suggestions that improved this document.

Author's Address

Tom Herbert
Intel
Santa Clara, CA
United States of America
Email: tom@quantonium.net