Limits on Sending and Processing IPv6 Extension Headers
draft-herbert-6man-eh-limits-00

Abstract
This specification defines various limits that may be applied to receiving, sending, and otherwise processing packets that contain IPv6 extension headers. The need for such limits is pragmatic to facilitate interoperability amongst hosts and routers in the presence of extension headers and thereby increasing the feasibility of deployment of extension headers.

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 December 24, 2021.

Copyright Notice
Copyright (c) 2021 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 ............................................. 2
   1.1. Related work ........................................ 3
   1.2. Adherence to the Robustness Principle .............. 4
       1.2.1. Be conservative in what you send ............ 4
       1.2.2. Be liberal in what you receive .............. 4
2. Overview of extension header limits ....................... 5
   2.1. Types of nodes ...................................... 5
   2.2. Types of limits ..................................... 6
       2.2.1. Limits on extension header length .......... 6
       2.2.2. Limits on option length ..................... 6
       2.2.3. Limits on number of extension headers ...... 6
       2.2.4. Limits on number of options ................. 6
       2.2.5. Limits on padding options ................. 7
       2.2.6. Limit on IPv6 header chain length ......... 8
2.3. Requirements for extension header limits .............. 11
3. Requirements .............................................. 12
   3.1. Host requirements ................................... 12
       3.1.1. Sending extension headers .................. 12
       3.1.2. Receiving extension headers ............... 13
   3.2. Intermediate node and intermediate destination
       requirements ............................................. 15
   3.3. Intermediate destination requirements ............. 16
4. References ............................................... 17
   4.1. Normative References ................................. 17
   4.2. Informative References .............................. 18
Author’s Address ......................................... 18

1. Introduction

Extension headers are a core component of the IPv6 protocol as specified in [RFC8200]. IPv6 extension headers were originally defined with few restrictions. For instance, there is no specified limit on the number of extension headers a packet may have, nor is there a limit on the length in bytes of extension headers in a packet (other than being limited by the MTU). Similarly, variable length extension headers typically do not have prescribed limits such as limits on the number of Hop-by-Hop or Destination options in a packet. The lack of limits essentially requires implementations to handle every conceivable usage of the protocol, including a myriad of use cases that are obviously outside the realm of ever being realistic or useful in real world deployment.
The lack of limits and the requirements for supporting virtually open-ended protocol have led to a significant lack of support and deployment of extension headers [RFC7872]. Instead of attempting to satisfy the protocol requirements concerning extension headers, some router and middlebox vendors have opted to either invent and apply their own ad hoc limits, relegate packets with extension headers to slow path processing, or have gone so far as to summarily discard all packets with extension headers. The net result of this situation is that deployment and use of extension headers is underwhelming to the extent that they are often considered unusable, and hence IPv6 extension headers have not lived up to their potential as the extensibility mechanism of IPv6.

As an example, consider that Hop-by-Hop Options and Destination Options have no limit on how many options may be placed in a packet nor any limits as to how many options a receiver must process. A single 1500 byte MTU sized packet could legally contain a Hop-by-Hop Options extension header with over seven hundred two byte options. There is no use case for this other than being a Denial of Service attack where an attacker simply creates packets with hundreds of small unknown Hop-by-Hop Options with the two high order bits in the option type set to 00 meaning to skip the unknown option. Any node in that path that attempts to dutifully process all these options per the requirements of [RFC8200] would be easily overwhelmed by the processing needed to parse these options (this is true for both hardware or software implementations).

This specification describes various limits that hosts and intermediate nodes may apply to the processing of extension headers. The goal of establishing limits is to narrow the requirements to better match reasonable use cases thereby facilitating practical implementation. Subsequently, this increases the viability of extension headers as the extensibility mechanism of IPv6.

1.1. Related work

Some of the problems of unlimited extension headers have been addressed in certain aspects.

[RFC8200] relaxed the requirement that all nodes in the path must process Hop-by-Hop Options to be:

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.
Section 5.3 of [RFC8504] defines a number of limits that hosts may apply to processing extensions. For instance:

A host MAY set a limit on the maximum number of non-padding options allowed in the destination options and Hop-by-Hop extension headers. If this feature is supported, the maximum number SHOULD be configurable, and the default value SHOULD be set to 8.

[RFC8883] defines a set of ICMP errors that may be sent if a limit concerning extension headers is exceeded and a node discards a packet as a result. This RFC allows both hosts and routers to send such messages (effectively acknowledging that some routers drop packets with extension headers even though such behavior is non-conformant).

[RFC7872] presents real-world data regarding the extent to which packets with IPv6 Extension Headers (EHs) are dropped in the Internet, and [I-D.gont-v6ops-ipv6-ehs-packet-drops] summarizes the operational implications of IPv6 extension headers, and attempts to analyze reasons why packets with IPv6 extension headers are often dropped in the public Internet.

1.2. Adherence to the Robustness Principle

The robustness principle, or Postel’s Law, can be stated as "Be conservative in what you send, liberal in what you receive". This section considers the limits defined in this specification with respect to the robustness principle.

1.2.1. Be conservative in what you send

The limits on sending extension headers are well aligned with the send clause of the robustness principle. A sender of extension headers is generally constrained in its use of extension headers. Most of these limits are assumed to be the default to apply in an arbitrary environment such as the public Internet, that is they can be considered "baseline limits". These limits may be relaxed if a sender has a priori information that all possible nodes in path will properly handle packets that exceed the baseline limits. In particular, if a sender is sending in a limited domain, it might be known that all nodes in the limited domain have sufficient capabilities to handle packets exceeding the baseline limits.

1.2.2. Be liberal in what you receive

Considering the receive clause of the robustness principle, this specification recommends that receivers accept all packets with extension headers, however they may ignore extension headers or
options within extension headers. In particular, the philosophy of this specification is that intermediate nodes should not drop packets on the basis that they don’t have sufficient capabilities to process all the headers in a packet. As such, intermediate nodes may define arbitrarily restrictive limits on what they process with regards to extension headers as long as the action taken when those limits are exceeded is to ignore items beyond the limit. Hosts are more constrained in this regard since they generally can’t correctly process a packet without processing all the headers, so when limits are exceeded on a host, packets are dropped. It should be noted that hosts stacks inherently have more processing capabilities than intermediate nodes, so it is expected that they should be able to support higher limits.

This specification does specify one hard requirement for receiving nodes, namely nodes must be able to properly handle packets having an IPv6 header chain length up to 104 bytes. This requirement acknowledges that some intermediate nodes perform deep packet inspection at least to extract information from the transport layer headers. In this case, the data exceeding the limit may contain information that the node considers critical for correct processing, so that data cannot be ignored.

2. Overview of extension header limits

This specification considers extension header limits in three dimensions: 1) The types of nodes that may process extension headers and the requirements specific to each type, 2) The types of limits that may be applied, 3) The action taken when a limit is exceeded.

2.1. Types of nodes

For the purposes of describing handling of extension headers this specification considers three types of node in an IPv6 network:

* Hosts: The source of an IPv6 packet, as addressed by the source address, or the final destination node of a packet as addressed by the destination address in a packet with no Routing header or as addressed by final segment in a Routing header.

* Intermediate destination: An intermediate destination node in a Routing header as addressed by the destination address of a packet with a Routing header where the address is not the final destination in the Routing header

* Intermediate nodes: A router on the path that is not addressed by the packet’s destination address.
2.2. Types of limits

The limits and requirements for handling extension headers defined in this specification fall in the following categories:

* Limits on extension header length
* Limits on option length
* Limits on number of extension headers
* Limits on number of options
* Limits on padding for extension headers with options
* Limits on the length of the IPv6 header chain

2.2.1. Limits on extension header length

[RFC8504] defines limits that may be defined for the length of an extension header. Those limits are extended to be applicable to intermediate nodes. [RFC8883] defines ICMP Parameter Problem codes that may be sent when an extension header is exceeded.

2.2.2. Limits on option length

A node may establish a limit on the size Hop-by-Hop or Destination options. Conceivably, such a limit could apply to all option types, or length limits may be specific to individual options. [RFC8883] defines ICMP Parameter Problem codes that may be sent when an option length limit is exceeded.

2.2.3. Limits on number of extension headers

A node may define a limit on the number of extension headers it will process. Although [RFC8200] only defines four types of extension headers, it does not preclude the same type of extension header being present multiple times. A limit on the number of extension headers could be useful to disallow packets that contain multiple instances of the same extension header.

2.2.4. Limits on number of options

Limits may be established for the number of options sent or received (specifically applicable to Hop-by-Hop options and Destination options). The need for this limit arises from the fact that [RFC8200] does not specify a limit. Requiring nodes to process packets with tens or hundreds of options has no foreseeable use cases
in deployment except as a denial of service attack. [RFC8504] has proposed such a limit for host processing of Hop-by-Hop and Destination options with a default of eight options. This specification extends that limit to be applicable to intermediate nodes. Specific limits may be established for number of non-padding options or the number of all options including padding.

To derive a limit on all options, one can assume that at most one padding option is used between two non-padding options (an explicit limit on consecutive padding options is described below). With this assumption, we can extrapolate a reasonable limit on the number of all options that should be twice the limit of the number of non-padding options. Per [RFC8504], the recommended default limit for number of non-padding options is eight, so this specification establishes a maximum default limit of sixteen options including padding options. The choice of sixteen options as a default limit attempts to strike a balance between allowing extensibility and maintaining reasonable expectations for node processing requirements.

With regards to extensibility, it is observed that in the almost thirty year history of IPv6 there are only thirteen defined non-deprecated Destination options and Hop-by-Hop options and three temporary assigned options. Current evidence suggests that having more than one Destination option or Hop-by-Hop option in a packet is rare, and extrapolating that point with the rate of new options being defined suggests a limit of eight non-padding options allows for sufficient extensibility in the foreseeable future.

With regards to processing requirements, TLVs, e.g. Hop-by-Hop options and Destination options, have historically been considered difficult to process efficiently due to their serial processing requirements and combinatorial nature. TLV processing has been a particularly acute problem for ASIC devices. Recently, there is a strong trend in programmable implementation even in high performance routers using emerging programming frameworks such as PANDA and P4. Programmable implementations are better equipped to handle TLVs, at least for a reasonably small number. It might also be pointed that the need to efficiently process TLVs exists in other protocols, for instance processing TCP requires processing of TLVs which are an intrinsic part of the protocol.

2.2.5. Limits on padding options

[RFC8200] defines PAD1 and PADN options that respectively provide one byte or N bytes of padding in an extension header. The purpose of padding is to properly align the following non-padding option to its expected alignment, or to add padding after the last Destination or Hop-by-Hop option so that the length of the extension header is a
multiple of eight bytes as required by [RFC8200]. [RFC8504] defines limits on number of bytes used for consecutive padding where the amount of padding between options or at the end of the extension header is no more than eight bytes; this limit is sufficient to align any following data after the padding to eight bytes. These limits are extended to be applicable to intermediate nodes.

This specification allows a receiving node to set a requirement that consecutive padding options are not present in a packet; which in turn requires a sender to not place consecutive padding options in a packet. The rationale for this limit is that a PAD1 or PADN option is able to provide one to 257 bytes of padding, so a single padding option is sufficient for any expected use case of padding. When the sender creates options, it can compute the amount of padding necessary to satisfy the alignment requirements of the following data. If one byte of padding is needed a PAD1 option is used, if more than one byte of padding is needed then an appropriate PADN options.

2.2.6. Limit on IPv6 header chain length

Intermediate nodes often perform deep packet inspection (DPI) in order to implement various functions in the network. Routers perform DPI when they inspect packets beyond the IPv6 header or beyond Hop-by-Hop options if they are present. Some router implementations must inspect the transport layer headers in order to process and forward the packet, and if the transport layer headers are not readable a packet may be dropped. Even if a transport layer header is in plain text within a packet, some devices may not be capable of reading it if the header is too deep in the packet.

Hardware devices often have constraints on how much of the headers in a packet can be parsed for DPI. A typical design is that some portion of the beginning of a received packet is loaded into a memory buffer for header parsing (i.e. the parsing buffer). The size of this parsing buffer is often fixed per device.

To derive a size limit on the IPv6 header chain, we need to take into account headers in a packet that might be subject to DPI which include the link layer header through at least the pertinent fields of the transport layer header. The most common required information is the transport layer port numbers which typically occupy the first four bytes of the transport headers (e.g. TCP, UDP, SCTP, DCCP, etc.). Inspection of port numbers may be needed for stateless load balancing as well as port filtering. There are middleboxes that may need to inspect more of transport layer headers or the transport payload, however those can be considered specialized devices that
perform work beyond simple packet forwarding and filtering and hence should have more capabilities for DPI.

In addition to limits on the length of the IP header chain, it is conceivable that there could be a limit on the length of the whole header chain. The whole header chain would comprise the IPv6 header chain as well as any headers that are part of network encapsulation that precede the innermost transport layer. The definition of such a limit is out of scope for this document, however [RFC8883] defines an ICMP error to send when a limit on size of an aggregate header chain is exceeded.

This document specifies that the minimum supported limit for IPv6 header chains is 104 bytes. The value is derived by assuming that nodes have the ability to process at least the first 128 bytes of a packet (that is they have a parsing buffer that can contain at least 128 bytes). The 128 byte parsing buffer would be expected to at least contain:

* 16 bytes for a Layer 2 header (e.g. Ethernet header)
* 40 bytes for the IPv6 header
* 64 bytes for the extension headers
* 8 bytes for the transport layer (i.e the first eight bytes of the transport layer header

This scheme thus establishes a requirement that all Internet devices are capable of correctly processing packets with up to sixty-four bytes of extension headers, and subsequently it establishes a requirement that a host shouldn’t send packets with more than sixty-four bytes of extension headers. Note that this establishes a global baseline requirement across the Internet, within a limited domain higher limits could be applied.

128 bytes is likely the minimal useful parsing buffer size in deployment today. Devices performing a very narrow DPI could conceptually use a smaller parsing buffer, for instance that could be as small as sixty-four bytes which accommodates an L2 header, IPv6 header, and eight bytes of transport header; however, such a device would be extremely limited in capabilities and if they do exist they are likely legacy devices that will eventually be decommissioned. Many routers now have the capability to perform DPI into encapsulation headers which implies they already have a larger parsing buffer than this baseline minimum.
Similar to limiting the number of options allowing in a packet, setting a limit for IP header length chain is a tradeoff between extensibility and feasible implementation.

For extensibility, the pertinent extension headers contributing to the sixty-four byte limit are mostly Hop-by-Hop and Destination options. The Routing Header extension header is really intended for limited domains (e.g., SRv6 Routing Header is confined to a Segment Routing Domain) and therefore would be subject to a domain specific limit for IP header chain length. Encryption Header may be used on the Internet, however encryption obfuscates the encapsulated transport headers such that intermediate nodes can’t inspect them regardless of their position in a packet. Fragmentation may be used in the Internet, however only the first fragment of a fragmented packet might contain transport layer headers that could be read by an Intermediate node. In any case, the Fragment Header is only four bytes so that would not be a particularly large portion of a sixty-four byte limit.

The Authentication Header is usable on the Internet and does allow the transport layer headers to be in readable in plain text. However, Authentication Header is relatively large, typically thirty-two bytes or more, so it would contribute significantly to a limit on IP header chain length. On the other hand, the use of Authentication Header, without encryption, is currently rare on the Internet.

Individual Hop-by-Hop Destination Options may also be categorized as being intended for use over the Internet or just in limited domains. For instance, the IOAM Hop-by-Hop option is intended for use in limited domains.

Paring this down, the types extension headers and Destination and Hop-by-Hop options that might be used outside of limited domains are fairly limited. Options that are intended for use over the public Internet could be defined to be small and compact to promote not exceeding a sixty-four byte limit on extension headers, whereas options constrained to a limited domain could be larger since larger limits can be assumed.

2.2.6.1. Action when limit is exceeded

For each limit that is defined, an action is specified for when the limit is exceeded. The appropriate action depends on whether the processing node is the destination host, an intermediate destination, or an intermediate node. For a destination host, the typical action to take when a limit is exceeded is to discard the packet. This is appropriate since the destination host is required to process all of
the headers in a packet, and if a limit is exceeded then it cannot process the packet so there is no other alternative but to discard.

For intermediate nodes, the typical action to take when a limit is exceeded is to stop processing headers at the point the limit is reached and to forward the packet on. [RFC8200] allows that an intermediate may not process the Hop-by-Hop Options extension headers therefore an intermediate node may ignore all of the Hop-by-Hop options in a packet. This specification expands on that requirement to allow an intermediate node to process some arbitrary subset of consecutive Hop-by-Hop options in the TLV list and to ignore the following ones. In the case of an egregious violation of a limit, for instance an attacker sends three hundred options in a packet, the destination host can decide if the appropriate response is to drop (the destination host must process all options). Note that this provision motivates the sender to place Hop-by-Hop Options in the packet so that those considered more important are placed first. It should also be noted that [RFC8200] sets a default limit of eight; this specification adds a counterpart for sending hosts that they shouldn’t send more than eight Hop-by-Hop options.

Intermediate destinations have characteristics of both hosts and intermediate modes. If a limit is exceeded related to Hop-by-Hop options then the suggested action in this specification is to assume the same processing of limits as intermediate nodes. If limits are exceeded that affect the processing specific to an intermediate destination, such as limits on Destination options before the Routing header, then the action should be to discard packet.

2.3. Requirements for extension header limits

The set of limits that a node may apply when processing extension headers include:

* Too many non-padding or padding options
* Extension header too big
* Option too big
* Too many consecutive padding options
* Too many consecutive bytes of padding
* Extension header chain too long
* Aggregate header chain too long
* Too many extension headers

3. Requirements

This section lists the normative requirements related to sending and processing extension headers.

3.1. Host requirements

3.1.1. Sending extension headers

The requirements are:

* A host MUST NOT send more than 8 non-padding options in Destination Options in a packet unless it has explicit knowledge that the destination, or all intermediate destinations in the case of Destination Options before the routing header, are able to process a greater number of options.

* A host MUST NOT send more than 8 non-padding options in Hop-by-Hop Options in a packet unless it has explicit knowledge that the final destination host is able to process a greater number of options.

* A host SHOULD NOT send more than 8 non-padding options in Hop-by-Hop Options in a packet unless it has explicit knowledge that all possible intermediate nodes are able to process a greater number of options or will ignore options that exceeds their limit.

* A host MUST NOT send a packet with an extension header larger than 64 bytes unless it has explicit knowledge that all nodes that might process the extension header are capable of processing a larger header.

* A host MUST NOT send a packet with a Destination option or Hop-by-Hop option with Data Length greater than 60 bytes unless it has explicit knowledge that all nodes that might process the option are capable of processing ones with a larger Data Length.

* A host node MUST NOT send a packet with an IPv6 header chain larger than 104 bytes unless it has explicit knowledge that all nodes in the path are capable of properly handling packets with larger header chains. This requirements is equivalently stated as a host MUST NOT send a packet with more than 64 bytes of aggregate extension headers.

* A host MUST NOT set more than one consecutive pad option, either PAD1 or PADN, in Destination options or Hop-by-Hop options.
A host MUST NOT send a PadN option in Hop-by-Hop Options or Destination Options with total length of more than seven bytes.

A host node MUST NOT send more than 16 options (padding or non-padding) Destination options in a packet unless it has explicit knowledge that the destination, or all intermediate destinations in the case of Destination Options before the routing header, are able to process a greater number of options. Note that if the above requirements on a host sending non-padding Destination options and requirements on option padding are met, then this requirement is implicitly satisfied.

A host node MUST NOT send more than 16 options (padding or non-padding) in Hop-by-Hop Options in a packet unless it has explicit knowledge that the final destination host is able to process a greater number of options. Note that if the above requirements on a host sending non-padding Hop-by-Hop options and requirements on padding are met, then this requirement is implicitly satisfied.

3.1.2. Receiving extension headers

Per [RFC8200], a host node that receives a packet with extension headers must process all the extension headers in the packet before accepting the payload and processing the payload.

As described in [RFC8504] a host may establish limits on the processing of extension headers. This specification reiterates and updates those requirements to allow for a host to send an RFC8883 error if a limit has been exceeded.

A host MAY set a limit on the maximum number of non-padding options allowed in the Destination Options or Hop-by-Hop Options extension headers. If this limit is supported then the maximum number SHOULD be configurable, the limit MUST be greater than or equal to 8, and the default value SHOULD be set to 8. The limits for Destination options and Hop-by-Hop options MAY be separately configurable. If a packet is received and the number of Destination or Hop-by-Hop options exceeds the limit, then the packet SHOULD be discarded and an ICMP Parameter Problem with code 9 MAY be sent to the packet’s source address.

A host MAY set a limit on the maximum number of options (padding or non-padding) allowed in Destination Options or Hop-by-Hop Options extension headers. If this limit is supported then the maximum number SHOULD be configurable and the limit MUST be greater than or equal to 16. The limits for Destination options and Hop-by-Hop options MAY be separately configurable. If a packet is received and the number of destination or Hop-by-Hop
options exceeds the limit, then the packet SHOULD be discarded and
and an ICMP Parameter Problem with code 9 MAY be sent to the
packet’s source address

* A host node MAY set a limit on the length of an extension header.
If this limit is supported then the limit SHOULD be configurable
and the limit MUST be greater than or equal to 64 bytes. The
length limits for different extension headers MAY be separately
configurable.

* A host node MAY set a limit on the Data Length of a Hop-by-Hop or
Destination option. If this limit is supported then the limit
SHOULD be configurable, and the limit MUST be greater than or
equal to 60 bytes. The limits for Destination options and Hop-by-
Hop options MAY be separately configurable. If a packet is
received and a Hop-by-Hop or destination option has a length that
exceeds the limit, then the packet SHOULD be discarded and an ICMP
Parameter Problem with code 10 MAY be sent to the packet’s source
address.

* A host MAY limit the number of consecutive PAD1 options in
destination options or Hop-by-Hop options to 7. In this case, if
there are more than 7 consecutive PAD1 options present, the packet
SHOULD be discarded and an ICMP Parameter Problem with code 10 MAY
be sent to the packet’s source address

* A host MAY limit the number of bytes in a PADN option to be less
than 8. In such a case, if a PADN option is present that has a
length greater than 7, the packet SHOULD be discarded and an ICMP
Parameter Problem with code 10 MAY be sent to the packet’s source
address.

* A host MAY set a limit on the maximum length of Destination
Options or Hop-by-Hop Options extension headers. This value
SHOULD be configurable, and if the limit is used then the limit
MUST be greater than or equal to 64 bytes. If a packet is
received and the length of the Destination or Hop-by-Hop Options
extension header exceeds the length limit, then the packet SHOULD
be discarded and an ICMP Parameter Problem with code 6 MAY be sent
to the packet’s source address.

* A host node MAY set a limit on the maximum length of the IPv6
header chain, or equivalently a host MAY set a limit on the
aggregate length of extension headers in a packet. If the limit
is used then it MUST be greater than or equal to 104 bytes, or,
equivalently, the limit on aggregate header extension length MUST
be greater than or equal to 64 bytes. If a packet is received and
the aggregate length of the IPv6 header chain exceeds the limit
then the packet SHOULD be discarded and an ICMP Parameter Problem with code 7 MAY be sent to the packet’s source address.

Additional host requirements for receive.

* A host MAY disallow consecutive padding options, either PAD1 or PADN, to be present in a packet. If consecutive padding options are received and disallowed by the host, the then packet SHOULD be discarded and an ICMP Parameter Problem with code 9 MAY be sent to the packet’s source address.

3.2. Intermediate node and intermediate destination requirements

The following requirements are established for intermediate nodes and intermediate destination nodes that receive and process packets with extension header.

* An intermediate node MUST be able to correctly forward packets that contain an IPv6 header chain of 104 or fewer bytes, or equivalently an intermediate node MUST be able to process a packet with an aggregate length of extension headers less than or equal to 64 bytes.

* Per [RFC8200] an intermediate node MAY be configured to not process Hop-by-Hop Options. If a node is configured as such and a packet with Hop-by-Hop options is received, the extension header MUST be be skipped and the packet MUST otherwise be properly processed and forwarded.

* An intermediate node MAY limit the number of non-padding Hop-by-Hop options that it processes. If a limit is exceeded, that is a packet contains more non-padding options than are configured to process, the intermediate SHOULD stop processing the Hop-by-Hop Option and ignore any options in the chain beyond the limit. It is NOT RECOMMENDED that an intermediate node discards the packet because the limit is exceeded, however if it does so then the intermediate node MAY send an ICMP Parameter Problem with code 10 MAY be sent to the packet's source address.

* An intermediate node MAY limit the number of Hop-by-Hop options (padding or non-padding) that it processes. If a limit is exceeded, that is a packet contains more non-padding options than are configured to process, the intermediate SHOULD stop processing the Hop-by-Hop options and ignore any options in the chain beyond the limit. It is NOT RECOMMENDED that the intermediate node discards the packet because the limit is exceeded, however if it does so then the intermediate node MAY send an ICMP Parameter Problem with code 10 MAY be sent to the packet’s source address.
* If an intermediate node encounters an unknown Hop-by-Hop option and the two high order bits are not 00 then the node SHOULD immediately stop processing the option chain and ignore any options in the chain beyond the unknown option. An intermediate node MAY either elect to discard the packet and MAY send an ICMP Parameter Problem per the requirements of [RFC8200]; or the intermediate node MAY forward the packet.

* An intermediate node MAY set a limit on the maximum length of Hop-by-Hop Options extension headers. This value SHOULD be configurable. If this limit is exceeded, that is a packet has an extension header larger then the limit, then the intermediate SHOULD stop processing the Hop-by-Hop Option and ignore any options in the chain beyond the limit. It is NOT RECOMMENDED that the intermediate node discards the packet because the limit is exceeded, however if it does so then the intermediate node MAY send an ICMP Parameter Problem with code 10 MAY be sent to the packet’s source address.

3.3. Intermediate destination requirements

The following are requirements specific to intermediate destinations pertaining to the processing of Destination Options before the Routing header.

* An intermediate destination MAY set a limit on the maximum length of Destination Options extension header before the Routing header. This value SHOULD be configurable, and the default is to accept options of any length. If a limit is defined is MUST be at least 64 bytes. If the limit is exceeded then the intermediate destination SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 6 to the packet’s source address.

* An intermediate destination node MAY limit the number of non-padding options in Destination Options before the Routing header. If a limit is exceeded, that is a packet contains more non-padding options than are configured to process, the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.

* An intermediate destination node MAY limit the number of options (padding or non-padding) in Destination Options before the Routing header. If a limit is exceeded, that is a packet contains more non-padding options than are configured to process, the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.
* An intermediate destination MAY limit the total number bytes in consecutive PAD1 options in destination options before the Routing Header 7. If the limit is exceeded, that is there are more than seven bytes in consecutive PAD1 or PADN options present, the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.

* A intermediate destination MAY limit the number of bytes in a PADN option in Destination Options before the Routing header to be less than 8. In such a case, if a PADN option is present that has a length greater than 7, the packet SHOULD be discarded and the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.

* A intermediate destination MAY set a limit on the maximum number of non-padding options allowed in Destination options before the Routing header. If this feature is supported, the maximum number SHOULD be configurable, and the default value SHOULD be set to 8. If a packet is received and the number of Destination options before the Routing header exceeds the limit, the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.

* A intermediate MAY set a limit on the maximum length of Destination Options extension header before the Routing header. This value SHOULD be configurable, and the default is to accept options of any length. If a packet is received and the length of the Destination or Hop-by-Hop Options extension header exceeds the length limit, the intermediate destination node SHOULD discard the packet and MAY send an ICMP Parameter Problem with code 10 to the packet’s source address.

4. References

4.1. Normative References


4.2. Informative References

[I-D.gont-v6ops-ipv6-ehs-packet-drops]


Author’s Address

Tom Herbert
SiPanda
Santa Clara, CA
USA

Email: tom@sipanda.io
IPv6 Hop-by-Hop Options Processing Procedures
draft-hinden-6man-hbh-processing-01

Abstract

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet. When published, this document updates RFC8200.

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 4 December 2021.

Copyright Notice

Copyright (c) 2021 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
1. Introduction

This document specifies procedures for how IPv6 Hop-by-Hop options are processed. It modifies the procedures specified in the IPv6 Protocol Specification (RFC8200) to make processing of IPv6 Hop-by-Hop options practical with the goal of making IPv6 Hop-by-Hop options useful to deploy and use in the Internet.

When published this document updates [RFC8200].

The current list of defined Hop-by-Hop options can be found at [IANA-HBH].

2. 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.

3. Terminology

This document uses the following loosely defined terms:
* Forwarding Plane: IPv6 hosts exchange user data through the forwarding plane. User data is processed by its recipient (i.e., an IPv6 host). User data can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads.

* Control Plane: IPv6 routers exchange management and routing information with controllers. They also exchange routing information with one another. Management and routing information is processed by its recipient (i.e., an IPv6 router or controller). Management and control information can traverse intermediate nodes (i.e., routers) between its source and its destination. These intermediate nodes process metadata contained in packet headers. However, they do not process information contained in packet payloads. So, from their perspective, this information is user data.

* Fast Path: A path through a router that is optimized for forwarding packets without processing their payloads. The Fast Path may be supported by Application Specific Integrated Circuits (ASICS), Network Processor (NP), or other special purpose hardware. This is the usual processing path within a router taken by the forwarding plane.

* Slow Path: A path through a router that is capable of general purpose processing and is not optimized for any particular function. This is processing path is used for packets that require special processing or differ from assumptions made in Fast Path heuristics, or to process router control protocols used by the control plane.

NOTE: This distinct separation between hardware and software processing from [RFC6398] does not apply to all router architectures. However, a router that performs all or most processing in software might still incur more processing cost when providing special processing (aka Slow Path).

[RFC6192] is an example of how designs can separate control plane (Slow Path) and forwarding plane (Fast Path) functions.

4. Background

In the first version of the IPv6 specification, Hop-by-Hop options were required to be processed by all nodes: routers and hosts. This proved to not be practical in high speed routers due to several factors, including:
* Inability to process the hop-by-hop options at wire speed on the Fast Path.

* Hop-by-Hop options would be sent to the Slow Path. This could degrade the router's performance and its ability to process important control traffic.

* A mechanism that forces packets from any source to the router's "Slow Path" could be exploited as a Denial of Service attack against the router.

* Packets could contain multiple Hop-by-Hop options making the previous issues worse by increasing the complexity required to process them.

When the IPv6 Specification was updated and published in July 2017 as [RFC8200], the procedures relating to hop-by-hop options were as follows:

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.

The changes meant that an implementation complied with the IPv6 specification even if it did not process hop-by-hop options, and that it was expected that routers would add configuration information to control which hop-by-hop options they would process.

Unfortunately, this did not improve the processing of Hop-by-Hop options and did not significantly improve deployment and use in the Internet. Essentially, it only documented how they were being used in the Internet at the time RFC8200 was published.
The main issues remain:

* Routers are commonly configured to drop transit packets containing hop-by-hop options that would have been processed in the Slow Path. This behavior is seen as protecting against a denial of service attack on the router. This is discussed in [I-D.ietf-v6ops-ipv6-ehs-packet-drops].

* Allowing multiple hop-by-hop options in a single packet makes it even more expensive in router resources to process these packets. It adds complexity to the number of permutations that might need to be processed.

* Any mechanism that can be used to force packets into the router’s Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources needed for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. This issue for the Router Alert option, which intentionally places packets on the Slow Path, is discussed in [RFC6398]. Section 3 of that RFC includes a good summary:

  "In a nutshell, the IP Router Alert Option does not provide a convenient universal mechanism to accurately and reliably distinguish between IP Router Alert packets of interest and unwanted IP Router Alert packets. This, in turn, creates a security concern when the IP Router Alert Option is used, because, short of appropriate router-implementation-specific mechanisms, the router Slow Path is at risk of being flooded by unwanted traffic."

There has been research that discussed the general problem with dropping packets containing IPv6 extension headers, including the Hop-by-Hop Options header. For example [Hendriks] states that "dropping all packets with Extension Headers, is a bad practice", and that "The share of traffic containing more than one EH however, is very small. For the design of hardware able to handle the dynamic nature of EHs, we therefore recommend to support at least one EH".

This document defines a set of procedures for the hop-by-hop option header that make the processing of hop-by-hop options practical in modern transit routers.

5. Hop-by-Hop Header Processing Procedures

   This section describes several changes to [RFC8200].
5.1. Hop-by-Hop Options Per Packet

The Hop-by-Hop Option Header as defined in Section 4.3 of [RFC8200] is identified by a Next Header value of 0 in the IPv6 header. Section 4.1 of [RFC8200] requires a Hop-by-Hop Options header to appear immediately after the IPv6 header. [RFC8200] also requires that a Hop-by-Hop Options header can only appear once in a packet.

The Hop-by-Hop Options Header as defined in [RFC8200] can contain one or more Hop-by-Hop options. This document updates [RFC8200] that a node MUST process the first Option in the Hop-by-Hop Header in the Fast Path and MAY process additional Hop-by-Hop Options if configured to do so. The motivation for this change is to simplify the processing of Hop-by-Hop options in the Fast Path.

Nodes creating packets with a Hop-by-Hop option headers SHOULD include a single Hop-by-Hop Option in the packet and MAY include more based on local configuration.

If there are more than one Hop-by-Hop options in the Hop-by-Hop Options header, the node MAY skip the rest of the options without having to examine these options using the "Hdr Ext Len" field in the Hop-by-Hop Options header. This field specifies the length of the Option Header in 8-octet units. The additional options do not need to be processed or verified.

5.2. Hop-by-Hop Headers Processing

Nodes that implement a differentiation between a Fast Path and a Slow Path MUST process all (with one exception noted below) Hop-by-Hop options in the Fast Path. The one exception to this is the Router Alert Option [RFC2711]. See Section 5.3 for discussion of the Router Alert.

If the node can not process an option in the Fast Path, it MUST behave as if it does not recognize the Option Type (as described in the next paragraph).

Section 4.2 of [RFC8200] defines the Option Type identifiers as internally encoded such that their highest-order 2 bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type. The text is:
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.

This document modifies this behaviour for the "10" and "11" values that the node MAY send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address, pointing to the unrecognized Option Type. The modified text for "10" and 11" values is:

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

The motivation for this change is to loosen the requirement to send ICMPv6 Parameter Problem messages to simplify what the router needs to do in the Fast Path when it does not recognize the Option Type.

When an ICMP Parameter Problem, Code 2, message is delivered to the source, the source can become aware that at least one node on the path has failed to recognize the option.

5.3. Router Alert Option

The Router Alert option [RFC2711] purpose is to tell the node that the packet needs additional processing on the Slow Path.

The Router Alert option includes a two octet Value field that describes the protocol that is carried in the packet. The current values can be found in the IANA Router Alert Value registry [IANA-RA].
DISCUSSION

The Router Alert Option is a problem since it’s function is to do what this specification is proposing to eliminate, that is, process the packet in the Slow Path. One approach would be to deprecate it as it’s usage appears to be limited and packets containing Hop-by-Hop options are frequently dropped. Deprecation would allow current implementations to continue and it’s use could be phased out over time.

The authors current thinking is that the Router Alert function may have reasonable potential use for new functions that have to be processed in the Slow Path. We think that keeping it as the single exception for Slow Path processing with the following restrictions is a reasonable compromise to allow future flexibility. These are compatible with Section 5 of [RFC6398].

A Fast Path implementation SHOULD verify that a Router Alert contains a protocol, as indicated by the Value field in the Router Alert option, that is configured as a protocol of interest to that router. A verified packet SHOULD be sent on the Slow Path for processing [RFC6398]. Otherwise, the router implementation SHOULD forward within the Fast Path (subject to all normal policies and forwarding rules). As specified in [RFC2711] the top two bits of Option Type for the Router Alert option are always set to "00" indicating the node should skip over this option and continue processing the header in this case.

Implementations of the IP Router Alert Option SHOULD offer the configuration option to simply ignore the presence of "IP Router Alert" in IPv4 and IPv6 packets" [RFC6398].

A node that is configured to process a Router Alert option using the Slow Path MUST protect itself from infrastructure attack that could result from processing on the Slow Path. This might include some combination of access control list to only permit from trusted nodes, rate limiting of processing, or other methods [RFC6398].

5.4. Configuration

Section 4 of [RFC8200] allows for a router to control it’s processing of IPv6 Hop-by-Hop options by local configuration. The text is:

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.
A possible approach to implementing this is to maintain a lookup table based on Option Type of the IPv6 options that are supported in the Fast Path. This would allow for a node to quickly determine if an option is supported and can be processed. If the option is not supported, then the node processes it as described in Section 5.2 of this document.

A node configured not to process HBH options, MUST drop the packet if the top two bits of the Option Type field of the first HBH option is non-zero.

The actions of the lookup table SHOULD be configurable by the operator of the router.

6. New Hop-by-Hop Options

Any new IPv6 Hop-by-Hop option designed in the future should be designed to be processed in the Fast Path. New options MUST NOT be defined that require Slow Path processing. New Hop-by-Hop options SHOULD have the following characteristics:

* Straight forward to process. That is, they should be designed to keep the time to process low.

* Fixed size in 8-octet units. Specifically any new Hop-by-Hop options should not be variable size that could extend beyond what can be executed in the Fast Path.

Any new Hop-by-Hop option that is standardized that does not meet these criteria needs to explain in detail in its specification why this can not be accomplished and that there is a reasonable expectation that it can be proceed in most Fast Path implementations.

7. IANA Considerations

There are no actions required for IANA defined in this document.
8. Security Considerations

Security issues with IPv6 Hop-by-Hop options are well known and have been documented in several places, including [RFC6398], [RFC6192], and [I-D.ietf-v6ops-ipv6-ehs-packet-drops]. The main issue, as noted in Section 4, is that any mechanism that can be used to force packets into the router’s Slow Path can be exploited as a denial of service attack on a transit router by saturating the resources need for router management protocols (e.g., routing protocols, network management protocols, etc.) that may cause the router to fail. Due to this it’s common for transit routers to drop packets with Hop-by-Hop options headers.

While Hop-by-Hop options are not required to be processed in the Slow Path, the Router Alert options is designed to do just that.

This document changes the way Hop-by-Hop options are processed in several ways that significantly reduces the attack surface. These changes include:

* All Hop-by-Hop options (with one exception) must be processed in the Fast Path. Only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.

* Only the Router Alert option can be processed in the Slow Path, and the router must be configured to do so.

* Added criteria to allow control over how Router Alert options are processed and that a node configured to support these options must protect itself from attacks using the Router Alert.

* Limited the default number of Hop-by-Hop options that that can be in a packet to a single Hop-by-Hop option.

* Additional Hop-by-Hop options MAY be included, based on local configuration. Although nodes only process these additional Hop-by-Hop Options if configured to do so.

* Added restrictions to any future new Hop-by-Hop options that limit their size and computational requirements.

The authors believe that these changes significantly reduces the security issues relating to IPv6 Hop-by-Hop options and will enable them to be used safely in the Internet.
9. Acknowledgments

Helpful comments were received from Brian Carpenter, Ron Bonica, Ole Troan, Mark Heard, Tom Herbert, [your name here], and other members of the 6MAN working group.

10. Change log [RFC Editor: Please remove]

draft-hinden-6man-hbh-processing-01, 2021-June-2:

* Expanded terminology section to include Forwarding Plane and Control Plane.
* Changed draft that only one HBH Option MUST be processed and additional HBH Options MAY be processed based on local configuration.
* Clarified that all HBH options (with one exception) must be processed on the Fast Path.
* Kept the Router Alert options as the single exception for Slow Path processing.
* Rewrote and expanded section on New Hop-by-Hop Options.
* Removed requirement for HBH Option size and alignment.
* Removed sections evaluating currently defined HBH Options.
* Added content to the Security Considerations section.
* Added people to the acknowledgements section.
* Numerous editorial changes

draft-hinden-6man-hbh-processing-00, 2020-Nov-29:

* Initial draft.

11. Normative References


12. Informative References


Authors’ Addresses

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
United States of America

Email: bob.hinden@gmail.com

Godred Fairhurst
University of Aberdeen
Operational Issues with Processing of the Hop-by-Hop Options Header
draft-ietf-v6ops-hbh-00

Abstract

This document describes the processing of the Hop-by-Hop Options Header (HBH) in today's routers in the aspects of standards specification, common implementations, and default operations. This document outlines the reasons why the Hop-by-Hop Options Header is rarely utilized in current networks. In addition, this document describes how the HBH could be used as a powerful mechanism allowing deployment and operations of new services requiring a more optimized way to leverage network resources of an infrastructure. The Hop-by-Hop Options Header is taken into consideration by several network operators as a valuable container for carrying the information facilitating the introduction of new services. The purpose of this draft is to document the reasons why the HBH is rarely used within networks and to define a proper list of requirements aiming to allow a better leverage of the HBH capability.

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 12 April 2022.

Copyright Notice

Copyright (c) 2021 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
2. Modern Router Architecture .................................. 4
3. Specification of RFC 8200 .................................... 7
4. Common Implementations ....................................... 8
   4.1. Historical Reasons ..................................... 9
   4.2. Consequences ........................................ 9
5. Operators’ Typical Processing ................................ 9
6. New Services ................................................ 10
7. Requirements ................................................ 10
8. Migration Strategies ......................................... 11
9. Security Considerations ...................................... 11
10. IANA Considerations ......................................... 11
11. Acknowledgements ........................................... 11
12. References ................................................ 11
   12.1. Normative References ................................. 11
   12.2. Informative References ............................... 12
Authors’ Addresses .............................................. 13
1. Introduction

Due to historical reasons, such as incapable ASICs, limited IPv6 deployments, and few service requirements, the most common Hop-by-Hop Options header (HBH) processing implementation is that the node sends the IPv6 packets with the Hop-by-Hop Options header to the control plane of the node. The option type of each option carried within the Hop-by-Hop Options header will not even be examined before the packet is sent to the control plane. Very often, such processing behavior is the default configuration or, even worse, is the only behavior of the ipv6 implementation of the node.

Such default processing behavior of the Hop-by-Hop Options header could result in various unpleasant effects such as a risk of Denial of Service (DoS) attack on the router control plane and inconsistent packet drops due to rate limiting on the interface between the router control plane and forwarding plane, which will impact the normal end-to-end IP forwarding of the network services.

This actually introduced a circular problem:

- An implementation problem caused HBH to become a DoS vector.
- Because HBH is a DoS vector, network operators deployed ACLs that discard packets containing HBH.
- Because network operators deployed ACLs that discard packets containing HBH, network designers stopped defining new HBH Options.
- Because network designers stopped defining new HBH Options, the community was not motivated to fix the implementation problem that cause HBH to become a DoS vector.

Driven by the wide deployments of IPv6 and ever-emerging new services, the Hop-by-Hop Options Header is taken as a valuable container for carrying the information to facilitate these new services.

The purpose of this work is to

* Break the endless cycle that resulted in HBH to become a DOS vector.
* Enable the HBH options header to be utilized in a safe and secure way without impacting the management plane.
* Ease the deployments of the new HBH based network services in a multi-vendor scenario that can now be deployed without operational impact.

In this draft, the reasons why the HBH is rarely used within networks will be documented and a proper list of requirements aiming to allow a better leverage of the HBH capability will be defined.

2. Modern Router Architecture

Modern router architecture design maintains a strict separation of the router control plane and its forwarding plane [RFC6192], as shown in Figure 1. Either the control plane or the forwarding plane is composed of both software and hardware, but each plane is responsible for different functions. In this draft, we focus on only the routers following the architecture as shown in Figure 1 and those being deployed in the network rather than those at home.

```
+----------------+
| Router Control |
|     Plane     |
+----------------+
   |   |
   | Interface Z |
   |   |
+----------------+
|   Forwarding   |
|     Plane     |
+----------------+
```

Figure 1. Modern Router Architecture

The router control plane supports routing and management functions, handling packets destined to the device as well as building and sending packets originated locally on the device, and also drives the programming of the forwarding plane. The router control plane is generally realized in software on general-purpose processors, and its hardware is usually not optimized for high-speed packet handling. Because of the wide range of functionality, it is more susceptible to security vulnerabilities and a more likely a target for a DoS attack.
The forwarding plane is typically responsible for receiving a packet on an incoming interface, performing a lookup to identify the packet’s next hop and determine the outgoing interface towards the destination, and forwarding the packet out through the appropriate outgoing interface. Typically, forwarding plane functionality is realized in high-performance Application Specific Integrated Circuits (ASICs) or Network Processors (NFs) that are capable of handling very high packet rates.

The router control plane interfaces with its forwarding plane through the Interface Z, as shown in the Figure 1, and the forwarding plane connects to other network devices via Interfaces such as X and Y. Since the router control plane is vulnerable to the DoS attack, usually a traffic filtering mechanism is implemented on Interface Z in order to block unwanted traffic. In order to protect the router control plane, a rate-limiting mechanism is always implemented on this interface. However, such rate limiting mechanism will always cause inconsistent packet drops, which will impact the normal IP forwarding.

Semiconductor chip technology has advanced significantly in the last decade, and as such the widely used network processing and forwarding process can now not only forward packets at line speed, but also easily support other feature processing such as QoS for DiffServ/MPLS, Access Control List (ACL), Firewall, and Deep Packet Inspection (DPI).

A Network Processing Unit (NPU) is a non-ASIC based Integrated Circuit (IC) that is programmable through software. It performs all packet header operations between the physical layer interface and the switching fabric such as packet parsing and forwarding, modification, and forwarding. Many equipment vendors implement these functions in fixed function ASICs rather than using "off-the-shelf" NPUs, because of proprietary algorithms.

Classification Co-processor is a specialized processor that can be used to lighten the processing load on an NPU by handling the parsing and classification of incoming packets such as IPv6 extended header HBH options processing. This advancement enables network processors to do the general process to handle simple control messages for traffic management, such as signaling for hardware programming, congestion state report, OAM, etc. Industry trend is for intelligent multi-core CPU hardware using modern NPUs for forwarding packets at line rate while still being able to perform other complex tasks such as HBH forwarding options processing without having to punt to the control plane.
Many of the packet-processing devices employed in modern switch and router designs are fixed-function ASICs to handle proprietary functions. While these devices can be very efficient for the set of functions they are designed for, they can be very inflexible. There is a tradeoff of price, performance and flexibility when vendors make a choice to use a fixed function ASIC as opposed to NPU. Due to the inflexibility of the fixed function ASIC, tasks that require additional processing such as IPv6 HBH header processing must be punted to the control plane. This problem is still a challenge today and is the reason why operators to protect against control plane DOS attack vector must drop or ignore HBH options. As industry shifts to Merchant Silicon based NPU evolution from fixed function ASIC, the gap will continue to close increasing the viability ubiquitous HBH use cases due to now processing in the forwarding plane.

Most modern routers maintain a strict separation between forwarding plane and control plane hardware. Forwarding plane bandwidth and resources are plentiful, while control plane bandwidth and resources are constrained. In order to protect scarce control plane resources, 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. In this case HBH control options would be required to be punted to control plane by fixed function ASICs as well as NPUs.

The maximum length of an HBH Options header is 2,048 bytes. A source node can encode hundreds of options in 2,048 bytes [I-D.herbert-6man-eh-limits]. With today’s technology it would be cost prohibitive to be able to process hundreds of options with either NPU or proprietary fixed function ASIC.

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 [RFC8200]. This can be beneficial in cases where transit nodes are legacy hardware and the destination endpoint PE is newer NPU based hardware that can process HBH in the forwarding plane.

IPv6 Extended Header limitations that need to be addressed to make HBH processing more efficient and viable in the forwarding plane:

[RFC8504] defines the IPv6 node requirements and how to protect a node from excessive header chain and excessive header options with various limitations that can be defined on a node. [RFC8883] defines
ICMPv6 Errors for discarding packets due to processing limits. Per [RFC8200] HBH options must be processed serially. However, an implementation of options processing can be made to be done with more parallelism in serial processing grouping of similar options to be processed in parallel.

The IPv6 standard does not currently limit the header chain length or number of options that can be encoded.

Each Option is encoded in a TLV and so processing of a long list of TLVs is expensive. Zero data length encoded options TLVs are a valid option. A DOS vector could be easily generated by encoding 1000 HBH options (Zero data length) in a standard 1500 MTU packet. So now imagine if you have a Christmas tree long header chain to parse each with many options.

3. Specification of RFC 8200

[RFC8200] defines several IPv6 extension header types, including the Hop-by-Hop (HBH) Options header. As specified in [RFC8200], the Hop-by-Hop (HBH) Options header is used to carry optional information that will be examined and processed by every node along a packet’s delivery path, and it is identified by a Next Header value of zero in the IPv6 header.

The Hop-by-Hop (HBH) Options header contains the following fields:

-- Next Header: 8-bit selector, identifies the type of header immediately following the Hop-by-Hop Options header.

-- Hdr Ext Len: 8-bit unsigned integer, the 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.

The Hop-by-Hop (HBH) Options header carries a variable number of "options" that are encoded in the format of type-length-value (TLV).

The highest-order two bits (i.e., the ACT bits) of the Option Type specify the action that must be taken if the processing IPv6 node does not recognize the Option Type. The third-highest-order bit (i.e., the CHG 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.
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 [RFC8200]. It means that the HBH processing behavior in a node depends on its configuration.

However, in the current [RFC8200], there is no explicit specification of the possible configurations. Therefore, the nodes may be configured to ignore the Hop-by-Hop Options header, drop packets containing a Hop-by-Hop Options header, or assign packets containing a Hop-by-Hop Options header to the control plane [RFC8200]. Because of these likely uncertain processing behaviors, new hop-by-hop options are not recommended.

4. Common Implementations

In the current common implementations, once an IPv6 packet, with its Next Header field set to 0, arrives at a node, it will be directly sent to the control plane of the node. With such implementations, the value of the Next Header field in the IPv6 header is the only trigger for the default processing behavior. The option type of each option carried within the Hop-by-Hop Options header will not even be examined before the packet is sent to the control plane.

Very often, such processing behavior is the default configuration on the node, which is embedded in the implementation and cannot be changed or reconfigured.

Another critical component of IPv6 HBH processing, in some cases overlooked, is the operator core network which can be designed to use the global Internet routing table for internet traffic and in other cases use an overlay MPLS VPN to carry Internet traffic.

In the global Internet routing table scenario where only an underlay global routing table exists, and no VPN overlay carrying customer Internet traffic, the IPv6 HBH options can be used as a DOS attack vector for both the operator nodes, adjacent inter-as peer nodes as well as customer nodes along a path.

In a case where the Internet routing table is carried in a MPLS VPN overlay payload, the HBH options header does not impact the operator underlay framework and only impacts the VPN overlay payload and thus the operator underlay topmost label global table routing FEC LSP instantiation is not impacted as the operator underlay is within the operators closed domain.
However, HBH options DOS attack vector in the VPN overlay can still impact the customer CE destination end nodes as well as other adjacent inter-as operators that only use underlay global Internet routing table. In an operator closed domain where MPLS VPN overlay is utilized to carry internet traffic, the operator has full control of the underlay and IPv6 Extended header chain length as well as the number of HBH options encoded.

In the global routing table scenario for Internet traffic there is no way to control the IPv6 Extended header chain length as well as the number of HBH options encoded.

4.1. Historical Reasons

When IPv6 was first implemented on high-speed routers, HBH options were not yet well-understood and ASICs were not as capable as they are today. So, early IPv6 implementations dispatched all packets that contain HBH options to their control plane.

4.2. Consequences

Such implementation introduces a risk of a DoS attack on the control plane of the node, and a large flow of IPv6 packets could congest the control plane, causing other critical functions (including routing and network management) that are executed on the control plane to fail. Rate limiting mechanisms will cause inconsistent packet drops and impact the normal end-to-end IP forwarding of the network services.

5. Operators’ Typical Processing

To mitigate this DoS vulnerability, many operators deployed Access Control Lists (ACLs) that discard all packets containing HBH Options.

[RFC6564] shows the Reports from the field indicating that some IP routers deployed within the global Internet are configured either to ignore or to drop packets having a hop-by-hop header. As stated in [RFC7872], many network operators perceive HBH Options to be a breach of the separation between the forwarding and control planes. Therefore, several network operators configured their nodes so as to discard all packets containing the HBH Options Extension Header, while others configured nodes to forward the packet but to ignore the HBH Options. [RFC7045] also states that hop-by-hop options are not handled by many high-speed routers or are processed only on a control plane.
Due to such behaviors observed and described in these specifications, new hop-by-hop options are not recommended in [RFC8200] hence the usability of HBH options is severely limited.

6. New Services

As IPv6 is being rapidly and widely deployed worldwide, more and more applications and network services are migrating to or directly adopting IPv6. More and more new services that require HBH are emerging and the HBH Options header is going to be utilized by the new services in various scenarios.

In-situ OAM (IOAM) with IPv6 encapsulation [I-D.ietf-ippm-ioam-ipv6-options] is one of the examples. IOAM in IPv6 is used to enhance diagnostics of IPv6 networks and complements other mechanisms, such as the IPv6 Performance and Diagnostic Metrics Destination Option described in [RFC8250]. The IOAM data fields are encapsulated in "option data" fields of the Hop-by-Hop Options header.

Alternate Marking Method can be used as the passive performance measurement tool in an IPv6 domain. The AltMark Option is defined as a new IPv6 extension header option to encode alternate marking technique and Hop-by-Hop Options Header is considered [I-D.ietf-6man-ipv6-alt-mark].

The Minimum Path MTU Hop-by-Hop Option is defined in [I-D.ietf-6man-mtu-option] to record the minimum Path MTU along the forward path between a source host to a destination host. This Hop-by-Hop option is intended to be used in environments like Data Centers and on paths between Data Centers as well as other environments including the general Internet. It provides a useful tool for allowing to better take advantage of paths able to support a large Path MTU.

As more services start utilizing the HBH Options header, more packets containing HBH Options are going to be injected into the networks. According to the current common configuration in most network deployments, all the packets of the new services are going to be sent to the control plane of the nodes, with the possible consequence of causing a DoS on the control plane. The packets will be dropped and the normal IP forwarding may be severely impacted. The deployment of new network services involving multi-vendor interoperability will become impossible.

7. Requirements
8. Migration Strategies

In order to make the HBH options header usable and facilitate the ever-emerging new services to be deployed across multiple vendors’ devices, the new HBH header scheme, SHOULD allow a smooth migration from old to new behavior without disruption time. Also, co-existence between old and news scheme MUST be possible.

9. Security Considerations

The same as the Security Considerations apply as in [RFC8200] for the part related with the HBH Options header.

10. IANA Considerations

This document does not include an IANA request.

11. Acknowledgements

The authors would like to acknowledge Ron Bonica, Fred Baker, Bob Hinden, Stefano Previdi, and Donald Eastlake for their valuable review and comments.

12. References

12.1. Normative References

[Page 11]
12.2. Informative References

[I-D.herbert-6man-eh-limits]
Herbert, T., "Limits on Sending and Processing IPv6 Extension Headers", Work in Progress, Internet-Draft, draft-herbert-6man-eh-limits-00, 22 June 2021, 
[I-D.ietf-6man-ipv6-alt-mark]

[I-D.ietf-6man-mtu-option]

[I-D.ietf-ippm-ioam-ipv6-options]


Authors’ Addresses

Shuping Peng
Huawei Technologies
Beijing
China

Email: pengshuping@huawei.com
Zhenbin Li
Huawei Technologies
Beijing
China
Email: lizhenbin@huawei.com

Chongfeng Xie
China Telecom
China
Email: xiechf@chinatelecom.cn

Zhuangzhuang Qin
China Unicom
Beijing
China
Email: qinzhuangzhuang@chinaunicom.cn

Gyan Mishra
Verizon Inc.
United States of America
Email: gyan.s.mishra@verizon.com
Abstract

Several IPv6 transition technologies have been developed to provide customers with IPv4-as-a-Service (IPv4aaS) for ISPs with an IPv6-only access and/or core network. All these technologies have their advantages and disadvantages, and depending on existing topology, skills, strategy and other preferences, one of these technologies may be the most appropriate solution for a network operator.

This document examines the five most prominent IPv4aaS technologies considering a number of different aspects to provide network operators with an easy to use reference to assist in selecting the technology that best suits their needs.
1. Introduction ................................................. 3
  1.1. Requirements Language .................................. 4
2. Overview of the Technologies ................................. 4
  2.1. 464XLAT .................................................. 4
  2.2. Dual-Stack Lite .......................................... 5
  2.3. Lightweight 4over6 ....................................... 6
  2.4. MAP-E .................................................... 6
  2.5. MAP-T .................................................... 7
3. High-level Architectures and their Consequences .......... 8
  3.1. Service Provider Network Traversal ....................... 8
  3.2. Network Address Translation .............................. 9
  3.3. IPv4 Address Sharing .................................... 10
  3.4. IPv4 Pool Size Considerations ........................... 11
  3.5. CE Provisioning Considerations .......................... 12
  3.6. Support for Multicast ................................... 13
4. Detailed Analysis ............................................ 13
  4.1. Architectural Differences ............................... 13
    4.1.1. Basic Comparison ..................................... 13
  4.2. Tradeoff between Port Number Efficiency and Stateless Operation ..................................... 14
  4.3. Support for Public Server Operation ..................... 16
  4.4. Support and Implementations .............................. 17
    4.4.1. OS Support ........................................... 17
    4.4.2. Support in Cellular and Broadband Networks ........ 17
    4.4.3. Implementation Code Sizes ............................ 18
  4.5. Typical Deployment and Traffic Volume Considerations .. 18
    4.5.1. Deployment Possibilities ............................. 18
    4.5.2. Cellular Networks with 464XLAT ....................... 18
    4.5.3. Wireline Networks with 464XLAT ....................... 19
  4.6. Load Sharing ............................................ 19
  4.7. Logging ................................................ 20
1. Introduction

As the deployment of IPv6 becomes more prevalent, it follows that network operators will move to building single-stack IPv6 core and access networks to simplify network planning and operations. However, providing customers with IPv4 services continues to be a requirement for the foreseeable future. To meet this need, the IETF has standardized a number of different IPv4aaS technologies for this [LEN2019] based on differing requirements and deployment scenarios.

The number of technologies that have been developed makes it time consuming for a network operator to identify the most appropriate mechanism for their specific deployment. This document provides a comparative analysis of the most commonly used mechanisms to assist operators with this problem.

Five different IPv4aaS solutions are considered. The following IPv6 transition technologies are covered:

1. 464XLAT [RFC6877]
2. Dual Stack Lite [RFC6333]
3. lw4o6 (Lightweight 4over6) [RFC7596]
4. MAP-E [RFC7597]
5. MAP-T [RFC7599]
We note that [RFC6180] gives guidelines for using IPv6 transition mechanisms during IPv6 deployment addressing a much broader topic, whereas this document focuses on a small part of it.

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Overview of the Technologies

The following sections introduce the different technologies analyzed in this document, describing some of their most important characteristics.

2.1. 464XLAT

464XLAT is a single/dual translation model, which uses a customer-side translator (CLAT) located in the customer’s device to perform stateless NAT64 translation [RFC7915] (more precisely, stateless NAT46, a stateless IP/ICMP translation from IPv4 to IPv6). IPv4-embedded IPv6 addresses [RFC6052] are used for both source and destination addresses. Commonly, a /96 prefix (either the 64:ff9b::/96 Well-Known Prefix, or a Network-Specific Prefix) is used as the IPv6 destination for the IPv4-embedded client traffic.

In the operator’s network, the provider-side translator (PLAT) performs stateful NAT64 [RFC6146] to translate the traffic. The destination IPv4 address is extracted from the IPv4-embedded IPv6 packet destination address and the source address is from a pool of public IPv4 addresses.

Alternatively, when a dedicated /64 is not available for translation, the CLAT device uses a stateful NAT44 translation before the stateless NAT46 translation.

Note that we generally do not see state close to the end-user as equally problematic as state in the middle of the network.

In typical deployments, 464XLAT is used together with DNS64 [RFC6147], see Section 3.1.2 of [RFC8683]. When an IPv6-only client or application communicates with an IPv4-only server, the DNS64 server returns the IPv4-embedded IPv6 address of the IPv4-only server. In this case, the IPv6-only client sends out IPv6 packets, thus CLAT functions as an IPv6 router and the PLAT performs a
stateful NAT64 for these packets. In this case, there is a single translation.

Alternatively, one can say that the DNS64 + stateful NAT64 is used to carry the traffic of the IPv6-only client and the IPv4-only server, and the CLAT is used only for the IPv4 traffic from applications or devices that use literal IPv4 addresses or non-IPv6 compliant APIs.

![Diagram of 464XLAT architecture](image)

Figure 1: Overview of the 464XLAT architecture

Note: in mobile networks, CLAT is commonly implemented in the user’s equipment (UE or smartphone).

2.2. Dual-Stack Lite

Dual-Stack Lite (DS-Lite) [RFC6333] was the first of the considered transition mechanisms to be developed. DS-Lite uses a ‘Basic Broadband Bridging’ (B4) function in the customer’s CE router that encapsulates IPv4 in IPv6 traffic and sends it over the IPv6 native service-provider network to a centralized ‘Address Family Transition Router’ (AFTR). The AFTR performs encapsulation/decapsulation of the 4in6 traffic and translates the IPv4 payload to public IPv4 source address using a stateful NAPT44 function.

![Diagram of DS-Lite architecture](image)

Figure 2: Overview of the DS-Lite architecture
2.3. Lightweight 4over6

Lightweight 4over6 (lw4o6) is a variant of DS-Lite. The main difference is that the stateful NAPT44 function is relocated from the centralized AFTR to the customer’s B4 element (called a lwB4). The AFTR (called a lwAFTR) function therefore only performs A+P routing and 4in6 encapsulation/decapsulation.

Routing to the correct client and IPv4 address sharing is achieved using the Address + Port (A+P) model [RFC6346] of provisioning each lwB4 with a unique tuple of IPv4 address unique range of layer-4 ports. The client uses these for NAPT44.

The lwAFTR implements a binding table, which has a per-client entry linking the customer’s source IPv4 address and allocated range of layer-4 ports to their IPv6 tunnel endpoint address. The binding table allows egress traffic from customers to be validated (to prevent spoofing) and ingress traffic to be correctly encapsulated and forwarded. As there needs to be a per-client entry, an lwAFTR implementation needs to be optimized for performing a per-packet lookup on the binding table.

Direct communication between two lwB4s is performed by hair-pinning traffic through the lwAFTR.

```
+-------------+             +----------+
Private |    lwB4     | IPv4-in-IPv6| Stateless|
| IPv4 |------->|      |Encap.|------------>|(encap/A+P|    ( IPv4  )
|Device|<-------| NAPT | 44  |Decap.|  /<-------- bind. tab +--( Internet ) |
+------+        |      +----------+
 Operator IPv6
network
```

Figure 3: Overview of the lw4o6 architecture

2.4. MAP-E

MAP-E uses a stateless algorithm to embed portions of the customer’s allocated IPv4 address (or part of an address with A+P routing) into the IPv6 prefix delegated to the client. This allows for large numbers of clients to be provisioned using a single MAP rule (called a MAP domain). The algorithm also allows for direct IPv4 peer-to-peer communication between hosts provisioned with common MAP rules.

The CE (Customer-Edge) router typically performs stateful NAPT44 [RFC2663] to translate the private IPv4 source addresses and source
ports into an address and port range defined by applying the MAP rule applied to the delegated IPv6 prefix. The client address/port allocation size is a design parameter. The CE router then encapsulates the IPv4 packet in an IPv6 packet [RFC2473] and sends it directly to another host in the MAP domain (for peer-to-peer) or to a Border Router (BR) if the IPv4 destination is not covered in one of the CE’s MAP rules.

The MAP BR is provisioned with the set of MAP rules for the MAP domains it serves. These rules determine how the MAP BR is to decapsulate traffic that it receives from client, validating the source IPv4 address and layer 4 ports assigned, as well as how to calculate the destination IPv6 address for ingress IPv4 traffic.

A MAP CE typically performs stateful NAPT44 to translate traffic to a public IPv4 address and port-range calculated by applying the provisioned Basic MAP Rule (BMR - a set of inputs to the algorithm) to the delegated IPv6 prefix. The CE then performs stateless translation from IPv4 to IPv6 [RFC7915]. The MAP BR is provisioned with the same BMR as the client, enabling the received IPv6 traffic to be statelessly NAT64 translated back to the public IPv4 source address used by the client.

Using translation instead of encapsulation also allows IPv4-only nodes to correspond directly with IPv6 nodes in the MAP-T domain that have IPv4-embedded IPv6 addresses.

2.5. MAP-T

MAP-T uses the same mapping algorithm as MAP-E. The major difference is that double stateless translation (NAT46 in the CE and NAT64 in the BR) is used to traverse the ISP’s IPv6 single-stack network. MAP-T can also be compared to 464XLAT when there is a double translation.

A MAP CE typically performs stateful NAPT44 to translate traffic to a public IPv4 address and port-range calculated by applying the provisioned Basic MAP Rule (BMR - a set of inputs to the algorithm) to the delegated IPv6 prefix. The CE then performs stateless translation from IPv4 to IPv6 [RFC7915]. The MAP BR is provisioned with the same BMR as the client, enabling the received IPv6 traffic to be statelessly NAT64 translated back to the public IPv4 source address used by the client.

Using translation instead of encapsulation also allows IPv4-only nodes to correspond directly with IPv6 nodes in the MAP-T domain that have IPv4-embedded IPv6 addresses.
3. High-level Architectures and their Consequences

3.1. Service Provider Network Traversal

For the data-plane, there are two approaches for traversing the IPv6 provider network:

- 4-6-4 translation
- 4-in-6 encapsulation

In the scope of this document, all of the encapsulation based mechanisms use IP-in-IP tunnelling [RFC2473]. This is a stateless tunneling mechanism which does not require any additional tunnel headers.

It should be noted that both of these approaches result in an increase in the size of the packet that needs to be transported across the operator’s network when compared to native IPv4. 4-6-4 translation adds a 20-bytes overhead (the 20-byte IPv4 header is replaced with a 40-byte IPv6 header). Encapsulation has a 40-byte overhead (an IPv6 header is prepended to the IPv4 header).

The increase in packet size can become a significant problem if there is a link with a smaller MTU in the traffic path. This may result in traffic needing to be fragmented at the ingress point to the IPv6 only domain (i.e., the NAT46 or 4in6 encapsulation endpoint). It may
also result in the need to implement buffering and fragment re-assembly in the BR node.

The advice given in [RFC7597] Section 8.3.1 is applicable to all of these mechanisms: It is strongly recommended that the MTU in the IPv6-only domain be well managed and that the IPv6 MTU on the CE WAN-side interface be set so that no fragmentation occurs within the boundary of the IPv6-only domain.

3.2. Network Address Translation

For the high-level solution of IPv6 service provider network traversal, MAP-T uses double stateless translation. First at the CE from IPv4 to IPv6 (NAT46), and then from IPv6 to IPv4 (NAT64), at the service provider network.

464XLAT may use double translation (stateless NAT46 + stateful NAT64) or single translation (stateful NAT64), depending on different factors, such as the use of DNS by the applications and the availability of a DNS64 function (in the host or in the service provider network). For deployment guidelines, please refer to [RFC8683].

The first step for the double translation mechanisms is a stateless NAT from IPv4 to IPv6 implemented as SIIT (Stateless IP/ICMP Translation Algorithm) [RFC7915], which does not translate IPv4 header options and/or multicast IP/ICMP packets. With encapsulation-based technologies the header is transported intact and multicast can also be carried.

Single and double translation results in native IPv6 traffic with a layer-4 next-header. The fields in these headers can be used for functions such as hashing across equal-cost multipaths or ACLs. For encapsulation, there is an IPv6 header followed by an IPv4 header. This results in less entropy for hashing algorithms, and may mean that devices in the traffic path that perform header inspection (e.g. router ACLs or firewalls) require the functionality to look into the payload header.

Solutions using double translation can only carry port-aware IP protocols (e.g. TCP, UDP) and ICMP when they are used with IPv4 address sharing (please refer to Section 4.3 for more details). Encapsulation based solutions can carry any other protocols over IP, too.

An in-depth analysis of stateful NAT64 can be found in [RFC6889].
3.3. IPv4 Address Sharing

As public IPv4 address exhaustion is a common motivation for deploying IPv6, transition technologies need to provide a solution for allowing public IPv4 address sharing.

In order to fulfill this requirement, a stateful NAPT function is a necessary function in all of the mechanisms. The major differentiator is where in the architecture this function is located.

The solutions compared by this document fall into two categories:

- CGN-based approaches (DS-Lite, 464XLAT)
- A+P-based approaches (lw4o6, MAP-E, MAP-T)

In the CGN-based model, a device such as a CGN/AFTR or NAT64 performs the NAPT44 function and maintains per-session state for all of the active client’s traffic. The customer’s device does not require per-session state for NAPT.

In the A+P-based model, a device (usually a CE) performs stateful NAPT44 and maintains per-session state only co-located devices, e.g. in the customer’s home network. Here, the centralized network function (lwAFTR or BR) only needs to perform stateless encapsulation/decapsulation or NAT64.

Issues related to IPv4 address sharing mechanisms are described in [RFC6269] and should also be considered.

The address sharing efficiency of the five technologies is significantly different, it is discussed in Section 4.2.

lw4o6, MAP-E and MAP-T can also be configured without IPv4 address sharing, see the details in Section 4.3. However, in that case, there is no advantage in terms of public IPv4 address saving. In the case of 464XLAT, this can be achieved as well through EAMT [RFC7757].

Conversely, both MAP-E and MAP-T may be configured to provide more than one public IPv4 address (i.e., an IPv4 prefix shorter than a /32) to customers.

Dynamic DNS issues in address-sharing contexts and their possible solutions using PCP (Port Control Protocol) are discussed in detail in [RFC7393].
3.4. IPv4 Pool Size Considerations

In most networks, it is possible to, using existing data about flows to CDNs/caches or other well-known IPv6-enabled destinations, calculate the percentage of traffic that would turn into IPv6 if it is enabled on that network or part of it.

Knowing that, it is possible to calculate the IPv4 pool size required for a given number of subscribers, depending on the IPv4aaS technology being used.

Often it is assumed that each user-device (computer, tablet, smartphone) behind a NAT, could simultaneously use about 300 ports. Typically, in the case of a residential subscriber, there will be a maximum of 4 of those devices in use simultaneously, which means a total of 1,200 ports.

If for example, 80% of the traffic is expected towards IPv6 destinations, only 20% will actually be using IPv4 ports, so in our example, that will mean 240 ports required per subscriber.

From the 65,535 ports available per IPv4 address, we could even consider reserving 1,024 ports, in order to allow customers that need EAMT entries for incoming connections to well-known ports, which means 64,511 ports actually available per each IPv4 address.

According to this, a /22 (1,024 public IPv4 addresses) will be sufficient for over 275,000 subscribers (1,024x64,511/240=275,246.93).

Similarly, a /18 (16,384 public IPv4 addresses) will be sufficient for over 4,403,940 subscribers, and so on.

This is a conservative approach, which is valid in the case of 464XLAT, because ports are assigned dynamically by the NAT64, so it is not necessary to consider if one user is actually using more or less ports: Average values work well.

Further to that, the NAT46 in the CLAT will actually hide the subscriber LANs port usage and only that will be the limiting factor. So even subscribers with higher demands of ports, will not make a big difference.

As the deployment of IPv6 progresses, the use of NAT64, and therefore of public IPv4 addresses, decreases (more IPv6/ports, less IPv4/ports), so either more subscribers can be accommodated with the same number of IPv4 addresses, or some of those addressed can be retired from the NAT64.
For comparison, if dual-stack is being used, any given number of users will require the same number of public IPv4 addresses. For instance, a /14 will provide 262,144 IPv4 public addresses for 262,144 subscribers, versus 275,000 subscribers being served with a only a /22.

In the other IPv4aaS technologies, this calculation will only match if the assignment of ports per subscriber can be done dynamically, which is not always the case (depending on the vendor implementation).

An alternative approximation for the other IPv4aaS technologies, when dynamically assignment of addresses is not possible, must ensure sufficient number of ports per subscriber. That means 1,200 ports, and typically, it comes to 2,000 ports in many deployments. In that case, assuming 80% of IPv6 traffic, as above, which will allow only 30 subscribers per each IPv4 address, so the closer approximation to 275,000 subscribers per our example with 464XLAT (with a /22), will be using a /19, which serves 245,760 subscribers (a /19 has 8,192 addresses, 30 subscribers with 2,000 ports each, per address).

If the CGN (in case of DS-Lite) or the CE (in case of lw4o6, MAP-E and MAP-T) make use of a 5-tuple for tracking the NAT connections, the number of ports required per subscriber can be limited as low as 4 ports per subscriber. However, the practical limit depends on the desired limit for parallel connections that any single host behind the NAT can have to the same address and port in Internet. Note that it is becoming more common that applications use AJAX and similar mechanisms, so taking that extreme limit is probably not a very a safe choice.

This extremely reduced number of ports "feature" could also be used in case the CLAT-enabled CE with 464XLAT makes use of the 5-tuple NAT connections tracking, and could also be further extended if the NAT64 also use the 5-tuple.

3.5. CE Provisioning Considerations

All of the technologies require some provisioning of customer devices. The table below shows which methods currently have extensions for provisioning the different mechanisms.
Table 2: Available Provisioning Mechanisms


3.6. Support for Multicast

The solutions covered in this document are all intended for unicast traffic. [RFC8114] describes a method for carrying encapsulated IPv4 multicast traffic over an IPv6 multicast network. This could be deployed in parallel to any of the operator’s chosen IPv4aaS mechanism.

4. Detailed Analysis

4.1. Architectural Differences

4.1.1. Basic Comparison

The five IPv4aaS technologies can be classified into 2x2=4 categories on the basis of two aspects:

- Technology used for service provider network traversal. It can be single/double translation or encapsulation.
- Presence or absence of NAPT44 per-flow state in the operator network.

Table 3: Available Provisioning Mechanisms
4.2. Tradeoff between Port Number Efficiency and Stateless Operation

464XLAT and DS-Lite use stateful NAPT at the PLAT/AFTR devices, respectively. This may cause scalability issues for the number of clients or volume of traffic, but does not impose a limitation on the number of ports per user, as they can be allocated dynamically on-demand and the allocation policy can be centrally managed/adjusted.

A+P based mechanisms (lw4o6, MAP-E, and MAP-T) avoid using NAPT in the service provider network. However, this means that the number of ports provided to each user (and hence the effective IPv4 address sharing ratio) must be pre-provisioned to the client.

Changing the allocated port ranges with A+P based technologies, requires more planning and is likely to involve re-provisioning both hosts and operator side equipment. It should be noted that due to the per-customer binding table entry used by lw4o6, a single customer can be re-provisioned (e.g., if they request a full IPv4 address) without needing to change parameters for a number of customers as in a MAP domain.

It is also worth noting that there is a direct relationship between the efficiency of customer public port-allocations and the corresponding logging overhead that may be necessary to meet data-retention requirements. This is considered in Section 4.7 below.

Determining the optimal number of ports for a fixed port set is not an easy task, and may also be impacted by local regulatory law, which may define a maximum number of users per IP address, and consequently a minimum number of ports per user.

On the one hand, the "lack of ports" situation may cause serious problems in the operation of certain applications. For example, Miyakawa has demonstrated the consequences of the session number limitation due to port number shortage on the example of Google Maps [MIY2010]. When the limit was 15, several blocks of the map were missing, and the map was unusable. This study also provided several examples for the session numbers of different applications (the highest one was Apple’s iTunes: 230-270 ports).

The port number consumption of different applications is highly varying and e.g. in the case of web browsing it depends on several factors, including the choice of the web page, the web browser, and sometimes even the operating system [REP2014]. For example, under certain conditions, 120-160 ports were used (URL: sohu.com, browser: Firefox under Ubuntu Linux), and in some other cases it was only 3-12 ports (URL: twitter.com, browser: Iceweasel under Debian Linux).
There may be several users behind a CE router, especially in the broadband case (e.g., Internet is used by different members of a family simultaneously), so sufficient ports must be allocated to avoid impacting user experience.

Furthermore, assigning too many ports per CE router will result in waste of public IPv4 addresses, which is a scarce and expensive resource. Clearly this is a big advantage in the case of 464XLAT where they are dynamically managed, so that the number of IPv4 addresses for the sharing-pool is smaller while the availability of ports per user don’t need to be pre-defined and is not a limitation for them.

There is a direct tradeoff between the optimization of client port allocations and the associated logging overhead. Section 4.7 discusses this in more depth.

We note that common CE router NAT44 implementations utilizing Netfilter, multiplexes active sessions using a 3-tuple (source address, destination address, and destination port). This means that external source ports can be reused for unique internal source and destination address and port sessions. It is also noted, that Netfilter cannot currently make use of multiple source port ranges (i.e. several blocks of ports distributed across the total port space as is common in MAP deployments), this may influence the design when using stateless technologies.

Stateful technologies, 464XLAT and DS-Lite (and also NAT444) can therefore be much more efficient in terms of port allocation and thus public IP address saving. The price is the stateful operation in the service provider network, which allegedly does not scale up well. It should be noticed that in many cases, all those factors may depend on how it is actually implemented.

Measurements have been started to examine the scalability of a few stateful solutions in two areas:

- How their performance scales up with the number of CPU cores?
- To what extent their performance degrades with the number of concurrent connections?

The details of the measurements and their results are available from [I-D.lencse-v6ops-transition-scalability].

We note that some CGN-type solutions can allocate ports dynamically "on the fly". Depending on configuration, this can result in the same customer being allocated ports from different source addresses.
This can cause operational issues for protocols and applications that expect multiple flows to be sourced from the same address. E.g., ECMP hashing, STUN, gaming, content delivery networks. However, it should be noticed that this is the same problem when a network has a NAT44 with multiple public IPv4 addresses, or even when applications in a dual-stack case, behave wrongly if happy eyeballs is flapping the flow address between IPv4 and IPv6.

The consequences of IPv4 address sharing [RFC6269] may impact all five technologies. However, when ports are allocated statically, more customers may get ports from the same public IPv4 address, which may results in negative consequences with higher probability, e.g. many applications and service providers (Sony PlayStation Network, OpenDNS, etc.) permanently black-list IPv4 ranges if they detect that they are used for address sharing.

Both cases are, again, implementation dependent.

We note that although it is not of typical use, one can do deterministic, stateful NAT and reserve a fixed set of ports for each customer, as well.

4.3. Support for Public Server Operation

Mechanisms that rely on operator side per-flow state do not, by themselves, offer a way for customers to present services on publicly accessible layer-4 ports.

Port Control Protocol (PCP) [RFC6877] provides a mechanism for a client to request an external public port from a CGN device. For server operation, it is required with NAT64/464XLAT, and it is supported in some DS-Lite AFTR implementations.

A+P based mechanisms distribute a public IPv4 address and restricted range of layer-4 ports to the client. In this case, it is possible for the user to configure their device to offer a publicly accessible server on one of their allocated ports. It should be noted that commonly operators do not assign the Well-Known-Ports to users (unless they are allocating a full IPv4 address), so the user will need to run the service on an allocated port, or configure port translation.

Lw4o6, MAP-E and MAP-T may be configured to allocated clients with a full IPv4 address, allowing exclusive use of all ports, and non-port-based layer 4 protocols. Thus, they may also be used to support server/services operation on their default ports. However, when public IPv4 addresses are assigned to the CE router without address
sharing, obviously there is no advantage in terms of IPv4 public addresses saving.

It is also possible to configure specific ports mapping in 464XLAT/NAT64 using EAMT [RFC7757], which means that only those ports are "lost" from the pool of addresses, so there is a higher maximization of the total usage of IPv4/port resources.

4.4. Support and Implementations

4.4.1. OS Support

A 464XLAT client (CLAT) is implemented in Windows 10, Linux (including Android), Windows Mobile, Chrome OS and iOS, but at the time of writing is not available in MacOS.

The remaining four solutions are commonly deployed as functions in the CE device only, however in general, except DS-Lite, the vendors support is poor.

The OpenWRT Linux based open-source OS designed for CE devices offers a number of different ‘opkg’ packages as part of the distribution:

- ‘464xlat’ enables support for 464XLAT CLAT functionality
- ‘ds-lite’ enables support for DSLite B4 functionality
- ‘map’ enables support for MAP-E and lw4o6 CE functionality
- ‘map-t’ enables support for MAP-T CE functionality

For the operator side functionality, some free open-source implementations exist:

CLAT, NAT64, EAMT: http://www.jool.mx

MAP-BR, lwAFTR, CGN, CLAT, NAT64: VPP/fd.io
https://gerrit.fd.io/r/#/admin/projects/

lwAFTR: https://github.com/Igalia/snabb

DSLite AFTR: https://www.isc.org/downloads/

4.4.2. Support in Cellular and Broadband Networks

Several cellular networks use 464XLAT, whereas we are not aware of any deployment of the four other technologies in cellular networks, as they are not standardised neither implemented in UE devices.
In broadband networks, there are some deployments of 464XLAT, MAP-E and MAP-T. Lw4o6 and DS-Lite have more deployments, with DS-Lite being the most common, but lw4o6 taking over in the last years.

Please refer to Table 2 and Table 3 of [LEN2019] for a limited set of deployment information.

4.4.3. Implementation Code Sizes

As hint to the relative complexity of the mechanisms, the following code sizes are reported from the OpenWRT implementations of each technology are 17kB, 35kB, 15kB, 35kB, and 48kB for 464XLAT, lw4o6, DS-Lite, MAP-E, MAP-T, and lw4o6, respectively (https://openwrt.org/packages/start).

We note that the support for all five technologies requires much less code size than the total sum of the above quantities, because they contain a lot of common functions (data plane is shared among several of them).

4.5. Typical Deployment and Traffic Volume Considerations

4.5.1. Deployment Possibilities

Theoretically, all five IPv4aaS technologies could be used together with DNS64 + stateful NAT64, as it is done in 464XLAT. In this case the CE router would treat the traffic between an IPv6-only client and IPv4-only server as normal IPv6 traffic, and the stateful NAT64 gateway would do a single translation, thus offloading this kind of traffic from the IPv4aaS technology. The cost of this solution would be the need for deploying also DNS64 + stateful NAT64.

However, this has not been implemented in clients or actual deployments, so only 464XLAT always uses this optimization and the other four solutions do not use it at all.

4.5.2. Cellular Networks with 464XLAT

Figures from existing deployments (end of 2018), show that the typical traffic volumes in an IPv6-only cellular network, when 464XLAT technology is used together with DNS64, are:

- 75% of traffic is IPv6 end-to-end (no translation)
- 24% of traffic uses DNS64 + NAT64 (1 translation)
- Less than 1% of traffic uses the CLAT in addition to NAT64 (2 translations), due to an IPv4 socket and/or IPv4 literal.
Without using DNS64, 25% of the traffic would undergo double translation.

4.5.3. Wireline Networks with 464XLAT

Figures from several existing deployments (end of 2020), mainly with residential customers, show that the typical traffic volumes in an IPv6-only network, when 464XLAT is used with DNS64, are in the following ranges:

- 65%-85% of traffic is IPv6 end-to-end (no translation)
- 14%-34% of traffic uses DNS64 + NAT64 (1 translation)
- Less than 1-2% of traffic uses the CLAT in addition to NAT64 (2 translations), due to an IPv4 socket and/or IPv4 literal.

Without using DNS64, 16%-35% of the traffic would undergo double translation.

4.6. Load Sharing

If multiple network-side devices are needed as PLAT/AFTR/BR for capacity, then there is a need for a load sharing mechanism. ECMP (Equal-Cost Multi-Path) load sharing can be used for all technologies, however stateful technologies will be impacted by changes in network topology or device failure.

Technologies utilizing DNS64 can also distribute load across PLAT/AFTR devices, evenly or unevenly, by using different prefixes. Different network specific prefixes can be distributed for subscribers in appropriately sized segments (like split-horizon DNS, also called DNS views).

Stateless technologies, due to the lack of per-flow state, can make use of anycast routing for load sharing and resiliency across network-devices, both ingress and egress; flows can take asymmetric paths through the network, i.e., in through one lwAFTR/BR and out via another.

Mechanisms with centralized NAPT44 state have a number of challenges specifically related to scaling and resilience. As the total amount of client traffic exceeds the capacity of a single CGN instance, additional nodes are required to handle the load. As each CGN maintains a stateful table of active client sessions, this table may need to be synchronized between CGN instances. This is necessary for two reasons:
To prevent all active customer sessions being dropped in event of a CGN node failure.

To ensure a matching state table entry for an active session in the event of asymmetric routing through different egress and ingress CGN nodes.

4.7. Logging

In the case of 464XLAT and DS-Lite, the user of any given public IPv4 address and port combination will vary over time, therefore, logging is necessary to meet data retention laws. Each entry in the PLAT/AFTR’s generates a logging entry. As discussed in Section 4.2, a client may open hundreds of sessions during common tasks such as web-browsing, each of which needs to be logged so the overall logging burden on the network operator is significant. In some countries, this level of logging is required to comply with data retention legislation.

One common optimization available to reduce the logging overhead is the allocation of a block of ports to a client for the duration of their session. This means that logging entry only needs to be made when the client’s port block is released, which dramatically reducing the logging overhead. This comes as the cost of less efficient public address sharing as clients need to be allocated a port block of a fixed size regardless of the actual number of ports that they are using.

Stateless technologies that pre-allocate the IPv4 addresses and ports only require that copies of the active MAP rules (for MAP-E and MAP-T), or binding-table (for lw4o6) are retained along with timestamp information of when they have been active. Support tools (e.g., those used to serve data retention requests) may need to be updated to be aware of the mechanism in use (e.g., implementing the MAP algorithm so that IPv4 information can be linked to the IPv6 prefix delegated to a client). As stateless technologies do not have a centralized stateful element which customer traffic needs to pass through, so if data retention laws mandate per-session logging, there is no simple way of meeting this requirement with a stateless technology alone. Thus a centralized NAPT44 model may be the only way to meet this requirement.

Deterministic CGN [RFC7422] was proposed as a solution to reduce the resource consumption of logging.
4.8. Optimization for IPv4-only devices/applications

When IPv4-only devices or applications are behind a CE connected with IPv6-only and IPv4aaS, the IPv4-only traffic flows will necessarily, be encapsulated/decapsulated (in the case of DS-Lite, lw4o6 and MAP-E) and will reach the IPv4 address of the destination, even if that service supports dual-stack. This means that the traffic flow will cross thru the AFTR, lwAFTR or BR, depending on the specific transition mechanism being used.

Even if those services are directly connected to the operator network (for example, CDNs, caches), or located internally (such as VoIP, etc.), it is not possible to avoid that overhead.

However, in the case of those mechanism that use a NAT46 function, in the CE (464XLAT and MAP-T), it is possible to take advantage of optimization functionalities, such as the ones described in [I-D.ietf-v6ops-464xlat-optimization].

Using those optimizations, because the NAT46 has already translated the IPv4-only flow to IPv6, and the services are dual-stack, they can be reached without the need to translate them back to IPv4.

5. Performance Comparison

We plan to compare the performances of the most prominent free software implementations of the five IPv6 transition technologies using the methodology described in "Benchmarking Methodology for IPv6 Transition Technologies" [RFC8219].

The Dual DUT Setup of [RFC8219] makes it possible to use the existing "Benchmarking Methodology for Network Interconnect Devices" [RFC2544] compliant measurement devices, however, this solution has two kinds of limitations:

- Dual DUT setup has the drawback that the performances of the CE and of the ISP side device (e.g. the CLAT and the PLAT of 464XLAT) are measured together. In order to measure the performance of only one of them, we need to ensure that the desired one is the bottleneck.

- Measurements procedures for PDV and IPDV measurements are missing from the legacy devices, and the old measurement procedure for Latency has been redefined in [RFC8219].

The Single DUT Setup of [RFC8219] makes it possible to benchmark the selected device separately, but it either requires a special Tester or some trick is need, if we want to use legacy Testers. An example
for the latter is our stateless NAT64 measurements testing Throughput and Frame Loss Rate using a legacy [RFC5180] compliant commercial tester [LEN2020a]

Slitperf, an [RFC8219] compliant DPDK-based software Tester for benchmarking stateless NAT64 gateways has been developed recently and it is available from GitHub [SIITperf] as free software and documented in [LEN2021]. Originally, it literally followed the test frame format of [RFC2544] including "hard wired" source and destination port numbers, and then it has been complemented with the random port feature required by [RFC4814]. The new version is documented in [LEN2020b]

Further DPDK-based, [RFC8219] compliant software testers are being developed at the Budapest University of Technology and Economics as student projects. They are planned to be released as free software, too.

Information about the benchmarking tools, measurements and results will be made available in [I-D.lencse-v6ops-transition-benchmarking].

6. Acknowledgements

The authors would like to thank Ole Troan for his thorough review of this draft and acknowledge the inputs of Mark Andrews, Edwin Cordeiro, Fred Baker, Alexandre Petrescu, Cameron Byrne, Tore Anderson, Mikael Abrahamsson, Gert Doering, Satoru Matsushima, Mohamed Boucadair, Tom Petch, Yannis Nikolopoulos, and TBD ...

7. IANA Considerations

This document does not make any request to IANA.

8. Security Considerations

According to the simplest model, the number of bugs is proportional to the number of code lines. Please refer to Section 4.4.3 for code sizes of CE implementations.

For all five technologies, the CE device should contain a DNS proxy. However, the user may change DNS settings. If it happens and lw4o6, MAP-E and MAP-T are used with significantly restricted port set, which is required for an efficient public IPv4 address sharing, the entropy of the source ports is significantly lowered (e.g. from 16 bits to 10 bits, when 1024 port numbers are assigned to each subscriber) and thus these technologies are theoretically less resilient against cache poisoning, see [RFC5452]. However, an efficient cache poisoning attack requires that the subscriber
operates an own caching DNS server and the attack is performed in the service provider network. Thus, we consider the chance of the successful exploitation of this vulnerability as low.

An in-depth security analysis of all five IPv6 transition technologies and their most prominent free software implementations according to the methodology defined in [LEN2018] is planned.

As the first step, the theoretical security analysis of 464XLAT was done in [Azz2020].

9. References

9.1. Normative References


[RFC7915]  Bao, C., Li, X., Baker, F., Anderson, T., and F. Gont,
"IP/ICMP Translation Algorithm", RFC 7915,
DOI 10.17487/RFC7915, June 2016,

RFC 7950, DOI 10.17487/RFC7950, August 2016,

[RFC8114]  Boucadair, M., Qin, C., Jacquenet, C., Lee, Y., and Q.
Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients
over an IPv6 Multicast Network", RFC 8114,
DOI 10.17487/RFC8114, March 2017,
<https://www.rfc-editor.org/info/rfc8114>.

[RFC8174]  Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,

Methodology for IPv6 Transition Technologies", RFC 8219,
DOI 10.17487/RFC8219, August 2017,

[RFC8415]  Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A.,
Richardson, M., Jiang, S., Lemon, T., and T. Winters,
"Dynamic Host Configuration Protocol for IPv6 (DHCPv6)",
RFC 8415, DOI 10.17487/RFC8415, November 2018,

[RFC8512]  Boucadair, M., Ed., Sivakumar, S., Jacquenet, C.,
Vinapamula, S., and Q. Wu, "A YANG Module for Network
Address Translation (NAT) and Network Prefix Translation
(NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019,

Boucadair, Ed., "RADIUS Attributes for Softwire Mechanisms
Based on Address plus Port (A+P)", RFC 8658,
DOI 10.17487/RFC8658, November 2019,

[RFC8683]  Palet Martinez, J., "Additional Deployment Guidelines for
NAT64/464XLAT in Operator and Enterprise Networks",
RFC 8683, DOI 10.17487/RFC8683, November 2019,
9.2. Informative References


Appendix A. Change Log

A.1. 01 - 02

- Ian Farrer has joined us as an author.
- Restructuring: the description of the five IPv4aaS technologies was moved to a separate section.
- More details and figures were added to the description of the five IPv4aaS technologies.
- Section titled "High-level Architectures and their Consequences" has been completely rewritten.
o Several additions/clarification throughout Section titled "Detailed Analysis".

o Section titled "Performance Analysis" was dropped due to lack of results yet.

o Word based text ported to XML.

o Further text cleanups, added text on state sync and load balancing. Additional comments inline that should be considered for future updates.

A.2. 02 - 03

o The suggestions of Mohamed Boucadair are incorporated.

o New considerations regarding possible optimizations.

A.3. 03 - 04

o Section titled "Performance Analysis" was added. It mentions our new benchmarking tool, siitperf, and highlights our plans.

o Some references were updated or added.

A.4. 04 - 05

o Some references were updated or added.

A.5. 05 - 06

o Some references were updated or added.

A.6. 06 - 00-WG Item

o Stats dated and added for Broadband deployments.

o Other clarifications and references.

o New section: IPv4 Pool Size.

o Typos.

A.7. 00 - 01

To facilitate WGLC, the unfinished parts were moved to two new drafts:
o New I-D for scale up measurements. (Including the results of iptables.)

o New I-D for benchmarking measurements. (Only a stub.)

Authors’ Addresses

Gabor Lencse
Budapest University of Technology and Economics
Magyar tudosok korutja 2.
Budapest H-1117
Hungary

Email: lencse@hit.bme.hu

Jordi Palet Martinez
The IPv6 Company
Molino de la Navata, 75
La Navata - Galapagar, Madrid 28420
Spain

Email: jordi.palet@theipv6company.com
URI: http://www.theipv6company.com/

Lee Howard
Retevia
9940 Main St., Suite 200
Fairfax, Virginia 22031
USA

Email: lee@asgard.org

Richard Patterson
Sky UK
1 Brick Lane
London EQ 6PU
United Kingdom

Email: richard.patterson@sky.uk
Scalability of IPv6 Transition Technologies for IPv4aaS
draft-lencse-v6ops-transition-scalability-00

Abstract

Several IPv6 transition technologies have been developed to provide customers with IPv4-as-a-Service (IPv4aaS) for ISPs with an IPv6-only access and/or core network. All these technologies have their advantages and disadvantages, and depending on existing topology, skills, strategy and other preferences, one of these technologies may be the most appropriate solution for a network operator.

This document examines the scalability of the five most prominent IPv4aaS technologies (464XLAT, Dual Stack Lite, Lightweight 4over6, MAP-E, MAP-T) considering two aspects: (1) how their performance scales up with the number of CPU cores, (2) how their performance degrades, when the number of concurrent sessions is increased until hardware limit is reached.

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 19 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

IETF has standardized several IPv6 transition technologies [LEN2019] and occupied a neutral position trusting the selection of the most appropriate ones to the market. [I-D.ietf-v6ops-transition-comparison] provides a comprehensive comparative analysis of the five most prominent IPv4aaS technologies to assist operators with this problem. This document adds one more detail: measurement data regarding the scalability of the examined IPv4aaS technologies.

Currently, this document contains only the scalability measurements of the iptables stateful NAT44 implementation. It serves as a sample to test if the disclosed results are (1) useful and (2) sufficient for the network operators.
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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Scalability of iptables

2.1. Measurement Method

[RFC8219] has defined a benchmarking methodology for IPv6 transition technologies. [I-D.lencse-bmwg-benchmarking-stateful] has amended it by addressing how to benchmark stateful NATxy gateways using pseudorandom port numbers recommended by [RFC4814]. It has defined a measurement procedure for maximum connection establishment rate and reused the classic measurement procedures like throughput, latency, frame loss rate, etc. from [RFC8219]. We used two of them: maximum connection establishment rate and throughput to characterize the performance of the examined system.

The scalability of iptables is examined in two aspects:

* How its performance scales up with the number of CPU cores?

* How its performance degrades, when the number of concurrent sessions is increased?
The test setup in Figure 1 was followed. The two devices, the Tester and the DUT (Device Under Test), were both Dell PowerEdge R430 servers having two 2.1GHz Intel Xeon E5-2683 v4 CPUs, 384GB 2400MHz DDR4 RAM and Intel 10G dual port X540 network adapters. The NICs of the servers were interconnected by direct cables, and the CPU clock frequency was set to fixed 2.1 GHz on both servers. They had Debian 9.13 Linux operating system with 4.9.0-16-amd64 kernel. The measurements were performed by siitperf [LEN2021] using the "stateful" branch (latest commit Aug. 16, 2021). The DPDK version was 16.11.11-1+deb9u2. The version of iptables was 1.6.0.

The ratio of number of connections in the connection tracking table and the value of the hashsize parameter of iptables significantly influences its performance. Although the default setting is hashsize=nf_conntrack_max/8, we have usually set hashsize=nf_conntrack_max to increase the performance of iptables, which was crucial, when high number of connections were used, because then the execution time of the tests was dominated by the preliminary phase, when several hundreds of millions connections had to be established. (In some cases, we had to use different settings due to memory limitations. The tables presenting the results always contain these parameters.)

The size of the port number pool is an important parameter of the benchmarking method for stateful NATxy gateways, thus it is also given for all tests.

2.2. Performance scale up against the number of CPU cores

To examine how the performance of iptables scales up with the number of CPU cores, the number of active CPU cores was set to 1, 2, 4, 8, 16 using the "maxcpus=" kernel parameter.

The number of connections was always 4,000,000 using 4,000 different source port numbers and 1,000 different destination port numbers. Both the connection tracking table size and the hash table size was set to 2^23.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times.

Besides the connection establishment rate and the throughput of iptables, also the throughput of the IPv4 packet forwarding of the Linux kernel was measured to provide a basis for comparison.
The results are presented in Figure 2. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

<table>
<thead>
<tr>
<th>num. CPU cores</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>src ports</td>
<td>4,000</td>
<td>4,000</td>
<td>4,000</td>
<td>4,000</td>
<td>4,000</td>
</tr>
<tr>
<td>dst ports</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>num. conn.</td>
<td>4,000,000</td>
<td>4,000,000</td>
<td>4,000,000</td>
<td>4,000,000</td>
<td>4,000,000</td>
</tr>
<tr>
<td>conntrack t. s.</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
</tr>
<tr>
<td>hash table size</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
<td>$2^{23}$</td>
</tr>
<tr>
<td>c.t.s/num.conn.</td>
<td>2.097</td>
<td>2.097</td>
<td>2.097</td>
<td>2.097</td>
<td>2.097</td>
</tr>
<tr>
<td>num. experiments</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>error</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>cps median</td>
<td>223.5</td>
<td>371.1</td>
<td>708.7</td>
<td>1,341</td>
<td>2,383</td>
</tr>
<tr>
<td>cps min</td>
<td>221.6</td>
<td>367.7</td>
<td>701.7</td>
<td>1,325</td>
<td>2,304</td>
</tr>
<tr>
<td>cps max</td>
<td>226.7</td>
<td>375.9</td>
<td>723.6</td>
<td>1,376</td>
<td>2,417</td>
</tr>
<tr>
<td>cps rel. scale up</td>
<td>1</td>
<td>0.830</td>
<td>0.793</td>
<td>0.750</td>
<td>0.666</td>
</tr>
<tr>
<td>throughput median</td>
<td>414.9</td>
<td>742.3</td>
<td>1,379</td>
<td>2,336</td>
<td>4,557</td>
</tr>
<tr>
<td>throughput min</td>
<td>413.9</td>
<td>740.6</td>
<td>1,373</td>
<td>2,311</td>
<td>4,436</td>
</tr>
<tr>
<td>throughput max</td>
<td>416.1</td>
<td>746.9</td>
<td>1,395</td>
<td>2,361</td>
<td>4,627</td>
</tr>
<tr>
<td>tp. rel. scale up</td>
<td>1</td>
<td>0.895</td>
<td>0.831</td>
<td>0.704</td>
<td>0.686</td>
</tr>
</tbody>
</table>

IPv4 packet forwarding (using the same port number ranges)

| error      | 200 | 500 | 1,000 | 1,000 | 1,000 |
| throughput median | 910.9 | 1,523 | 3,016 | 5,920 | 11,561 |
| throughput min | 874.8 | 1,485 | 2,951 | 5,811 | 10,998 |
| throughput max | 914.3 | 1,534 | 3,037 | 5,940 | 11,627 |
| tp. rel. scale up | 1 | 0.836 | 0.828 | 0.812 | 0.793 |
| throughput ratio (%) | 45.5 | 48.8 | 45.7 | 39.5 | 39.4 |

Figure 2: Scale up of iptables against the number of CPU cores

Whereas the throughput of IPv4 packet forwarding scaled up from 0.91Mpps to 11.56Mpps showing a relative scale up of 0.793, the throughput of iptables scaled up from 414.9kpps to 4,557kpps showing a relative scale up of 0.686 (and the relative scale up of the maximum connection establishment rate is only 0.666). On the one hand, this is the price of the stateful operation. On the other hand, this result is quite good compared to the scale-up results of NSD (a high performance authoritative DNS server) presented in Table 9 of [LEN2020], which is only 0.52. (1,454,661/177,432=8.2-fold performance using 16 cores.) And DNS is not a stateful technology.
2.3. Performance degradation caused by the number of sessions

To examine how the performance of iptables degrades with the number of connections in the connection tracking table, the number of connections was increased fourfold by doubling the size of both the source port number range and the destination port number range. Both the connection tracking table size and the hash table size was also increased four fold. However, we reached the limits of the hardware at 400,000,000 connections: we could not set the size of the hash table to $2^{29}$ but only to $2^{28}$. The same value was used at 800,000,000 connections too, when the number of connections was only doubled, because 1.6 billion connections would not fit into the memory.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times (except for the very long lasting measurements with 800,000,000 connections).

The results are presented in Figure 3. The unit for the maximum connection establishment rate is 1,000,000 connections per second. The unit for throughput is 1,000,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

<table>
<thead>
<tr>
<th>num. conn.</th>
<th>1.56M</th>
<th>6.25M</th>
<th>25M</th>
<th>100M</th>
<th>400M</th>
<th>800M</th>
</tr>
</thead>
<tbody>
<tr>
<td>src ports</td>
<td>2,500</td>
<td>5,000</td>
<td>10,000</td>
<td>20,000</td>
<td>40,000</td>
<td>40,000</td>
</tr>
<tr>
<td>dst ports</td>
<td>625</td>
<td>1,250</td>
<td>2,500</td>
<td>5,000</td>
<td>10,000</td>
<td>20,000</td>
</tr>
<tr>
<td>conntrack t. s.</td>
<td>$2^{21}$</td>
<td>$2^{23}$</td>
<td>$2^{25}$</td>
<td>$2^{27}$</td>
<td>$2^{29}$</td>
<td>$2^{30}$</td>
</tr>
<tr>
<td>hash table size</td>
<td>$2^{21}$</td>
<td>$2^{23}$</td>
<td>$2^{25}$</td>
<td>$2^{27}$</td>
<td>$2^{28}$</td>
<td>$2^{28}$</td>
</tr>
<tr>
<td>num. exp.</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>error</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>n.c./h.t.s.</td>
<td>0.745</td>
<td>0.745</td>
<td>0.745</td>
<td>0.745</td>
<td>1.490</td>
<td>2.980</td>
</tr>
<tr>
<td>cps median</td>
<td>2.406</td>
<td>2.279</td>
<td>2.278</td>
<td>2.237</td>
<td>2.013</td>
<td>1.405</td>
</tr>
<tr>
<td>cps min</td>
<td>2.358</td>
<td>2.226</td>
<td>2.226</td>
<td>2.124</td>
<td>1.983</td>
<td>1.390</td>
</tr>
<tr>
<td>cps max</td>
<td>2.505</td>
<td>2.315</td>
<td>2.317</td>
<td>2.290</td>
<td>2.050</td>
<td>1.440</td>
</tr>
<tr>
<td>throughput med.</td>
<td>5.326</td>
<td>4.369</td>
<td>4.510</td>
<td>4.516</td>
<td>4.244</td>
<td>3.689</td>
</tr>
<tr>
<td>throughput max</td>
<td>5.533</td>
<td>4.408</td>
<td>4.572</td>
<td>4.537</td>
<td>4.342</td>
<td>3.709</td>
</tr>
</tbody>
</table>
The performance of iptables shows degradation at 6.25M connections compared to 1.56M connections very likely due to the exhaustion of the L3 cache of the CPU of the DUT. Then the performance of iptables is fairly constant up to 100M connections. A small performance decrease can be observed at 400M connections due to the lower hash table size. A more significant performance decrease can be observed at 800M connections. It is caused by two factors:

* on average, about 3 connections were hashed to the same place
* non NUMA local memory was also used.

We note that the CPU has 2 NUMA nodes, cores 0, 2, ... 14 belong to NUMA node 0, and cores 1, 3, ... 15 belong to NUMA node 1. The maximum memory consumption with 400,000,000 connections was below 150GB, thus it could be stored in NUMA local memory.

Therefore, we have pointed out important limitations of the stateful NAT44 technology:

* there is a performance decrease, when approaching hardware limits
* there is a hardware limit, beyond which the system cannot handle the connections at all (e.g. 1600M connections would not fit into the memory).

Therefore, we can conclude that, on the one hand, a well tailored hashing may guarantee an excellent scale-up of stateful NAT44 regarding the number of connections in a wide range, however, on the other hand, stateful operation has its limits resulting both in performance decrease, when approaching hardware limits and also in inability to handle more sessions, when reaching the memory limits.

3. Acknowledgements

The measurements were carried out by remotely using the resources of NICT StarBED, 2-12 Asahidai, Nomi-City, Ishikawa 923-1211, Japan. The author would like to thank Shuhei Takimoto for the possibility to use StarBED, as well as to Satoru Gonno and Makoto Yoshida for their help and advice in StarBED usage related issues.

The author would like to thank Ole Troan for his comments on the v6ops mailing list, while the scalalability measurements of iptables were intended to be a part of [I-D.ietf-v6ops-transition-comparison].
4. IANA Considerations

This document does not make any request to IANA.

5. Security Considerations

TBD.

6. References

6.1. Normative References


6.2. Informative References


Lencse Expires 19 April 2022
Appendix A.  Change Log

A.1.  00

Initial version: scale up of iptables.

Author’s Address

Gabor Lencse
Szechenyi Istvan University
Gyor
Egyetem ter 1.
H-9026
Hungary

Email: lencse@sze.hu
Abstract

Neighbor Discovery (ND) is an integral part of IPv6 first-hop. Due to limitation of certain L2 media’s support to ND, a number of issues can happen in certain scenarios. The corresponding solutions for these issues have also been designed. These issues and solutions are summarized in RFC3756, RFC6583, OPSECv27. However, there is no guideline on how to avoid the issues or how to select the proper solutions.

This document analyzes existing solutions and summarizes the wisdoms embedded in these solutions into an insight: isolating hosts in L2 and L3 can be effective in preventing ND issues. In deployment scenarios where the ND issues can occur, this prevention approach can be more effective than deploying various solutions to solve the issues. Based on this insight, a set of guidelines is proposed for future ND deployments. These guidelines describe where and when to isolate hosts in L2 and L3 to prevent ND issues, and how to select suitable solutions for the remaining issues. This will likely simplify ND deployments. The impact of the guidelines to other components of IPv6 first-hop is also analyzed and appears small. Therefore, the guidelines will likely simplify the overall IPv6 first-hop deployments.

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 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction...................................................3
   1.1. Terminology...............................................3
2. Summary of ND Issues and Existing Solutions....................4
   2.1. Multicast Issues and Solutions............................4
   2.2. DAD-unreliable Issues and Solutions.......................6
   2.3. On-demand NCE Installation Issues and Solutions..........6
   2.4. Security Issues and Solutions.............................7
   2.5. Observations on the Solutions and the Insight Learned.....9
3. Isolating Hosts in L2 and L3 to Simplify ND Deployments.......11
   3.1. Applicability of Host Isolation in L2 and L3.............11
   3.2. Deployment Guidelines....................................14
4. Impact of L2 and L3 Host Isolation to Other Components of IPv6
First-hop........................................................16
5. Applying the Guidelines to Real World Scenarios...............17
6. Security Considerations........................................19
7. IANA Considerations...........................................19
8. References....................................................19
   8.1. Informative References...................................19
9. Acknowledgments...............................................22

Xiao                   Expires January 12, 2022                [Page 2]
1. Introduction

ND is an integral part of IPv6 first-hop. Due to limitation of certain L2 media's support to the protocol, a number of issues have been discovered, and the corresponding solutions for these issues have also been designed. These issues and solutions are dispersed in more than 20 RFCs. [RFC6583] summarized the issues and solutions up to 2012. [OPSECv27] summarized known IPv6 security issues, including ND issues, up to 2021. Wireless ND (WiND) [RFC6775][RFC8505][RFC8928][RFC8929] discussed ND issues and solutions in Low-Power and Lossy Networks (LLNs) [RFC7102]. However, two ND deployment problems still exist:

1. Neither [RFC6583] nor [OPSECv27] or WiND provides guidelines on how to prevent the issues;
2. Some issues have multiple solutions. There is no guideline on how to select the suitable solutions depending on the usage scenarios.

[RFC8273] recommends using "Unique IPv6 Prefix per Host" as a best practice for ND. By doing so, certain ND issues can be prevented. So it partially solved Problem 1 above. But [RFC8273] cannot be used everywhere. In fact, some concerns about [RFC8273] have been expressed in [8273D].

This document aims to solve both problems by providing deployment guidelines for ND. Depending on the usage scenarios, the guidelines first try to prevent the issues as much as possible, then recommend suitable solutions for the remaining issues. This can result in the simplest ND solution for future usage scenarios.

1.1. Terminology

Familiarity with [ND] and [SLAAC] are pre-requisite to understand this document. Familiarity with the ND issues and solutions discussed in [RFC6583] and [OPSECv27] Section 2.3 are also essential to understand this document.

Some frequently used terms are defined in this section.

Host isolation in L2 - A host cannot send packets via the L2 medium to any other hosts. The router will be the only node reachable in L2. This can be realized on P2P media, or on multi-access media with Private VLAN [PVLAN] or Split Horizon [TR177] or wireless isolation [W-Iso] to prevent hosts from communicating with each other in L2.
Host isolation in L3 - Separate hosts into different subnets. It is also known as "unique prefix per host" [RFC8273].

NCE - Neighbor Cache Entry, a binding of a neighbor’s IPv6 address and link layer address.

2. Summary of ND Issues and Existing Solutions

This section summarizes the main ND issues and solutions. More information can be found in [RFC6583][OPSECv27][RFC6775].

2.1. Multicast Issues and Solutions

ND uses multicast (L3) extensively for Node Solicitations (NSs), Router Solicitations (RSs) and Router Advertisements (RAs). When multicast messages are sent over an L2 medium, they are usually mapped into L2 broadcast messages. For many wireless media, L2 broadcast may consume more network resources than multiple P2P unicast combined [RFC6775][RFC7668], and have lower probability to be delivered. In addition, multicast has no acknowledgement. Consequently, multicast may cause a number of issues:

. Multicast-resource-consumption issue: multicast in L3 and the resulted broadcast in L2 may consume many times more resources than unicast;
. Multicast-power-consumption issue: multicast consumes more power of the sender; In addition, an L2 broadcast message may reach more nodes than the L3 multicast intended. This may consume some receiving nodes’ power unnecessarily. For power constrained nodes, this is an issue;
. Multicast-reliability issue: some multicast messages may not reach the destinations. Sleeping nodes may not respond to a multicast message. As a result, protocol procedures that rely on multicast can be unreliable.

Existing ND solutions that can prevent or address multicast issues include:
Mobile Broadband IPv6 (MBBv6) and Fixed Broadband IPv6 (FBBv6): MBBv6 is defined in "IPv6 in 3GPP EPS" [RFC6459] and "IPv6 for 3GPP Cellular Hosts" [RFC7066]. MBBv6 isolates each host in L2 by putting it in a P2P link with the router. FBBv6 is defined in "IPv6 in the context of TR-101" [TR177]. FBBv6 also isolates each host in L2, either by putting it in a P2P link with the router, or by implementing split horizon on Ethernet to prevent direct host communication. Note that a host here is either a mobile User Equipment (UE), or a routed Residential Gateway (RG), or a real host (e.g. a computer) behind a bridged RG. The router here is either the mobile gateway or the Broadband Network Gateway (BNG). MBBv6 and FBBv6 also isolate each host in L3 by giving it a unique prefix and thus putting it in its own subnet. Consequently, every host can only communicate with the router in both L2 and L3, and there is effectively no multicast. Because in FBBv6, bridged RG situation is very similar to routed RG situation, except that in the former, the hosts are real hosts (e.g. computers), while in the latter, the hosts are the routed RGs. For description simplicity, bridged RGs will not be further discussed in the document.

Wireless ND (WiND): the solution is defined in a series of RFCs [RFC6775][RFC8505][RFC8928][RFC8929]. WiND changes host and router behaviors to use multicast only for router discovery and not for other protocol procedures. The key points are (1) hosts use unicast to proactively register their addresses at the routers; (2) routers use unicast to communicate with hosts, and become the central address register and arbitrator for the hosts. Router also proactively installs Neighbor Cache Entries (NCEs) for the hosts; (3) each host communicates only with the router. Consequently, multicast is largely eliminated.

Unique IPv6 Prefix per Host (UPPH) [RFC8273]: this solution reduces the number of hosts in each subnet to one, as a subnet is defined by a prefix. Multicast is greatly reduced, because (1) a host will not use multicast for address resolution of other hosts, because the router is the only L3 next-hop; (2) downstream multicast from router to hosts are eliminated and turned into unicast ([RFC8273] Section 4). The pros and cons of RFC8273 will be briefly reviewed in Section 3.1. An in-depth discussion can be found in [8273D].

IP Point to Point Ethernet Subnet Model [P2Peth]: if progressed, this draft may provide a solution similar to RFC8273, making use of unique prefix per host to reduce multicast.
2.2. DAD-unreliable Issues and Solutions

Duplicate Address Detection (DAD) uses absence of response as indication of no duplicate. This can be unreliable because either the Neighbor Solicitation or the Neighbor Advertisement messages may be lost in transmission, especially in wireless environment, or the sleeping nodes may not respond. Duplication thus may not be detected.

Existing ND solutions that can prevent or address DAD-unreliable issue include:

- MBBv6 and FBBv6: for MBBv6, the UE’s Interface ID (IID) is assigned by the mobile gateway, and is guaranteed to be unique [RFC6459]. There is no need for DAD. For FBBv6, the RG’s Global Unicast Address (GUA) prefix is unique. There will be no duplicate for GUA address, and no DAD will be performed. RG will perform DAD for its Link Local Address (LLA), but only to the BNG [TR177]. In such an environment, there is no sleeping node or lossy media. DAD has no reliability issue.

- WiND: every host must register its address at the router before using it. The registration will succeed only if router explicitly approves it, and router will not approve if it is a duplicate. Therefore, the DAD procedure becomes reliable.

- RFC8273: For GUA addresses, since each host is given a different prefix, duplicate address will not exist. For link local address, since each subnet has only one host, there is no possibility of duplication in the subnet. A duplicate link local address may exist in another subnet but that does not matter. Therefore, DAD-unreliable issue is prevented.

2.3. On-demand NCE Installation Issues and Solutions

ND address resolution is on-demand. It is done only when a packet needs to be sent but the link layer address of the on-link destination is unknown. Consequently, the packet has to be queued until link layer address of the destination is determined. This increases delay and may affect application performance. For high performance environment, this can be an issue.

Existing ND solutions that can prevent or address on-demand NCE installation issue include:
MBBv6 and FBBv6: MBBv6 and some FBBv6 use IPv6 over PPP [RFC5072]. In this case, there is no need to find out the link layer address before sending a packet on the interface. In other words, there is no NCE installation issue. Some FBBv6 implementations use IPoE. There is a need for NCE. But because every host is given a unique prefix by the BNG in FBBv6, the BNG needs to know the host’s link layer address which uniquely identify the host in order to do so. Therefore, the BNG can install NCE when assigning a prefix to the host, not waiting until a data packet is to be sent to that host. On-demand NCE installation issue is therefore avoided in this case too.

Wireless ND (WiND): when hosts register their IPv6 addresses, they will also present their link layer address. Therefore, NCE entries can be installed proactively before data packets arrive.

Gratuitous Neighbor Discovery [GRAND]: this solution changes router and host behaviors to allow routers to proactively create an NCE when a new IPv6 address is assigned to a host, and to recommend hosts sending unsolicited Neighbor Advertisements upon assigning a new IPv6 address. It can be considered as the IPv6 equivalent of Gratuitous ARP in IPv4.

2.4. Security Issues and Solutions

ND assumes all nodes can be trusted. Otherwise, ND has various security issues. These issues are described in [RFC3756][RFC6583] and [OPSECv27]. They are briefly summarized here.

The security issues can be classified into two categories:

- Redirect attacks, in which a malicious node redirects packets away from the first-hop router or other legitimate receiver to another node on the link. This is usually done by spoofing the source IPv6 address to pretend to be another node, or by faking Router Advertisement to pretend to be a router. For example:
  
  - an attacker can send out Router Advertisement claiming itself as the router, and redirect other hosts’ traffic to itself.
o an attacker can send Neighbor Advertisements to the first-
  hop router, spoofing the IPv6 address of another node
  while using its own link layer address. This will redirect
  traffic from the router to the victim to the attacker
  instead.

. Denial-of-Service (DoS) attacks, in which a malicious node
  prevents communication between the node under attack and all
  other nodes or a specific destination address. For example:

  o Whenever a host performs DAD, an attacker can reply to
    claim that the selected address is in use. The host will
    not be able to get an address.

  o /64 scan attacks: an attacker sends packets to up to 2**64
    mostly non-existing hosts, forcing the first-hop router to
    try to install NCEs for this huge number of non-existing
    hosts. If unprotected, the router will run out of resource
    and stop functioning. Note that for other attacks to
    happen, the attacker must be on-link. But for this /64
    scan attack, the attacker can be off-link.

It is worth noting that redirect attacks are generally considered as
more harmful than DoS attacks. Therefore, higher priority is given
to protect against redirect attacks.

Existing ND solutions that can prevent or address the security
issues include:

. MBBv6 and FBBv6: because every host is isolated in L2, all on-
  link security issues are prevented. Because every host has its
  own prefix, and the mobile gateway or BNG maintains state
  information for the prefixes not for individual IPv6 address,
  off-link /64 scan attack will not cause harm because the mobile
  gateway or BNG will not create any NCE when receiving such
  messages. Such attack messages can simply be dropped.

. WiND: normally, every host is isolated in L2 and can only
  communicate with the first-hop router. Therefore, all on-link
  security issues are prevented. But if hosts are not isolated in
  L2, e.g. when there is a bridge between the hosts and the
  router, then on-link security issue can happen. Off-link /64
  scan attack will not cause harm because in WiND, NCE
  installation is proactive not reactive. In other words, the
  router will not create any NCE when receiving such messages.
  Such attack messages can simply be dropped.
RFC8273: by giving each host a different prefix and keep track of host-prefix binding, an attacker host cannot change the NCE at the router for another host by sending Neighbor Advertisement with a spoofed source IPv6 address of that host. The attacker thus cannot redirect traffic from router to that host to itself. The router will maintain only one NCE entry for each prefix/host. Therefore, off-link /64 scan attack will not cause harm. RFC8273’s protection effect against other security issues depends on whether the hosts are also isolated in L2. If yes, all the on-link security issues will be prevented. If not, certain on-link security issues remain.

Source Address Validation Improvement [SAVI], Router Advertisement Guard [RA-Guard][RA-Guard+]: these solutions protect against redirect attacks and some DoS attacks. SAVI binds an address to a port, and rejects claims from other ports for that address. Therefore, a node cannot spoof the IP address of another node. RA-Guard and RA-Guard+ only allow RAs from a port that a router is connected. Therefore, nodes on other ports cannot pretend to be a router. In order to protect against some other DoS attacks, e.g. off-link /64 scan attack, other security measures are needed, e.g. rate limiting on NSs, and filtering on NAs/RAs.

Secure Neighbor Discovery [SeND] and Cryptographically Generated Addresses [CGA]: these solutions have tried to make ND secure, but have not been widely deployed. They will not be further discussed in this document.

2.5. Observations on the Solutions and the Insight Learned

MBBv6 and FBBv6 prevent or solve all the ND issues. These solutions have two common characteristics that are effective for preventing or solving ND issues:

1. Hosts (i.e. UEs or RGs or real hosts) are isolated in both L2 and L3.
2. The first-hop router (i.e. mobile gateway or BNG) maintains some state information across reboots, e.g. prefix to host binding. The router also maintains some controls over the hosts, e.g. which prefix each host gets.

WiND prevents or solves all the ND issues in its deployment scenarios, e.g. Low power and Lossy Networks (LLNs) [RFC7102]. WiND also has two characteristics:
Hosts are normally isolated in L2 but not in L3. In fact, many hosts are in the same subnet. In the rare cases where hosts are not isolated in L2, e.g. because there is an bridge between the hosts and the router, then some ND issues like on-link security issues may happen, and additional solutions will be needed to address those issues.

The first-hop router maintains some state information across reboots, e.g. host’s address registration result including IPv6 address to link layer address binding. The router also uses such state information to maintain some controls over the hosts, e.g. which host wins when there is an address contention.

RFC8273 solves certain ND issues but not all. It has two characteristics:

(1) Hosts are isolated in L3 in their own subnet, but RFC8273 does not specify whether hosts should also be isolated in L2. If hosts are also isolated in L2, all ND issues will be prevented.

(2) The first-hop router maintains some state information across reboots, e.g. prefix-host binding. The router also maintains some controls over the hosts, e.g. which prefix a host gets.

SAVI, RA-Guard, RA-Guard+ and GRAND solve specific ND issues. They have two characteristics:

(1) These solution make no assumptions on isolation of hosts in L2 or L3. They just solve the ND issues that they are designed to solve.

(2) SAVI maintains some state information at the Ethernet switch between the hosts and the first-hop router(s). Such states are learned dynamically from packet snooping, or configured manually. The Ethernet switch uses such state information to control the hosts, e.g., RAs are not allowed from the hosts.

An insight can be learned from observing ND practices in these existing IPv6 first-hop deployments. That is, isolating hosts in L2 and L3 can be effective in preventing ND issues. But isolating hosts in L2 and L3 also has a price to pay. That is, because hosts are isolated and cannot directly coordinate with each other, the router must have new functionality to coordinate on behalf of the hosts, e.g. to be the arbitrator when there is an address contention. In order to do so, the router will also need to maintain some state information, e.g. IP address-link layer address binding for each host. This insight can be used to guide future ND deployments.
3. Isolating Hosts in L2 and L3 to Simplify ND Deployments

3.1. Applicability of Host Isolation in L2 and L3

This section describes how to isolate hosts in L2 and L3, and the advantages and disadvantages of doing so.

Isolating hosts in L2 can be done by: (1) putting a host in a P2P link with the router, or (2) using private VLAN [PVLAN], or split horizon [TR177], or wireless isolation [W-Iso] on multi-access medium to prevent hosts from communicating with each other. These are a matter of device configuration so it can usually be done as long as the devices support these isolating features.

Isolating hosts in L2 is different from setting ND Prefix Information Option (PIO) L-bit=0. In the former, multicast messages from a host will not reach other hosts in the same L2 broadcast domain. In the latter, a host will not do address resolution for other hosts with that prefix, but multicast messages from the host will be mapped into L2 broadcast, and will reach other hosts in the same L2 broadcast domain.

When hosts are isolated in L2, DAD messages can only reach the router but not other hosts. Therefore, the router must act to prevent hosts from using duplicate GUA or link local address. This functionality is called DAD Proxy [TR177][RFC6957].

The advantages of isolating hosts in L2 include:

- Prevention of on-link security and upstream multicast issues (from hosts), because hosts cannot reach each other directly.

The disadvantage of isolating hosts in L2 include:

- The router must support additional functionality: DAD Proxy.
- Downstream multicast issues, DAD-unreliable issue, On-demand NCE Installation issue and off-link /64 scan issue still remain.
- All host communication must go through the router. In a high performance environment, the router may become the bottleneck.
- Services relying on multicast, e.g. Multicast DNS [mDNS], will not work, unless the router can provide multicast proxy.
There is additional work (e.g. PVLAN, split horizon or wireless isolation) to isolate hosts in L2 in multi-access media, but this is small amount of work.

From the analysis above, it is clear that L2 isolation alone is not advantageous. The benefits are relatively small while the costs are relatively high. Therefore, L2 isolation is better used with something else, e.g. L3 isolation or some specially designed solutions like WiND.

The known solutions supporting host isolation in L2 include WiND, and in more special cases (i.e. with both L2 and L3 isolation), MBBv6 and FBBv6.

Host isolation in L3 (i.e. Unique Prefix Per Host, or UPPH) can be done either with [DHCPv6], where the prefix can be of any length supported by DHCPv6, or with SLAAC as specified in RFC8273, where the prefix must be /64 or shorter. If [P2Peth] is progressed, it can provide another solution.

In order to isolate hosts in L3, the router must be able to assign a unique prefix to each host and keep track of the prefix-host link layer address binding. This will be referred to as "UPPH support" later in this document. This may be achieved by some mechanism that RFC8273 alluded to but did not specify, or by some other mechanisms if DHCPv6 is used [TR177]. Note that with SLAAC/RFC8273, such router implementation exist [8273D], while with DHCPv6, FBBv6-capable BNG is one of such implementations.

The advantages of isolating hosts in L3 are:

- Downstream Multicast, DAD-unreliable, on-demand NCE installation and off-link /64 scan issues are prevented.
- It is the only way for the router to do subscriber management on the hosts in a SLAAC environment. Imagine a public Wi-Fi scenario where the mobile UEs only support SLAAC. If the router does not give each UE a unique prefix and keep track of UE-prefix binding, network administrators do not know which IPv6 address corresponds to which UE, because each UE picks its own IID and uses the same prefix. Therefore, network administrators cannot keep track of which UE did what. This would be unacceptable from an operation perspective.

The disadvantages of isolating hosts in L3 are:
o The routers must support new functionality: "UPPH support".

o Upstream multicast and on-link security issues can happen, unless the hosts are also isolated in L2.

o Many prefixes will be needed instead of just one. But this disadvantage may not be as severe as it appears. After all, 3GPP has given every mobile UE a /64 [RFC6459], and BBF has given every routed RG a /56 with DHCPv6 PD [TR177]. Outside of MBB, FBB and IoT, not many scenarios have a large number of hosts. Giving each host a /64 prefix may not be as deficient as it appears.

The known solutions supporting host isolation in L3 include RFC8273, and in more special cases, MBBv6 and FBBv6.

Careful analysis of the advantages and disadvantages of L2 isolation and L3 isolation will reveal that they are synergetic. When they are done together, the advantages are:

o All ND issues are prevented.

o It provides a way for the router to do subscriber management on the hosts in a SLAAC environment.

o DAD Proxy needed for L2 isolation is no longer needed, because with unique prefix per host, GUA cannot be duplicate. For link local address, since each subnet has only one host, there is no possibility of duplication. A duplicate link local address may exist in another subnet but that does not matter. Therefore, there is no need for DAD Proxy to prevent duplicate GUA/LLA addresses.

The disadvantages are:

o The routers must support new functionality: "UPPH support".

o Many prefixes will be needed instead of just one.

o All host communication must go through the router. In a high performance environment, the router may become the bottleneck.

o Services relying on multicast, e.g. mDNS, will not work, unless the router can provide multicast proxy.
There is additional work (e.g. PVLAN, split horizon or wireless isolation) to isolate hosts in L2 in multi-access media, but this is small amount of work.

The known solutions supporting host isolation in L2 and L3 include RFC8273, MBBv6 and FBBv6.

3.2. Deployment Guidelines

Given the applicability analysis in Section 3.1, network administrators can decide where to use which isolation option. Note that in all the following steps, if DHCPv6 (IA_NA or PD) is supported and desired, use DHCPv6 to assign address prefix, as it may provide more prefix length flexibility. Otherwise, use SLAAC as SLAAC is more widely supported.

1. If isolating hosts in both L2 and L3 is desirable and feasible:

   Based on the applicability discussion in Section 3.1, the scenarios here likely have some or all of the following characteristics: (1) It is a public access environment where subscriber management is needed; (2) Many ND issues can happen and will require solutions. This is why it is desirable to prevent as many issues as possible, to simplify the deployment; (3) Neither high performance communication nor multicast service discovery is applicable here.

   With suitable "UPPH support", all the ND issues will be prevented. Some possible "UPPH support" solutions are:

   a) If the deployment scenario is MBB or FBB, then MBBv6 or FBBv6 can be used.

   b) Otherwise, use a RFC8273 implementation (using SLAAC) to realize "UPPH support". Note that this is a special use case of RFC8273 where each host is not only given a unique prefix, but also isolated from other hosts in L2.

2. Otherwise, if isolating hosts in L2 but not in L3 is considered appropriate:

   In this scenario, hosts are in different links but in the same subnet. This architecture is called Multi-Link SubNet (MLSN). So far only WiND and [OMNI] support MLSN. But OMNI is not widely supported so WiND is the only solution available. Therefore, the scenario here is most likely one that WiND is suitable for, e.g.
Low-power and Lossy Networks (LLNs). Because WiND has more functionalities than DAD Proxy (required by L2 host isolation), e.g. proactively address registration, WiND prevents all ND issues.

3. Otherwise, if isolating hosts in L3 but not in L2 is considered appropriate:

Based on the applicability discussion in Section 3.1, the scenarios here likely have the following characteristics: (1) Subscriber management is needed. This is likely a public access scenario. (2) Upstream multicast is needed or cannot be avoided (otherwise L2 host isolation would have been selected to prevent more issues), e.g. for service discovery. RFC8273 can be used as the solution here. On-link security is not prevented in this case. Because this is a public access scenario, SAVI/RA-Guard/RA-Guard+ may be needed to address the on-link security issue.

4. Otherwise, no isolation in L2 or L3 is desired or feasible.

The scenarios here likely have the following characteristics: (1) It is a private environment, because subscriber management is not needed. As such, on-link security is not a big concern. (2) Either multicast service is needed, or this is a high performance scenario, because L2 host isolation is not desired. Either way, multicast and DAD-unreliable are not a concern. Normal ND can be used as the solution. Off-link /64 scan and on-demand NCE installation are the two issues left. Off-link /64 scan issue can be handled by rate limiting or unused-address filtering. If this is indeed a high performance environment, e.g. a DC network, GRAND can be used to enable proactive NCE installation.

In short, the guidelines can be summarized as: if many of the ND issues discussed in Section 3 are valid concerns and need to be addressed, then isolate hosts in L2 and L3 to prevent the issues, and use RFC8273. If the scenario is LLN, use WiND. But if the scenario is a private environment where many ND issues are not real concerns, then just use normal ND, and adds other solutions like GRAND only when needed. Overall, these guidelines will likely result in the simplest ND solution.
4. Impact of L2 and L3 Host Isolation to Other Components of IPv6 First-hop

The guidelines should simplify ND deployments. But will they complicate other components of IPv6 first-hop? A preliminary impact analysis is done in this section.

IPv6 first-hop consists of:
- The routers and the hosts, whose requirements & behaviors are defined in [RFC8504];
- Addressing scheme;
- The basic protocol suite: ND, SLAAC, DHCPv6, DNS, ICMPv6, MLD v1/v2;
- Other extended protocols: mDNS.

The impact of host isolation in L2 and L3 to other components of IPv6 first-hop comes from three parts: (1) the special topology introduced by L2 host isolation; (2) the special addressing scheme introduced by L3 host isolation (i.e. unique prefix per host); (3) the new functionalities introduced to the router, i.e. "DAD Proxy" for L2 isolation and "UPPH support" for L3 isolation. (4) When WiND is used, hosts must be changed to support proactive address registration etc. This is a high requirement that can be considered as complicating the IPv6 first-hops. But WiND is the only solution in its applicable scenarios like LLNs. When there is no alternative, there is no need to discuss whether the solution unduly complicates other components of IPv6 first-hop. Therefore, WiND will not be further discussed. Because DAD Proxy is only needed in L2 but not L3 isolation, and this scenario uses WiND which already provides DAD Proxy, DAD Proxy will not be further discussed either.

First, regarding the impact from the special topology:

Isolating hosts should only affect the protocols that rely on multicast, i.e. ND, SLAAC and mDNS, but not the multicast protocols themselves, i.e. MLD v1/v2. As discussed in Sections 2 and 3, the impact on ND and SLAAC are positive. The impact on mDNS can be negative, if new functionality like "multicast proxy at router" has to be introduced. But the guidelines give network administrators the option not to isolate hosts at all. So if network administrators choose to isolate, the benefit must outweigh the cost.
Second, regarding the impact from the special addressing scheme:

IPv6 first-hop should allow any addressing scheme. Other than using more prefixes, it is not clear that unique prefix per host complicates addressing scheme in any way. After all, unique prefix per host is already widely in use in MBBv6 and FBBv6 deployments.

Third, regarding the impact of new router functionality "UPPH support":

The impact of "UPPH support" with RFC8273 using SLAAC has been discussed in [8273D] before RFC8273 became an RFC. The following concerns had been raised:

1. RFC8273 makes the router stateful;
2. How to reclaim unused prefix is not specified;
3. If there are multiple first-hop routers, how the solution works is unspecified: (1) do they assign prefixes to hosts independently? (2) do they need to sync up with each other?
4. Resiliency of SLAAC may be reduced as a result of the increased state at the router, i.e. the prefix-host binding.

Given that RFC8273 became an RFC, the rough consensus may have been its benefit outweighs the cost. Because MBBv6 and FBBv6 also support UPPH and are widely deployed, the impact of "UPPH support" is likely manageable.

All things considered, it appears that isolating hosts in L2 and L3 can simplify ND without unduly complicating other components of IPv6 first-hop.

5. Applying the Guidelines to Real World Scenarios

The guidelines are intended for future IPv6 first-hop deployments. But if we test the guidelines on well-known IPv6 scenarios to find the solutions, the results will be as follows:

- MBB and FBB will end at Step 1.a: isolating hosts in L2 and L3, with MBBv6 as the solution with SLAAC and FBBv6 as the solution with DHCPv6, respectively;
o Public Wi-Fi network will end at Step 1.b: isolating hosts in L2 and L3, with RFC8273 as the solution with SLAAC, since the hosts here may be mobile UEs that do not support DHCPv6. Note that in Wi-Fi with the Infrastructure Mode [WiFi-inf], each host (i.e. STA) communicates only with the Access Point (AP). With wireless isolation, every host can only communicate with the AP (and the router if the AP is not a router), not directly with other hosts. This is how L2 isolation can be achieved in this scenario. If L2 isolation is not done, then public Wi-Fi will end at Step 3. SAVI/RA-Guard/RA-Guard+ may be needed to address the on-link security issue.

o Low-power and Lossy Networks (LLNs) will end at Step 2: isolating hosts in L2 but not L3, with WiND as the solution with SLAAC. Note that although WiND did not mandate L2 isolation, WiND works better when hosts are isolated in L2. Otherwise, additional mechanisms may be needed to address on-link security issues.

o High speed DC Networks will end at Step 4: using normal ND with GRAND without host isolation in L2 or L3. SAVI, RA-Guard and RA-Guard+ may not be needed as high physical access security is likely maintained.

o Common enterprise LANs with mixed Ethernet and Wi-Fi (not DC networks) will end at:
  • If security is a concern to justify host isolations:
    . Step 1.b: isolating hosts in L2 and L3, with RFC8273 as the solution with SLAAC;
  • If security is not a concern to justify host isolations:
    . Step 4: using normal ND with no special host isolation. GRAND and SAVI, RA-Guard and RA-Guard+ may not be needed.

o [HomeNet] will end at Step 4: using normal ND with no special host isolation. GRAND and SAVI, RA-Guard and RA-Guard+ are not needed.

These results match current practices in reality. This validates the guidelines to some extent.
6. Security Considerations

This document provides guidelines on how and where to isolate hosts in L2 and L3 to prevent ND issues, and how to select existing solutions for the remaining issues. When a solution is selected, the security considerations of that solution apply. This document does not introduce any new mechanisms. Therefore, it does not introduce new security issues.

7. IANA Considerations

This document has no request to IANA.

8. References

8.1. Informative References

[8273D] Discussion on the pros and cons of RFC8273, https://mailarchive.ietf.org/arch/msg/v6ops/M471N8lyXaWVcx
8nivkxMfBNA/

[CGA] T. Aura, "Cryptographically Generated Addresses (CGA)", RFC3972


S. Varada, D. Haskins, E. Allen, "IP Version 6 over PPP", RFC 5072


Z. Shelby, S. Chakrabarti, E. Nordmark, C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775.

F. Costa, J-M. Combes, X. Pougard, H. Li, "Duplicate Address Detection Proxy", RFC 6957


[SeND] J. Arkko, J. Kempf, B. Zill, P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC3971

9. Acknowledgments

The authors would like to thank Eduard Vasilenko, Pascal Thubert, and Ole Troan for the discussion and input.

Authors’ Addresses

XiPeng Xiao
Huawei Technologies
Hansaallee 205, 40549 Dusseldorf, Germany
Email: xipengxiao@huawei.com

Eduard Metz
KPN N.V.
Maanplein 55, 2516CK The Hague, The Netherlands
Email: eduard.metz@kpn.com

Gyan Mishra
Verizon Inc.
Email: gyan.s.mishra@verizon.com


