

intarea
Internet-Draft
Intended status: Informational
Expires: January 29, 2018

P. Pfister, Ed.
Cisco
D. Schinazi
T. Pauly
Apple
E. Vyncke
Cisco
B. Bruneau
Ecole Polytechnique
July 28, 2017

Discovering Provisioning Domain Names and Data
draft-bruneau-intarea-provisioning-domains-02

Abstract

An increasing number of hosts and networks are connected to the Internet through multiple interfaces, some of which may provide multiple ways to access the internet by the mean of multiple IPv6 prefix configurations.

This document describes a way for hosts to retrieve additional information about their network access characteristics. The set of configuration items required to access the Internet is called a Provisioning Domain (PvD) and is identified by a Fully Qualified Domain Name (FQDN). This identifier, retrieved using a new Router Advertisement (RA) option, is associated with the set of information included within the RA and may later be used to retrieve additional information associated with the PvD by the mean of an HTTP request.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 29, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. Provisioning Domain Identification using Router Advertisements	4
3.1. PvD ID Option for Router Advertisements	4
3.2. Router Behavior	5
3.3. Host Behavior	5
3.3.1. DHCPv6 configuration association	6
3.3.2. DHCPv4 configuration association	7
3.3.3. Interconnection Sharing by the Host	7
4. Provisioning Domain Additional Information	7
4.1. Retrieving the PvD Additional Information	7
4.2. Providing the PvD Additional Information	9
4.3. PvD Additional Information Format	9
4.3.1. Connectivity Characteristics Information	10
4.3.2. Private Extensions	11
4.3.3. Example	11
5. Security Considerations	11
6. Privacy Considerations	12
7. IANA Considerations	12
8. Acknowledgements	12
9. References	13
9.1. Normative references	13
9.2. Informative references	13
Appendix A. Changelog	15
A.1. Version 00	15
A.2. Version 01	15
A.3. Version 02	16
Appendix B. Connection monetary cost	16
B.1. Conditions	17
B.2. Price	17

B.3. Examples	18
Authors' Addresses	19

1. Introduction

It has become very common in modern networks that hosts have internet or more specific network access through different networking interfaces, tunnels, or next-hop routers. The concept of Provisioning Domain (PvD) was defined in [RFC7556] as a set of network configuration information which can be used by hosts in order to access the network.

This specification provides a way to identify explicit PvDs with Fully Qualified Domain Names called PvD IDs, which are included in a new Router Advertisement [RFC4861] option. This new option, when present, is used to associate the correlated set of configuration information with the identified PvD. It is worth noting that multiple PvDs with different PvD IDs could be provisioned on any host interface, as well as noting that the same PvD ID could be used on different interfaces in order to inform the host that both PvDs, on different interfaces, ultimately provide identical services.

This document also introduces a way for hosts to retrieve additional information related to a specific PvD by the mean of an HTTP-over-TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional network information that would typically be considered unfit, or too large, to be directly included in the Router Advertisements. This information can be used by the networking stack, the applications, or even be partially displayed to the users (e.g., by displaying a localized network service name).

2. Terminology

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

In addition, this document uses the following terminology:

PvD: A Provisioning Domain, a set of network configuration information; for more information, see [RFC7556].

PvD ID: A Fully Qualified Domain Name (FQDN) used to identify a PvD.

Explicit PvD: A PvD uniquely identified with a PvD ID. for more information, see [RFC7556].

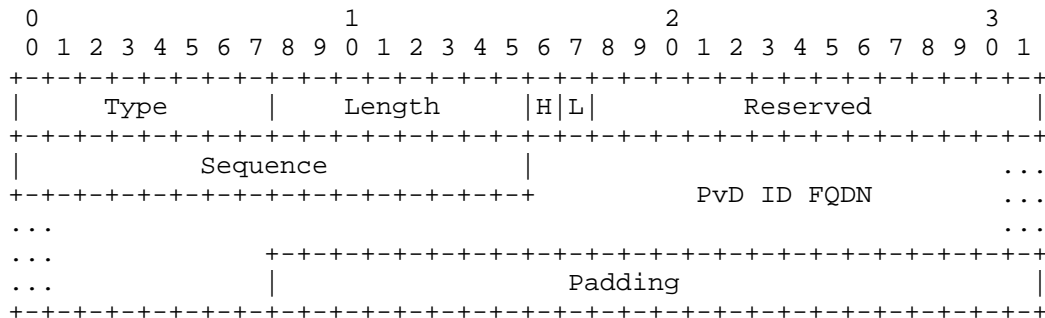
Implicit PvD: A PvD associated with a set of configuration information that, in the absence of a PvD ID, is associated with the advertising router.

3. Provisioning Domain Identification using Router Advertisements

Each provisioning domain is identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) which MUST belong to the network operator in order to avoid ambiguity. The same PvD ID MAY be used in several access networks when the set of configuration information is identical (e.g. in all home networks subscribed to the same service).

3.1. PvD ID Option for Router Advertisements

This document introduces a new Router Advertisement (RA) option called the PvD ID Router Advertisement Option, used to convey the FQDN identifying a given PvD.



PvD ID Router Advertisements Option format

- Type : (8 bits) To be defined by IANA.
- Length : (8 bits) The length of the option (including the Type and Length fields) in units of 8 octets.
- H-flag : (1 bit) Whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 4.
- L-flag : (1 bit) Whether the router is also providing IPv4 access using DHCPv4 (see Section 3.3.2).
- Reserved : (14 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.
- Sequence : (16 bits) Sequence number for the PvD Additional Information, as described in Section 4.

PvD ID FQDN : An ASCII string representation of the FQDN used as PvD ID. The string ends at the first byte set to zero, or the end of the option, whichever comes first.

Padding : Zero or more padding octets such as to set the option length (Type and Length fields included) to eight times the value of the Length field. It MUST be set to zero by the sender and ignored by the receiver.

Routers MUST NOT include more than one PvD ID Router Advertisement Option in each RA. In case multiple PvD ID options are found in a given RA, hosts MUST ignore all but the first PvD ID option.

Note: The existence and/or size of the sequence number is subject to discussion. The validity of a PvD Additional Information object is included in the object itself, but this only allows for 'pull based' updates, whereas the RA options usually provide 'push based' updates.

3.2. Router Behavior

A router MAY insert at most one PvD ID Option in its RAs. The included PvD ID is associated with all the other options included in the same RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191] options).

In order to provide multiple independent PvDs, a router MUST send multiple RAs using different source link-local addresses (LLA) (as proposed in [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]), each of which MAY include a PvD ID option. In such cases, routers MAY originate the different RAs using the same datalink layer address.

If the router is actually a VRRP instance [RFC5798], then the procedure is identical except that the virtual datalink layer address is used as well as the virtual IPv6 addresses.

3.3. Host Behavior

RAs are used to configure IPv6 hosts. When a host receives an RA message including a PvD ID Option, it MUST associate all the configuration objects which are updated by the received RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191] options) with the PvD identified by the PvD ID Option, even if some objects are already associated with a different explicit or implicit PvD.

If the received RA does not include a PvD ID Option, the host MUST associate the configuration objects which are updated by the received RA with an implicit PvD, even if some objects were already associated

with a different explicit or implicit PvD. This implicit PvD is identified by the link-local address of the router sending the RA and the interface on which the RA was received.

This document does not update the way Router Advertisement options are processed. But in addition to the option processing defined in other documents, hosts implementing this specification **MUST** associate each created or updated object (e.g. address, default route, more specific route, DNS server list) with the PvD associated with the received RA.

Note: There is a discussion whether there can be multiple implicit PvDs on a single interface (i.e. whether the router link-local address should be used to identify the implicit PvDs).

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm ([RFC2461], [RFC4191] and [RFC8028]), hosts **MAY** consider only the configuration associated with an arbitrary set of PvDs.

For example, a host **MAY** associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used for a given connection. In particular, constrained devices such as small battery operated devices (e.g. IoT), or devices with limited CPU or memory resources may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

3.3.1. DHCPv6 configuration association

When a host retrieves configuration elements using DHCPv6, they **MUST** be associated with the explicit or implicit PvD of the RA received on the same interface, using the same link-local address, and with the O-flag set [RFC4861]. If no such PvD is found, or whenever multiple different PvDs are found, the host behavior is unspecified.

This process requires hosts to keep track of received RAs, associated PvD IDs, and routers link-local addresses.

3.3.2. DHCPv4 configuration association

When a host retrieves configuration elements from DHCPv4, they MUST be associated with the explicit PvD received on the same interface, whose PVD ID Options L-flag is set and, in the case of a non point-to-point link, using the same link-layer address. If no such PvD is found, or whenever multiple different PvDs are found, the configuration elements coming from DHCPv4 MUST be associated with an IPv4-only implicit PvD identified by the interface on which the DHCPv4 transaction happened.

3.3.3. Interconnection Sharing by the Host

The situation when a host becomes also a router by acting as a router or ND proxy on a different interface (such as WiFi) to share the connectivity of another interface (such as cellular), also known as "tethering" is TBD but it is expected that the one or several PvD associated to the shared interface will also be advertised to the clients.

4. Provisioning Domain Additional Information

Once a new PvD ID is discovered, it may be used to retrieve additional information about the characteristics of the provided connectivity. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC7159].

The purpose of this additional set of information is to securely provide additional information to hosts about the connectivity that is provided using a given interface and source address pair. It typically includes data that would be considered too large, or not critical enough, to be provided within an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.3.

4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD ID Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/well-known/pvd` [RFC5785]. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

Note: Should the PvD AI retrieval be a MAY or a SHOULD ? Could the object contain critical data, or should it only contain informational data ?

Note that the DNS name resolution of <PvD-ID> as well as the actual query MUST be performed using the PvD associated with the PvD ID. In other words, the name resolution, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.3. In some cases, it may therefore be necessary to wait for an address to be available for use (e.g., once the Duplicate Address Detection or DHCPv6 processes are complete) before initiating the HTTP over TLS query.

If the HTTP status of the answer is greater than or equal to 400 the host MUST abandon and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 included and 399 included it MUST follow the redirection(s). If the HTTP status of the answer is between 200 included and 299 included the host MAY get a file containing a single JSON object. When a JSON object could not be retrieved, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

After retrieval of the PvD Additional Information, hosts MUST watch the PvD ID Sequence field for change. In case a different value than the one in the RA Sequence field is observed, or whenever the validity time included in the PVD Additional Information JSON object is expired, hosts MUST either perform a new query and retrieve a new version of the object, or deprecate the object and stop using it.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields Section 4.3. A retrieved object including an outdated expiration time or missing a mandatory element MUST be ignored. In order to avoid traffic spikes toward the server hosting the PvD Additional Information when an object expires, a host which last retrieved an object at a time A, including a validity time B, SHOULD renew the object at a uniformly random time in the interval $[(B-A)/2, A]$.

The PvD Additional Information object includes a set of IPv6 prefixes which MUST be checked against all the Prefix Information Options advertised in the Router Advertisement. If any of the prefixes included in the Prefix Information Options is not included in at least one of the listed prefixes, the PvD associated with the tested prefix MUST be considered unsafe and MUST NOT be used. While this does not prevent a malicious network provider, it does complicate some attack scenarios, and may help detecting misconfiguration.

The server providing the JSON files SHOULD also check whether the client address is part of the prefixes listed into the additional information and SHOULD return a 403 response code if there is no

match. The server MAY also use the client address to select the right JSON object to be returned.

4.2. Providing the PvD Additional Information

Whenever the H-flag is set in the PvD RA Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA. In particular, when a captive portal is present, hosts MUST still be allowed to access the object, even before logging into the captive portal.

Routers MAY increment the PVD ID Sequence number in order to inform host that a new PvD Additional Information object is available and should be retrieved.

4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

The following array presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
name	Human-readable service name	UTF-8 string	"Awesome Wifi"
expires	Date after which this object is not valid	[RFC3339]	"2017-07-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PVD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include a valid string associated with the "name" key at the root of the object, or a valid date associated with the "expiration" key, also at the root of the object, MUST be ignored. In such cases, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
localizedName	Localized user-visible service name, language can be selected based on the HTTP Accept-Language header in the request.	UTF-8 string	"Wifi Genial"
noInternet	No Internet, set when the PvD only provides restricted access to a set of services.	boolean	true
characteristics	Connectivity characteristics	JSON object	See Section 4.3.1
metered	metered, when the access volume is limited.	boolean	false

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements.

4.3.1. Connectivity Characteristics Information

The following set of keys can be used to signal certain characteristics of the connection towards the PvD.

They should reflect characteristics of the overall access technology which is not limited to the link the host is connected to, but rather a combination of the link technology, CPE upstream connectivity, and further quality of service considerations.

JSON key	Description	Type	Example
maxThroughput	Maximum achievable throughput	object({down(int), up(int)}) in kb/s	{"down": 10000, "up": 5000}
minLatency	Minimum achievable latency	object({down(int), up(int)}) in ms	{"down": 10, "up": 20}
rl	Maximum achievable reliability	object({down(int), up(int)}) in losses every 1000 packets	{"down": 0.1, "up": 1}

4.3.2. Private Extensions

JSON keys starting with "x-" are reserved for private use and can be utilized to provide information that is specific to vendor, user or enterprise. It is RECOMMENDED to use one of the patterns "x-FQDN-KEY" or "x-PEN-KEY" where FQDN is a fully qualified domain name or PEN is a private enterprise number [PEN] under control of the author of the extension to avoid collisions.

4.3.3. Example

Here are two examples based on the keys defined in this section.

```
{
  "name": "Foo Wireless",
  "localizedName": "Foo-France Wifi",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes" : ["2001:db8:1::/48", "2001:db8:4::/48"],
  "characteristics": {
    "maxThroughput": { "down":200000, "up": 50000 },
    "minLatency": { "down": 0.1, "up": 1 }
  }
}

{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "metered": true,
  "characteristics": {
    "maxThroughput": { "down":80000, "up": 20000 }
  }
}
```

5. Security Considerations

Although some solutions such as IPsec or SEND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, actual deployments largely rely on link layer or physical layer security mechanisms (e.g. 802.1x [IEEE8021X]) in conjunction with RA Guard [RFC6105].

This specification does not improve the Neighbor Discovery Protocol security model, but extends the purely link-local configuration retrieval mechanisms with HTTP-over-TLS communications.

During the exchange, the server authenticity is verified by the mean of a certificate, validated based on the FQDN found in the Router Advertisement (e.g. using a list of pre-installed CA certificates, or DNSSEC [RFC4035] with DNS Based Authentication of Named Entities [RFC6698]). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTP server. But this does not mean the Advertising Router and the PvD server belong to the same entity.

The IPv6 prefixes list included in the PvD Additional Information JSON object is used to validate that the prefixes included in the Router Advertisements are really part of the PvD. An adversarial router willing to fake the use of a given explicit PvD, without any access to the actual PvD, would need to perform NAT66 in order to circumvent this check.

It is also RECOMMENDED that the PvD server checks the source addresses of incoming connexions (see Section 4.1). This check ensures that the internet access provided by any router advertising a given PvD eventually reaches the internet using the actual PvD (Tunneling can still be used).

For privacy reasons, it is desirable that the PvD Additional Information object may only be retrieved by the hosts using the given PvD. Host identity SHOULD be validated based on the client address that is used during the HTTP query.

6. Privacy Considerations

TBD

7. IANA Considerations

IANA is kindly requested to allocate a new IPv6 Neighbor Discovery option number for the PvD ID Router Advertisement option.

The URI used to retrieve the PvD Additional Information JSON object is the well known URI (see [RFC5785]) with the URI suffix "pvd".

TBD: JSON keys will need a new registry.

8. Acknowledgements

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns].

Thanks also to Ray Bellis, Lorenzo Colitti, Thierry Danis, Marcus Keane, Erik Kline, Jen Lenkova, Mark Townsley, James Woodyatt and Mikael Abrahamson for useful and interesting discussions.

Finally, many thanks to Thierry Danis for his implementation work ([github]), Tom Jones for his integration effort into the Neat project and Rigil Salim for his implementation work.

9. References

9.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.

9.2. Informative references

- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<http://www.rfc-editor.org/info/rfc5798>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<http://www.rfc-editor.org/info/rfc6105>>.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<http://www.rfc-editor.org/info/rfc6698>>.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<http://www.rfc-editor.org/info/rfc8028>>.
- [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]
Baker, F., Bowers, C., and J. Linkova, "Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution", draft-bowbakova-rtgwg-enterprise-pa-multihoming-01 (work in progress), October 2016.
- [I-D.stenberg-mif-mpvd-dns]
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.

[I-D.kline-mif-mpvd-api-reqs]

Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.

[PEN]

IANA, "Private Enterprise Numbers", <<https://www.iana.org/assignments/enterprise-numbers>>.

[IEEE8021X]

IEEE, "IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std", .

[github]

Cisco, "IPv6-mPvD github repository", <<https://github.com/IPv6-mPvD>>.

Appendix A. Changelog

Note to RFC Editors: Remove this section before publication.

A.1. Version 00

Initial version of the draft. Edited by Basile Bruneau + Eric Vyncke and based on Basile's work.

A.2. Version 01

Major rewrite intended to focus on the the retained solution based on corridors, online, and WG discussions. Edited by Pierre Pfister. The following list only includes major changes.

PvD ID is an FQDN retrieved using a single RA option. This option contains a sequence number for push-based updates, a new H-flag, and a L-flag in order to link the PvD with the IPv4 DHCP server.

A lifetime is included in the PvD ID option.

Detailed Hosts and Routers specifications.

Additional Information is retrieved using HTTP-over-TLS when the PvD ID Option H-flag is set. Retrieving the object is optional.

The PvD Additional Information object includes a validity date.

DNS-based approach is removed as well as the DNS-based encoding of the PvD Additional Information.

Major cut in the list of proposed JSON keys. This document may be extended later if need be.

Monetary discussion is moved to the appendix.

Clarification about the 'prefixes' contained in the additional information.

Clarification about the processing of DHCPv6.

A.3. Version 02

The FQDN is now encoded with ASCII format (instead of DNS binary) in the RA option.

The PvD ID option lifetime is removed from the object.

Use well known URI "https://<PvD-ID>/.well-known/pvd"

Reference RFC3339 for JSON timestamp format.

The PvD ID Sequence field has been extended to 16 bits.

Modified host behavior for DHCPv4 and DHCPv6.

Removed IKEv2 section.

Removed mention of RFC7710 Captive Portal option. A new I.D. will be proposed to address the captive portal use case.

Appendix B. Connection monetary cost

NOTE: This section is included as a request for comment on the potential use and syntax.

The billing of a connection can be done in a lot of different ways. The user can have a global traffic threshold per month, after which his throughput is limited, or after which he/she pays each megabyte. He/she can also have an unlimited access to some websites, or an unlimited access during the weekends.

An option is to split the bill in elementary billings, which have conditions (a start date, an end date, a destination IP address...). The global billing is an ordered list of elementary billings. To know the cost of a transmission, the host goes through the list, and the first elementary billing whose the conditions are fulfilled gives the cost. If no elementary billing conditions match the request, the host MUST make no assumption about the cost.

B.1. Conditions

Here are the potential conditions for an elementary billing. All conditions MUST be fulfill.

Key	Description	Type	JSON Example
beginDate	Date before which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
endDate	Date after which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
domains	FQDNs whose the billing is limited	array(string)	["deezer.com", "spotify.com"]
prefixes4	IPv4 prefixes whose the billing is limited	array(string)	["78.40.123.182/32", "78.40.123.183/32"]
prefixes6	IPv6 prefixes whose the billing is limited	array(string)	["2a00:1450:4007:80e::200e/64"]

B.2. Price

Here are the different possibilities for the cost of an elementary billing. A missing key means "all/unlimited/unrestricted". If the elementary billing selected has a trafficRemaining of 0 kb, then it means that the user has no access to the network. Actually, if the last elementary billing has a trafficRemaining parameter, it means that when the user will reach the threshold, he/she will not have access to the network anymore.

Key	Description	Type	JSON Example
pricePerGb	The price per Gigabit	float (currency per Gb)	2
currency	The currency used	ISO 4217	"EUR"
throughputMax	The maximum achievable throughput	float (kb/s)	100000
trafficRemaining	The traffic remaining	float (kB)	12000000

B.3. Examples

Example for a user with 20 GB per month for 40 EUR, then reach a threshold, and with unlimited data during weekends and to example.com:

```
[
  {
    "domains": ["example.com"]
  },
  {
    "prefixes4": ["78.40.123.182/32", "78.40.123.183/32"]
  },
  {
    "beginDate": "2016-07-16T00:00:00Z",
    "endDate": "2016-07-17T23:59:59Z",
  },
  {
    "beginDate": "2016-06-20T00:00:00Z",
    "endDate": "2016-07-19T23:59:59Z",
    "trafficRemaining": 12000000
  },
  {
    "throughputMax": 100000
  }
]
```

If the host tries to download data from example.com, the conditions of the first elementary billing are fulfilled, so the host takes this elementary billing, finds no cost indication in it and so deduces that it is totally free. If the host tries to exchange data with foobar.com and the date is 2016-07-14T19:00:00Z, the conditions of the first, second and third elementary billing are not fulfilled.

But the conditions of the fourth are. So the host takes this elementary billing and sees that there is a threshold, 12 GB are remaining.

Another example for a user abroad, who has 3 GB per year abroad, and then pay each MB:

```
[
  {
    "beginDate": "2016-02-10T00:00:00Z",
    "endDate": "2017-02-09T23:59:59Z",
    "trafficRemaining": 3000000
  },
  {
    "pricePerGb": 30,
    "currency": "EUR"
  }
]
```

Authors' Addresses

Pierre Pfister (editor)
Cisco
11 Rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: ppfister@cisco.com

David Schinazi
Apple

Email: dschinazi@apple.com

Tommy Pauly
Apple

Email: tpauly@apple.com

Eric Vyncke
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

Basile Bruneau
Ecole Polytechnique
Vannes 56000
France

Email: basile.bruneau@polytechnique.edu

IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

F. Gont
SI6 Networks / UTN-FRH
G. Gont
SI6 Networks
M. Garcia Corbo
SITRANS
C. Huitema
Private Octopus Inc.
October 30, 2017

Problem Statement Regarding IPv6 Address Usage
draft-gont-6man-address-usage-recommendations-04

Abstract

This document analyzes the security and privacy implications of IPv6 addresses based on a number of properties (such as address scope, stability, and usage type), and identifies gaps that currently prevent systems and applications from leveraging the increased flexibility and availability of IPv6 addresses.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 May 3, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<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
2. Terminology	3
3. Background	3
4. IPv6 Address Properties	4
4.1. Address Scope Considerations	4
4.2. Address Stability Considerations	5
4.3. Usage Type Considerations	6
5. Default Address Selection in IPv6	7
6. Current Possible Approaches for IPv6 Address Usage	8
6.1. Incoming communications	9
6.2. Outgoing communications	9
7. Problem Statement	10
7.1. Issues Associated with Sub-optimal IPv6 Address Usage	10
7.1.1. Correlation of Network Activity	10
7.1.2. Testing for the Presence of Node in the Network	10
7.1.3. Unexpected Address Discovery	11
7.1.4. Availability Outside the Expected Scope	11
7.2. Current Limitations in the Address Selection APIs	12
7.3. Sub-optimal IPv6 Address Configuration	12
7.4. Sub-optimal IPv6 Address Usage	13
8. IANA Considerations	14
9. Security Considerations	14
10. Acknowledgements	14
11. References	14
11.1. Normative References	14
11.2. Informative References	15
Authors' Addresses	16

1. Introduction

IPv6 addresses may differ in a number of properties, such as address scope (e.g. link-local vs. global), stability (e.g. stable addresses vs. temporary addresses), and intended usage type (outgoing communications vs. incoming communications). While often overlooked, these properties have impact on areas such as security, privacy, and performance.

IPv6 hosts typically configure a number of IPv6 addresses of different properties. For example, a host may configure one stable and one temporary address per each autoconfiguration prefix

advertised on the local network. Currently, the addresses to be configured typically depend on local system policy, with the aforementioned policy being static and irrespective of the network the host attaches to. This "one size fits all" approach limits the ability of systems and applications of fully-leveraging the increased flexibility and availability of IPv6 addresses.

Each application running on a given system may have its own set of requirements or expectations for the properties of the IPv6 addresses to be employed. For example, an application meaning to offer a public service might expect to employ global stable addresses for such purpose, while a privacy-sensible client application might prefer short-lived temporary addresses, or might even expect to employ single-use ("throw-away") IPv6 addresses when connecting to public servers. However, the subtleties associated with IPv6 addresses (and associated properties) are often ignored by application programmers and, in any case, current APIs (such as the BSD Sockets API) tend to be very limited in the amount of control they give applications to select the most appropriate IPv6 addresses for a given task, thus limiting a programmer's ability to leverage IPv6 address availability and properties.

This document analyzes the impact of a number of properties of IPv6 addresses on areas such as security and privacy, and analyzes how IPv6 addresses are currently generated and employed by different operating systems and applications. Finally, it provides a problem statement by identifying and analyzing gaps that prevent systems and applications from fully-leveraging IPv6 addressing capabilities, setting the basis for new work that could fill those gaps.

2. Terminology

This document employs the definitions of "public address", "stable address", and "temporary address" from Section 2 of [RFC7721].

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

3. Background

Predictable IPv6 addresses result in a number of security and privacy implications. For example, [Barnes2012] discusses how patterns in network prefixes can be leveraged for IPv6 address scanning. On the other hand, [RFC7707], [RFC7721] and [RFC7217] discuss the security and privacy implications of predictable IPv6 Interface Identifiers (IIDs).

Given the aforementioned previous work in this area, and the formal specification update produced by [RFC8064], we expect (and assume in the rest of this document) that implementations have replaced any schemes that produce predictable addresses with alternative schemes that avoid such patterns (e.g., RFC7217 in replacement of the traditional SLAAC addresses that embed link-layer addresses).

4. IPv6 Address Properties

There are three parameters that affect the security and privacy properties of an IPv6 address:

- o Scope
- o Stability
- o Usage type (client-like "outgoing connections" vs. server-like "incoming connections")

Section 4.1, Section 4.2, and Section 4.3 discuss the security and privacy implications (and associated tradeoffs) of the scope, stability and usage type properties of IPv6 addresses, respectively.

4.1. Address Scope Considerations

The IPv6 address scope can, in some scenarios, limit the attack exposure of a node as a result of the implicit isolation provided by a non-global address scope. For example, a node that only employs link-local addresses may, in principle, only be exposed to attack from other nodes in the local link. Hosts employing only Unique Local Addresses (ULAs) may be more isolated from attack than those employing Global Unicast Addresses (GUAs), assuming that proper packet filtering is enforced at the network edge.

The potential protection provided by a non-global addresses should not be regarded as a complete security strategy, but rather as a form of "prophylactic" security (see [I-D.gont-opsawg-firewalls-analysis]).

We note that the use of non-global addresses is usually limited to a reduced type of applications/protocols that e.g. are only meant to operate on a reduced scope, and hence their applicability may be limited.

A discussion of ULA usage considerations can be found in [I-D.ietf-v6ops-ula-usage-considerations].

4.2. Address Stability Considerations

The stability of an address has two associated security/privacy implications:

- o Ability of an attacker to correlate network activity
- o Exposure to attack

For obvious reasons, an address that is employed for multiple communication instances allows the aforementioned network activities to be correlated. The longer an address is employed (i.e., the more stable it is), the longer such correlation will be possible. In the worst-case scenario, a stable address that is employed for multiple communication instances over time will allow all such activities to be correlated. On the other hand, if a host were to generate (and eventually "throw away") one new address for each communication instance (e.g., TCP connection), network activity correlation would be mitigated.

NOTE:

The use of constant IIDs (as in traditional SLAAC) result in addresses that, while not constant as a whole (since the prefix changes), contain a globally-unique value that leaks out the node "identity". Such addresses result in the worst possible security and privacy implications, and their use has been deprecated by [RFC8064].

Typically, when it comes to attack exposure, the longer an address is employed the longer an attacker is exposed to attacks (e.g. an attacker has more time to find the address in the first place [RFC7707]). While such exposure is traditionally associated with the stability of the address, the usage type of the address (see Section 4.3) may also have an impact on attack exposure.

A popular approach to mitigate network activity correlation is the use of "temporary addresses" [RFC4941]. Temporary addresses are typically configured and employed along with stable addresses, with the temporary addresses employed for outgoing communications, and the stable addresses employed for incoming communications.

NOTE:

Ongoing work [I-D.gont-6man-non-stable-iids] aims at updating [RFC4941] such that temporary addresses can be employed without the need to configure stable addresses.

We note that the extent to which temporary addresses provide improved mitigation of network activity correlation and/or reduced attack

exposure may be questionable and/or limited in some scenarios. For example, a temporary address that is reachable for, say, a few hours has a questionable "reduced exposure" (particularly when automated attack tools do not typically require such a long period of time to complete their task). Similarly, if network activity can be correlated for the life of such address (e.g., on the order of several hours), such period of time might be long enough for the attacker to correlate all the network activity he is meaning to correlate.

In order to better mitigate network activity correlation and/or possibly reduce host exposure, an implementation might want to either reduce the preferred lifetime of a temporary address, or even better, generate one new temporary address for each new transport protocol instance. However, the associated lifetime/stability of an address may have a negative impact on the network. For example, if a node were to employ "throw away" IPv6 addresses, or employ temporary addresses [RFC4941] with a short preferred lifetime, local nodes might need to maintain too many entries in their Neighbor Cache, and a number of devices (possibly enforcing security policies) might also need to cope with such additional state.

Additionally, enforcing a maximum lifetime on IPv6 addresses may cause long-lived TCP connections to fail. For example, an address becoming "Invalid" (after transitioning through the "Preferred" and "Deprecated" states) would cause the TCP connections employing them to break. This, in turn, would cause e.g. long-lived SSH sessions to break/fail.

In some scenarios, attack exposure may be reduced by limiting the usage of temporary addresses to outgoing connections, and prevent such addresses from being used for incoming connections (please see Section 4.3).

4.3. Usage Type Considerations

A node that employs one of its addresses to communicate with an external server (i.e., to perform an "outgoing connection") may cause such address to become exposed to attack. For example, once the external server receives an incoming connection, the corresponding server might launch an attack against the aforementioned address. A real-world instance of this type of scenario has been documented in [Hein].

However, we note that employing an IPv6 address for outgoing communications need not increase the exposure of local services to other parties. For example, nodes could employ temporary addresses only for outgoing connections, but not for incoming connections.

Thus, external nodes that learn about client's addresses could not really leverage such addresses for actively contacting the clients.

There are multiple ways in which this could possibly be achieved, with different implications. Namely:

- o Run a host-based or network-based firewall
- o Bind services to specific (explicit) addresses
- o Bind services only to stable addresses

A client could simply run a host-based firewall that only allows incoming connections on the stable addresses. This is clearly more of an operational way of achieving the desired functionality, and may require good firewall/host integration (e.g., the firewall should be able to tell stable vs. temporary addresses), may require the client to run additional firewall software for this specific purpose, etc. In other scenarios, a network-based firewall could be configured to allow outgoing communications from all internal addresses, but only allow incoming communications to stable addresses. For obvious reasons, this is generally only applicable to networks where incoming communications are allowed to a limited number of hosts/servers.

Services could be bound to specific (explicit) addresses, rather than to all locally-configured addresses. However, there are a number of short-comings associated with this approach. Firstly, an application would need to be able to learn all of its addresses and associated stability properties, something that tends to be non-trivial and non-portable, and that also makes applications protocol-dependent, unnecessarily. Secondly, the BSD Sockets API does not really allow a socket to be bound to a subset of the node's addresses. That is, sockets can be bound to a single address or to all available addresses (wildcard), but not to a subset of all the configured addresses.

Binding services only to stable addresses provides a clean separation between addresses employed for client-like outgoing connections and server-like incoming connections. However, we currently lack an appropriate API for nodes to be able to specify that a socket should only be bound to stable addresses.

5. Default Address Selection in IPv6

Applications use system API's to select the IPv6 addresses that will be used for incoming and outgoing connections. These choices have consequences in terms of privacy, security, stability and performance.

Default Address Selection for IPv6 is specified in [RFC6724]. The selection starts with a set of potential destination addresses, such as returned by `getaddrinfo()`, and the set of potential source addresses currently configured for the selected interfaces. For each potential destination address, the algorithm will select the source address that provides the best route to the destination, while choosing the appropriate scope and preferring temporary addresses. The algorithm will then select the destination address, while giving a preference to reachable addresses with the smallest scope. The selection may be affected by system settings. We note that [RFC6724] only applies for outgoing connections, such as those made by clients trying to use services offered by other hosts.

We note that [RFC6724] selects IPv6 addresses from all the currently available addresses on the host, and there is currently no way for an application to indicate expected or desirable properties for the IPv6 source addresses employed for such outgoing communications. For example, a privacy-sensitive application might want that each outgoing communication instance employs a new, single-use IPv6 address, or to employ a new reusable address that is not employed or reusable by any other application on the host. Reuse of an IPv6 address by an application would allow the correlation of all network activities corresponding to such application as being performed by the same host, while reuse of an IPv6 address by multiple different applications would allow the correlation of all such network activities as being performed by the host with such IPv6 address.

When devices provide a service, the common pattern is to just wait for connections over all addresses configured on the device. For example, applications using the BSD Sockets API will commonly `bind()` the listening socket to the undefined address. This long-established behavior is appropriate for devices providing public services, but may have unexpected results for devices providing semi-private services, such as various forms of peer-to-peer or local-only applications.

This behavior leads to three problems: device tracking, discussed in Section 7.1.2; unexpected address discovery, discussed in Section 7.1.3; and availability outside the expected scope, discussed in Section 7.1.4. These problems are caused in part by the limitations of available address selection API, presented in Section 7.2.

6. Current Possible Approaches for IPv6 Address Usage

6.1. Incoming communications

There are a number of ways in which a system or network may affect which address (and how) may be employed for different services and cases. Namely,

- o TCP/IP stack address filtering
- o Application-based address filtering
- o Firewall-based address filtering

Clearly, the most elegant approach for address selection is for applications to be able to specify the properties of the addresses they are willing to employ by means of an API, such the TCP/IP stack itself can "filter" which addresses are allowed to be employed for the given service/application. This relieves the application from dealing with low level details of networking, improves portability, and avoids duplicate code in applications. However, constraints in the current APIs (see Section 7.2) may limit the ability of application programmers for leveraging this technique.

Another possible approach is for applications to e.g. bind services to all available addresses, and perform the associated selection/filtering at the application level. While possible this has a number of drawbacks. Firstly, it would require applications to deal with low-level networking details, require that all the associated code be duplicated in all applications, and also negatively affect portability. Besides, performing address/selection filtering at the application level may not mitigate some possible threats. For example, port scanning will still be possible, since the aforementioned filtering will only be performed e.g. once UDP packets are received or TCP connections are established.

Finally, a firewall may be employed to filter addresses based on their intended usage. For example, a firewall may block incoming requests to all addresses except to some whitelisted addresses (such as the stable addresses of the node). This technique not only requires the use of a firewall (which may or may not be present), but also implies knowledge of the firewall regarding the desired properties of the addresses that each application/service is intended to use.

6.2. Outgoing communications

An application might be able to obtain the list of currently-configured addresses, and subsequently select an address with desired

properties, and explicitly "bind" the address to the socket, to override the default source address selection.

However, this approach is problematic for a number of reasons. Firstly, there is no portable way of obtaining the list of currently-configured addresses on the local node, and even less to check for properties such "valid lifetime". Secondly, as discussed in Section 6.1, it would require application programmers to understand all the subtleties associated with IPv6 addressing, and would also lead to duplicate code on all applications. Finally, applications would be limited to use already-configured addresses and unable to trigger the generation of new addresses where desirable (e.g. the generation of a new temporary address for this application instance or communication instance).

7. Problem Statement

This section elaborates the problem statement on IPv6 address usage. Section 7.1 describes the security and privacy implications of improper IPv6 address usage, while Section 7.2, Section 7.4, Section 7.3, analyze the possible root of such improper address usage, suggesting possible future work.

7.1. Issues Associated with Sub-optimal IPv6 Address Usage

7.1.1. Correlation of Network Activity

As discussed in [RFC7721], a node that reuses an IPv6 address for multiple communication instances would allow the correlation of such network activities. This could be the case when the same IPv6 address is employed by several instances of the same application (e.g., a browser in "privacy" mode and a browser in "normal" mode), or when the same IPv6 address is employed by two different applications on the same node (e.g., a browser in "privacy" mode, and an email client).

Particularly for privacy-sensitive applications, an application or system might want to limit the usage of a given IPv6 address to a single communication instance, a single application, a single user on the system, etc. However, given current APIs, this is practically impossible.

7.1.2. Testing for the Presence of Node in the Network

The stable addresses recommended in [RFC8064] use stable IIDs defined in [RFC7217]. One key part of that algorithm is that if a device connects to a given network at different times, it will always configure the same IPv6 addresses on that network. If the device

hosts a service ready to accept connections on that stable address, adversaries can test the presence of the device on the network by attempting connections to that stable address. Stable addresses used by listening services will thus enable testing whether a specific device is returning to a particular network, which in a number of cases might be considered a privacy issue.

7.1.3. Unexpected Address Discovery

Systems like DNS-Based Service Discovery [RFC6763] allow clients to discover services within a limited scope, that can be defined by a domain name. These services are not advertised outside of that scope, and thus do not expect to be discovered by random parties on the Internet. However, such services may be easily discoverable if they listen for connections to IPv6 addresses that a client process also uses as source address when connecting to remote servers.

NOTE:

An example of such unexpected discovery is described in [Hein]. A network manager observed scanning traffic directed at the temporary addresses of local devices. The analysis in [Hein] shows that the scanners learned the addresses by observing the device contact an NTP service ([RFC5905]). The remote scanning was possible because the local devices were also accepting connections directed to the temporary addresses.

It is obvious from the example that the "attack surface" of the services is increased because they are bound to the same IPv6 addresses that are also used by clients for outgoing communications with remote systems. But the overlap between "client" and "server" addresses is only one part of the problem. Suppose that a device hosts both a video game and a home automation application. The video game users will be able to discover the IPv6 address of the game server. If the home automation server listens to the same IPv6 addresses, it is now exposed to connection attempts by all these users. That, too, increases the attack surface of the home automation server.

7.1.4. Availability Outside the Expected Scope

The IPv6 addressing architecture [RFC4291] defines multiple address scopes. In practice, devices are often configured with globally reachable unicast addresses, link local addresses, and Unique Local IPv6 Unicast Addresses (ULA) [RFC4193]. Availability outside the expected scope happens when a service is expected to be only available in some local scope, but inadvertently becomes available to remote parties. That could happen for example if a service is meant to be available only on a given link, but becomes reachable through

ULA or through globally reachable addresses, or if a service is meant to be available only inside some organization's perimeter and becomes reachable through globally reachable addresses. It will happen in particular if a service intended for some local scope is programmed to bind to "unspecified" addresses, which in practice means every address configured for the device (please see Section 7.2).

7.2. Current Limitations in the Address Selection APIs

Application developers using the BSD Sockets API can "bind" a listening socket to a specific address, and ensure that the application is only reachable through that address. In theory, careful selection of the binding address could mitigate the problems described in Section 7.1. Binding services to temporary addresses could mitigate the ability of an attacker from testing for the presence of the node in the network. Binding different services to different addresses could mitigate unexpected discovery. Binding services to link local addresses or ULA could mitigate availability outside the expected scope. However, explicitly managing addresses adds significant complexity to the application development. It requires that application developers master addressing architecture subtleties, and implement logic that reacts adequately to connectivity events and address changes. Experience shows that application developers would probably prefer some much simpler solution.

In addition, we should note that many application developers use high level APIs that listen to TLS, HTTP, or some other application protocol. These high level APIs seldom provide detailed access to specific IP addresses, and typically default to listening to all available addresses.

A more advanced API could allow an application programmer to select desired properties in an address (scope, lifespan, etc.), such that the best-suitable addresses are selected, while relieving the application for low-level IPv6 addressing details. Such API might also trigger the generation of new IPv6 addresses when the specified properties would require so.

7.3. Sub-optimal IPv6 Address Configuration

Most operating systems configure the same types of addresses regardless of the current "operating mode" or "profile" of the device (e.g., device connected to enterprise network vs roaming across untrusted networks). For example, many operating systems configure both stable [RFC8064] and temporary [RFC4941] addresses on all network interfaces. However, this "one size fits all" approach tends to be sub-optimal or inappropriate for some scenarios. For example,

enterprise networks typically prefer usage of only stable address, thus meaning that a network administrator needs to find the means for disabling the generation of temporary addresses on all those systems that would otherwise generate them. On the other hand, some mobile devices configure both stable and temporary addresses, even when their usage pattern (client-like operation, as opposed to offering services to other nodes) would allow for the more privacy-sensible option of configuring only temporary addresses.

The lack of better tuned address configuration policies has helped the "one size fits all" approach that, as noted, may lead to suboptimal results. Advice in this area might help achieve more optional address generation policies such that IPv6 addressing capabilities are fully leveraged.

NOTE:

One might envision a document that provides advice regarding the address generation for different typical scenarios (e.g., when to configure stable-only, temporary-only, or stable+temporary). In the most simple analysis, one might expect nodes in a typical enterprise network to employ only stable addresses. General-purpose nodes in a home or "trusted" network may want to employ both stable and temporary addresses. Finally, mobile nodes (e.g. when roaming across non-trusted networks) may want to employ only temporary addresses).

7.4. Sub-optimal IPv6 Address Usage

An application programmer, left with the question of which are the most appropriate addresses for a given usage type and application, typically resorts to the Default IPv6 Address Selection for IPv6 (see Section 5) for outgoing communications, and to accepting incoming communications on all available addresses for incoming communications. As discussed throughout this document, this leads to sub-optimal results. Besides, all applications on a node share the same pool of configured addresses, and applications are also prevented from triggering the generation of new addresses (e.g. to be employed for a particular application or communication instance).

Guidance in this area is warranted such that applications and systems fully-leverage IPv6 addressing.

NOTE:

Such guidance would elaborate, among other things, on the usage of IPv6 addresses when offering network services and when performing client-like communications. For example, for incoming communications, hosts might want to employ only the smallest-scope applicable addresses (if available) and, if stable addresses are

available, they might want to accept incoming connections only on such addresses (but **not** on temporary addresses). For client-like communications, hosts might prefer temporary addresses, unless the corresponding communication instances are expected to be long-lived (e.g., SSH sessions).

8. IANA Considerations

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

9. Security Considerations

The security and privacy implications associated with the predictability and lifetime of IPv6 addresses has been analyzed in [RFC7217] [RFC7721], and [RFC7707]. This document complements and extends the aforementioned analysis by considering other IPv6 properties such as the address scope and address usage type, and the associated tradeoffs. Finally, it describes possible future standards-track work to allow for greater flexibility in IPv6 address usage.

10. Acknowledgements

The authors would like to thank (in alphabetical order) Francis Dupont, Tatuya Jinmei, and Dave Thaler for providing valuable comments on earlier versions of this document.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.

- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <<https://www.rfc-editor.org/info/rfc8064>>.

11.2. Informative References

- [Barnes2012] Barnes, R., Altmann, R., and D. Kerr, "Mapping the Great Void Smarter scanning for IPv6", ISMA 2012 AIMS-4 - Workshop on Active Internet Measurements, February 2012, <https://www.caida.org/workshops/isma/1202/slides/aims1202_rbarnes.pdf>.
- [Hein] Hein, B., "The Rising Sophistication of Network Scanning", January 2016, <<http://netpatterns.blogspot.be/2016/01/the-rising-sophistication-of-network.html>>.
- [I-D.gont-6man-non-stable-iids] Gont, F., Huitema, C., Gont, G., and M. Corbo, "Recommendation on Temporary IPv6 Interface Identifiers", draft-gont-6man-non-stable-iids-01 (work in progress), March 2017.

- [I-D.gont-opsawg-firewalls-analysis]
Gont, F. and F. Baker, "On Firewalls in Network Security",
draft-gont-opsawg-firewalls-analysis-02 (work in
progress), February 2016.
- [I-D.ietf-v6ops-ula-usage-considerations]
Liu, B. and S. Jiang, "Considerations For Using Unique
Local Addresses", draft-ietf-v6ops-ula-usage-
considerations-02 (work in progress), March 2017.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6
Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016,
<<https://www.rfc-editor.org/info/rfc7707>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy
Considerations for IPv6 Address Generation Mechanisms",
RFC 7721, DOI 10.17487/RFC7721, March 2016,
<<https://www.rfc-editor.org/info/rfc7721>>.

Authors' Addresses

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

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

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

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

Madeleine Garcia Corbo
Servicios de Informacion del Transporte
Neptuno 358
Havana City 10400
Cuba

Email: madelen.garcial6@gmail.com

Christian Huitema
Private Octopus Inc.
Friday Harbor, WA 98250
U.S.A.

Email: huitema@huitema.net
URI: <http://privateoctopus.com>

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

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

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

Table of Contents

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

1. Introduction

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

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

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

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

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

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

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

2. Terminology

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

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

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

3. Protocol Overview

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

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

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

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

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

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

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

4. Protocol Requirements

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

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

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

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

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

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

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

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

5. Implementation Issues

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

The issues include:

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

5.1. Layering

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

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

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

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

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

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

5.2. Storing PMTU information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.3. Purging stale PMTU information

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

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

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

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

5.4. Packetization layer actions

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

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

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

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

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

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

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

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

5.5. Issues for other transport protocols

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

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

5.6. Management interface

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

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

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

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

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

6. Security Considerations

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

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

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

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

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

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

7. Acknowledgements

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

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

8. IANA Considerations

This document does not have any IANA actions

9. References

9.1. Normative References

[I-D.ietf-6man-rfc2460bis]

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

[ICMPv6]

Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.

9.2. Informative References

[FRAG]

Kent, C. and J. Mogul, "Fragmentation Considered Harmful", In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology, August 1987.

[ND]

Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.

[RFC1122]

Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.

[RFC1191]

Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<http://www.rfc-editor.org/info/rfc1191>>.

[RFC1981]

McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<http://www.rfc-editor.org/info/rfc1981>>.

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

Appendix A. Comparison to RFC 1191

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

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

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

Appendix B. Changes Since RFC 1981

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

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

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

B.1. Change History Since RFC1981

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

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

Working Group Internet Drafts

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

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

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

Individual Internet Drafts

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

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

Authors' Addresses

Jack McCann
Digital Equipment Corporation

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Jeffrey Mogul
Digital Equipment Corporation

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

Email: bob.hinden@gmail.com

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

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

Table of Contents

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

1. Introduction

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

- o Expanded Addressing Capabilities

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

- o Header Format Simplification

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

- o Improved Support for Extensions and Options

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

- o Flow Labeling Capability

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

- o Authentication and Privacy Capabilities

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

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

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

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

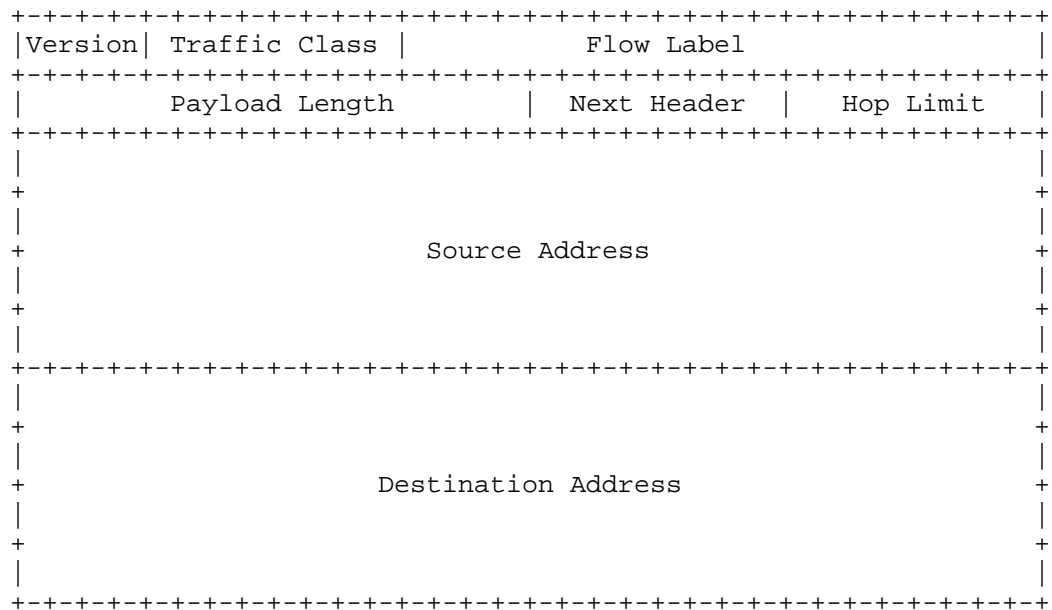
Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

2. Terminology

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

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

3. IPv6 Header Format



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

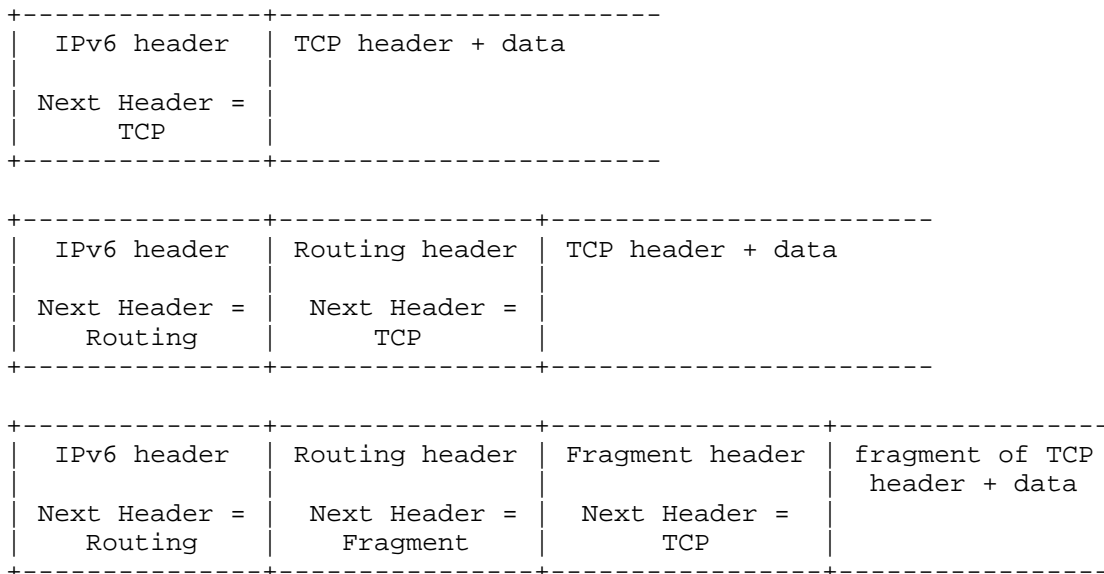
Next Header	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hop Limit	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. When forwarding, the packet is discarded if Hop Limit was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

4. IPv6 Extension Headers

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

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

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



Extension headers (except for the Hop-by-Hop Options header) are not processed, inserted, or deleted by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header.

The Hop-by-Hop Options header is not inserted or deleted, but may be examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

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

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

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

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

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

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

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

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

4.1. Extension Header Order

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

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

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

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

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

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

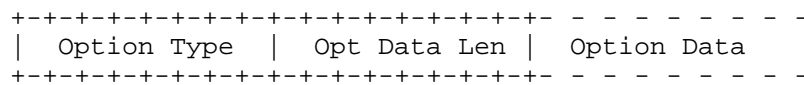
If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

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

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

4.2. Options

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



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

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

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

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

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

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

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

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

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

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

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

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

Pad1 option (alignment requirement: none)

```
+-----+
|           0           |
+-----+
```

NOTE! the format of the Pad1 option is a special case -- it does not have length and value fields.

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

PadN option (alignment requirement: none)

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+ - - - - - - -
|           1           | Opt Data Len | Option Data
+-----+-----+-----+-----+-----+-----+-----+-----+-----+ - - - - - - -
```

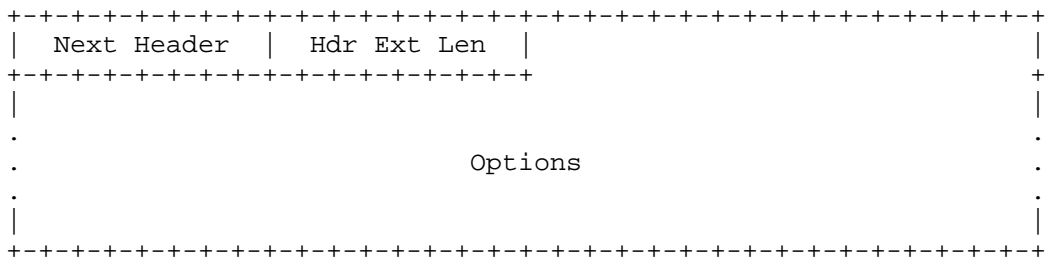
The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the

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

Appendix A contains formatting guidelines for designing new options.

4.3. Hop-by-Hop Options Header

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



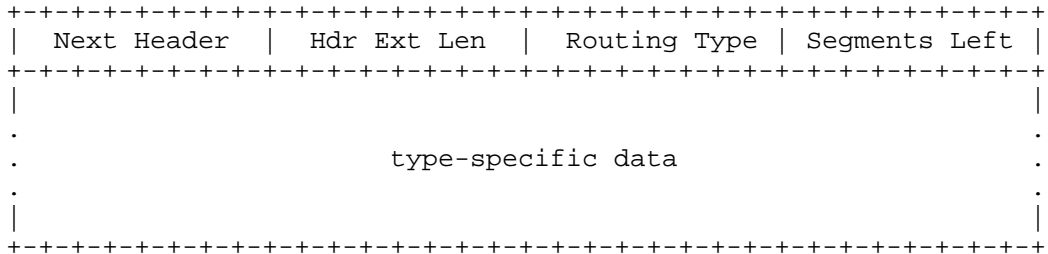
Next Header	8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Options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

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

4.4. Routing Header

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

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



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

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

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

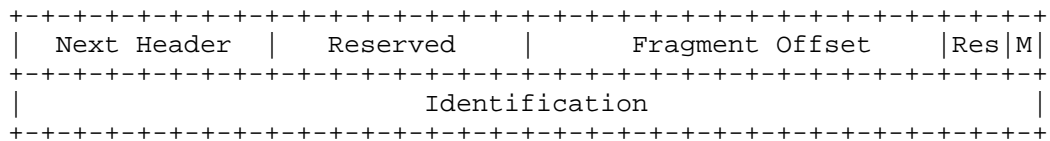
If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

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

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

4.5. Fragment Header

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



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

Identification 32 bits. See description below.

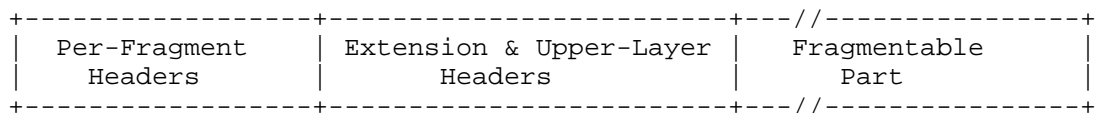
In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

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

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

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

original packet:



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

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

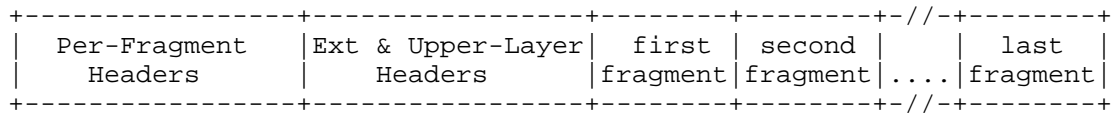
Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted ESP.

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

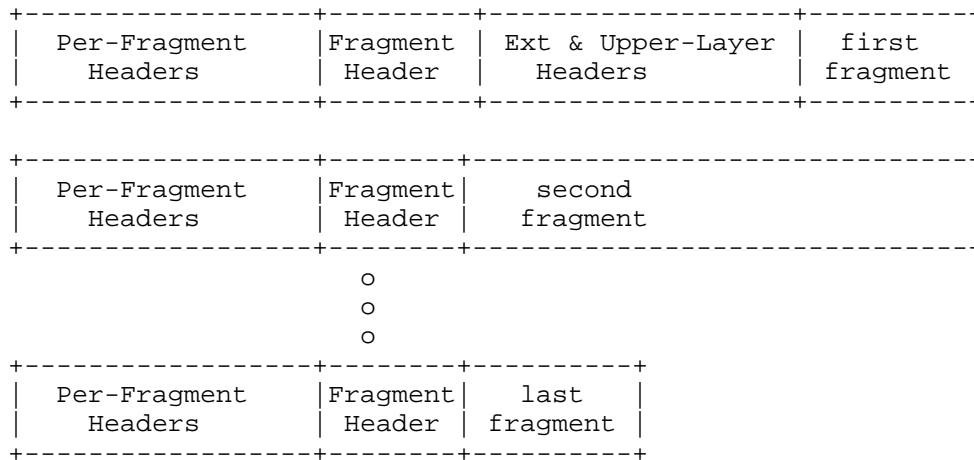
The Fragmentable Part of the original packet is divided into fragments. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s). Each complete fragment, except possibly the last ("rightmost") one, being an integer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

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

original packet:



fragment packets:



The first fragment packet is composed of:

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

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

(2) A Fragment header containing:

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

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

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

The Identification value generated for the original packet.

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

(4) The first fragment.

The subsequent fragment packets are composed of:

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

(2) A Fragment header containing:

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

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

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

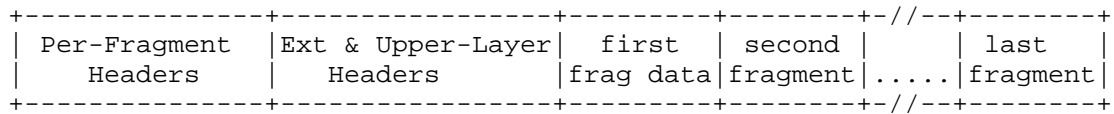
The Identification value generated for the original packet.

(3) The fragment itself.

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

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

reassembled original packet:



The following rules govern reassembly:

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

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

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

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

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

where

PL.orig = Payload Length field of reassembled packet.

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

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

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

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

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

The following error conditions may arise when reassembling fragmented packets:

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

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

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

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

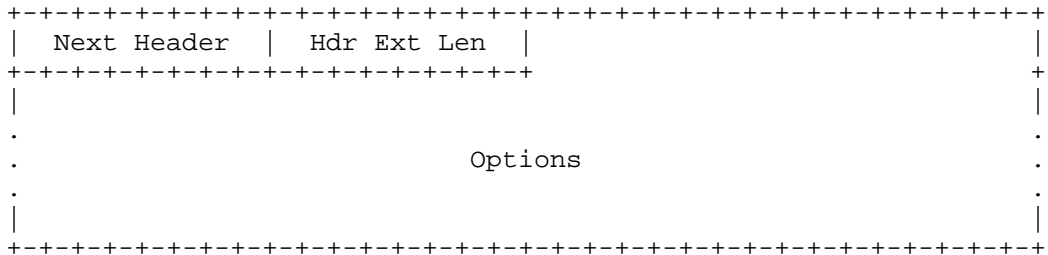
The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly.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

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

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

4.6. Destination Options Header

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



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

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

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

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

4.7. No Next Header

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

4.8. Defining New Extension Headers and Options

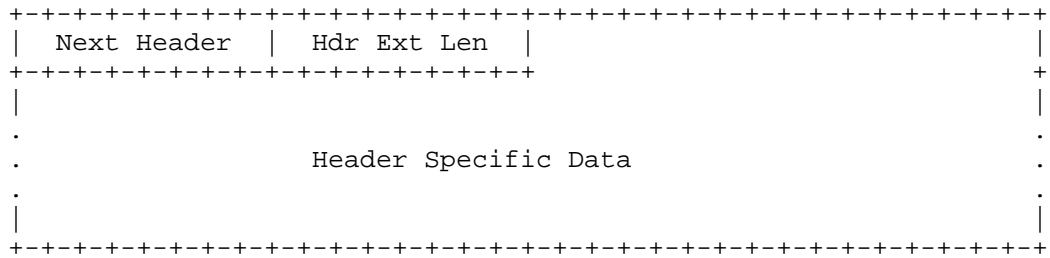
Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6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

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

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

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

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



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

5. Packet Size Issues

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

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

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

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

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

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

6. Flow Labels

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

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

7. Traffic Classes

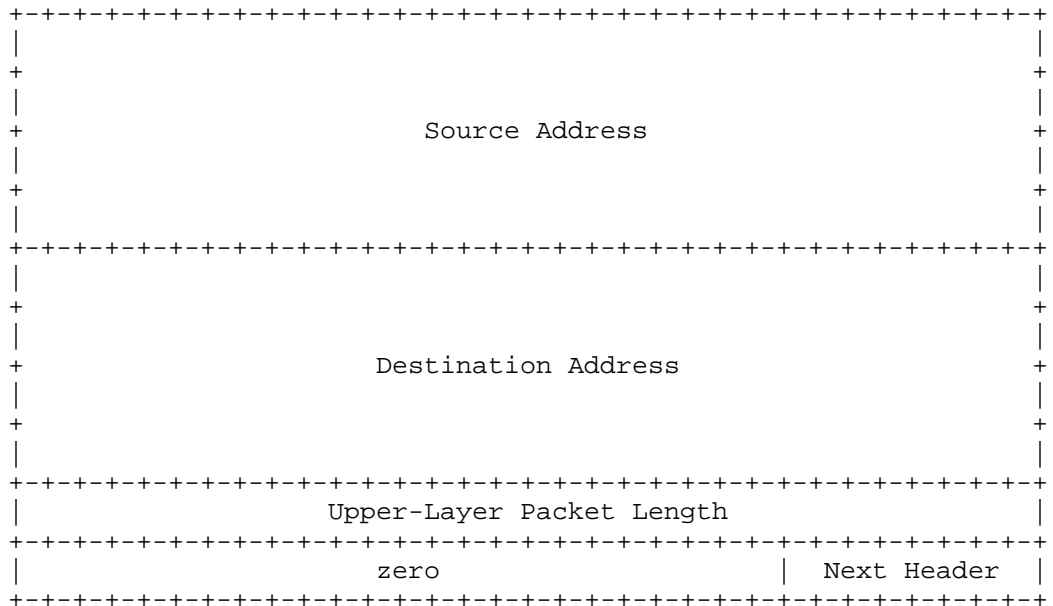
The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet or fragment might be different from the value sent by the packet's source.

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

8. Upper-Layer Protocol Issues

8.1. Upper-Layer Checksums

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



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

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

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

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

8.2. Maximum Packet Lifetime

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

8.3. Maximum Upper-Layer Payload Size

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

8.4. Responding to Packets Carrying Routing Headers

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

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

9. IANA Considerations

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

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

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

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

10. Security Considerations

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

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

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

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

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

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

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

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

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

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

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

11. Acknowledgments

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

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

12. References

12.1. Normative References

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

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

12.2. Informative References

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

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

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

Appendix A. Formatting Guidelines for Options

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

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

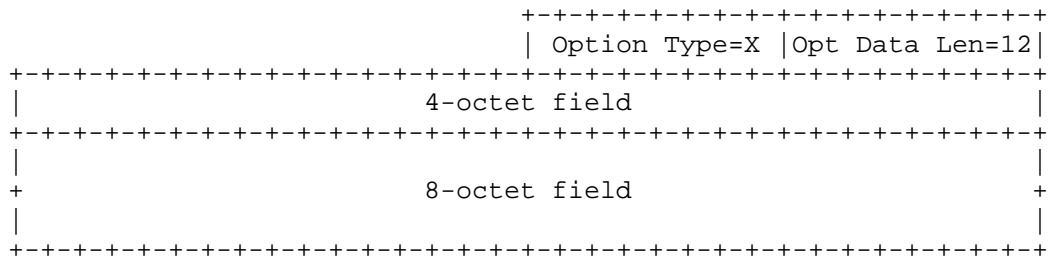
boundaries, i.e., fields of width n octets should be placed at an integer multiple of n octets from the start of the Hop-by-Hop or Destination Options header, for n = 1, 2, 4, or 8.

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

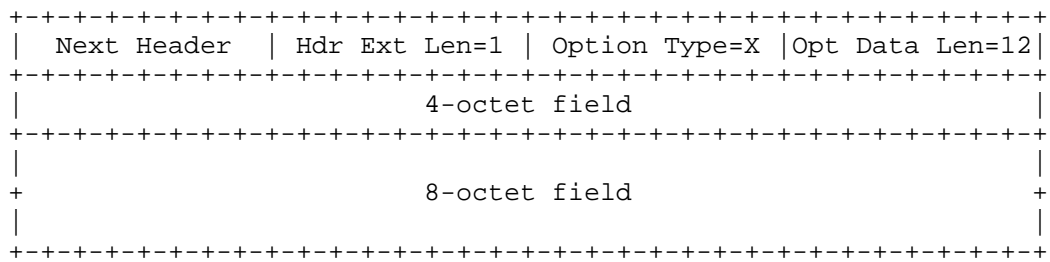
These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of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

Example 1

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

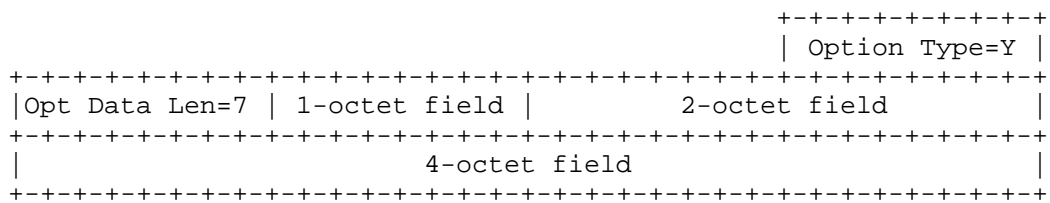


Its alignment requirement is 8n+2, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

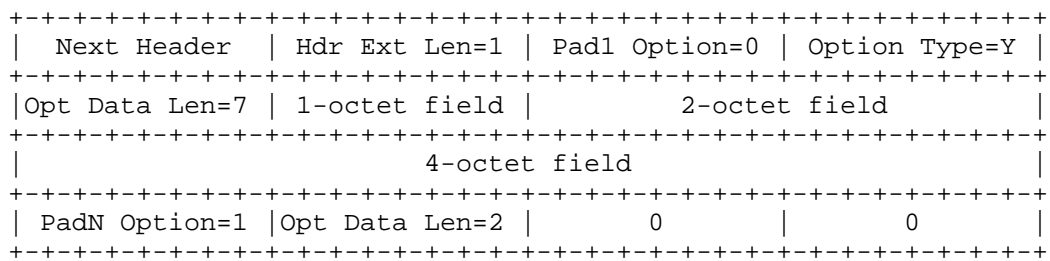


Example 2

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

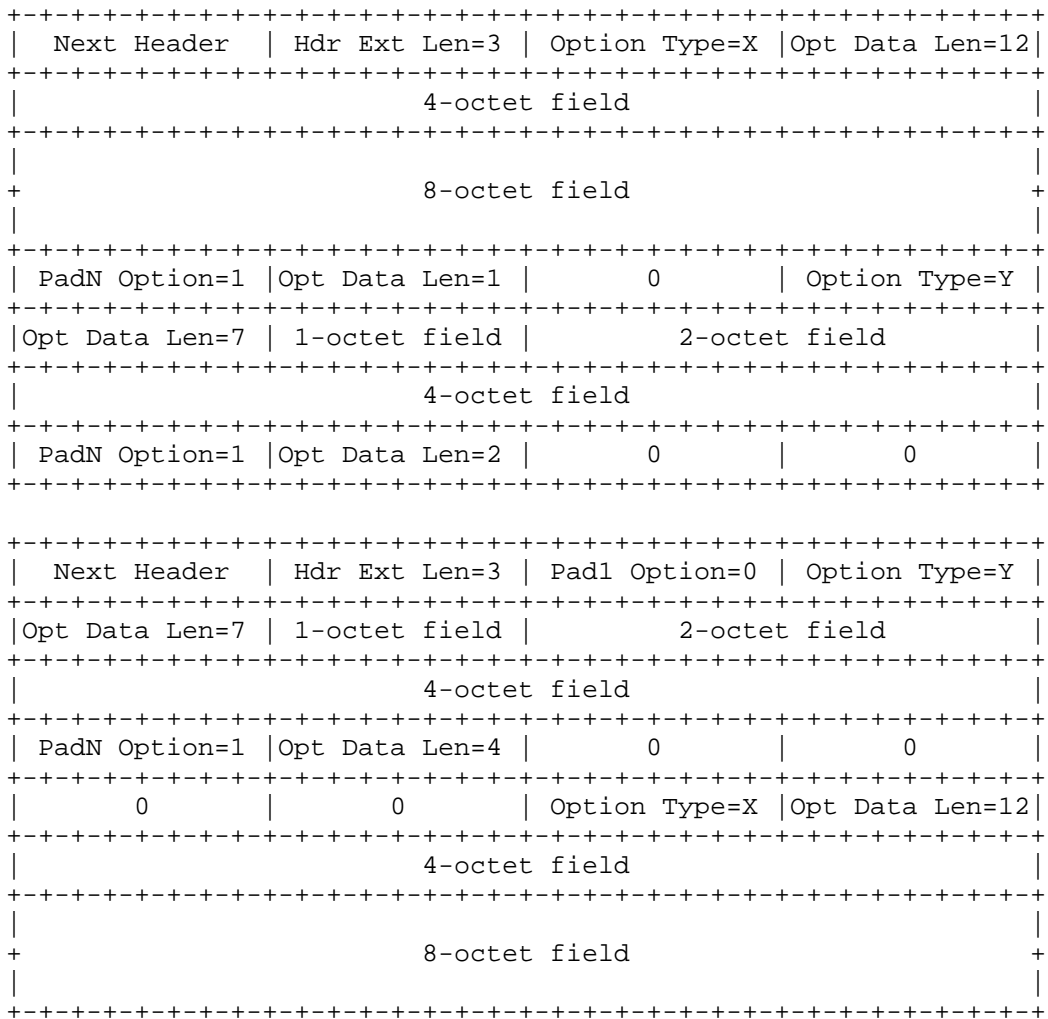


Its alignment requirement is $4n+3$, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:



Example 3

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



Appendix B. Changes Since RFC2460

This memo has the following changes from RFC2460.

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

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

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

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

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

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

B.1. Change History Since RFC2460

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

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

Working Group Internet Drafts

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

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

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

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

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

Individual Internet Drafts

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

07) Editorial changes.

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

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

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

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

06) Editorial changes.

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

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

05) Editorial changes.

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

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

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

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

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

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

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

Authors' Addresses

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

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

Email: bob.hinden@gmail.com

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

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

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

Abstract

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

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

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

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

Table of Contents

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

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

1. Introduction

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

2. IPv6 Addressing

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

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

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

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

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

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

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

2.1. Addressing Model

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

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

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

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

2.2. Text Representation of IPv6 Addresses

2.2.1. Text Representation of Addresses

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

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

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

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

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

For example, the following addresses

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

may be represented as

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

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

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

or in compressed form:

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

2.2.2. Text Representation of Address Prefixes

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

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

where

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

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

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

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

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

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

The following are NOT legal representations of the above prefix:

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

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

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

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

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

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

2.2.3. Recommendation for outputting IPv6 addresses

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

The recommendations are as follows:

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

```
2001:0db8::0001
```

is not correct and must be represented as

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

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

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

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

must be shortened to

2001:db8::2:1

Likewise,

2001:db8::0:1

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

2001:db8::1.

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

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

the sequence with three consecutive zero fields is shortened to

2001:0:0:1::1

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

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

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

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

is the correct representation.

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

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

is correct, but

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

is not correct.

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

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

should be shown as

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

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

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

should be shown as

::ffff:192.0.2.1

2.3. Address Type Identification

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

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

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

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

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

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

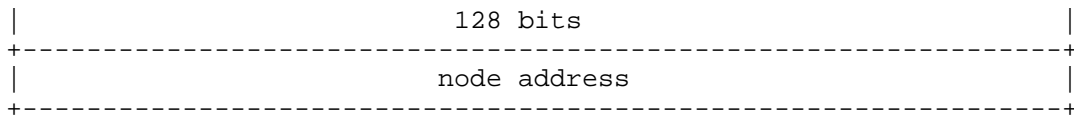
2.4. Unicast Addresses

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

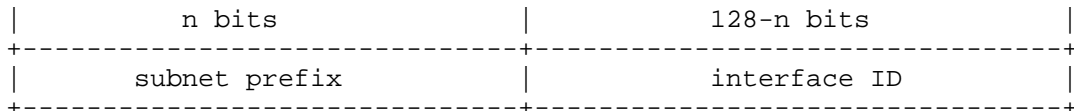
IPv6 unicast routing is based on prefixes of any valid length up to 128 [BCP198].

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

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



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



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

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

2.4.1. Interface Identifiers

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

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

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

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

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

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

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

2.4.2. The Unspecified Address

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

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

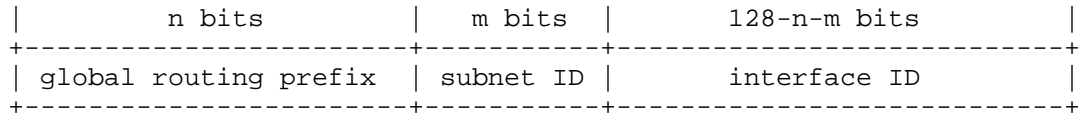
2.4.3. The Loopback Address

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

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

2.4.4. Global Unicast Addresses

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



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

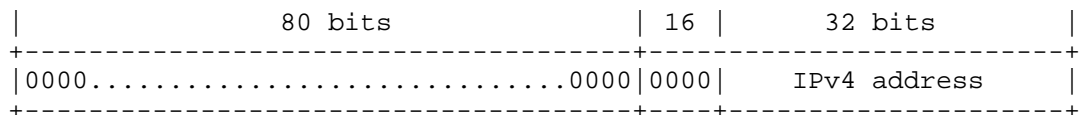
Examples of Global Unicast addresses that start with binary 000 are the IPv6 address with embedded IPv4 addresses described in Section 2.4.5. An example of global addresses starting with a binary value other than 000 (and therefore having a 64-bit interface ID field) can be found in [RFC3587].

2.4.5. IPv6 Addresses with Embedded IPv4 Addresses

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

2.4.5.1. IPv4-Compatible IPv6 Address

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

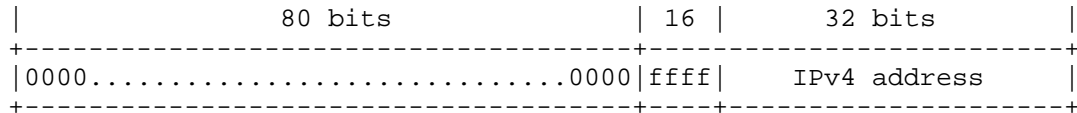


Note: The IPv4 address used in the "IPv4-Compatible IPv6 address" must be a globally-unique IPv4 unicast address.

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

2.4.5.2. IPv4-Mapped IPv6 Address

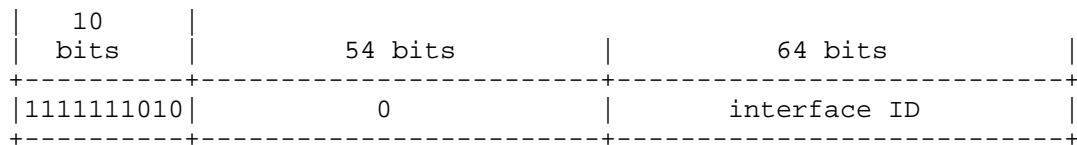
A second type of IPv6 address that holds an embedded IPv4 address is defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. The format of the "IPv4-mapped IPv6 address" is as follows:



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

2.4.6. Link-Local IPv6 Unicast Addresses

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



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

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

2.4.7. Other Local Unicast IPv6 Addresses

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

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

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

2.5. Anycast Addresses

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

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

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

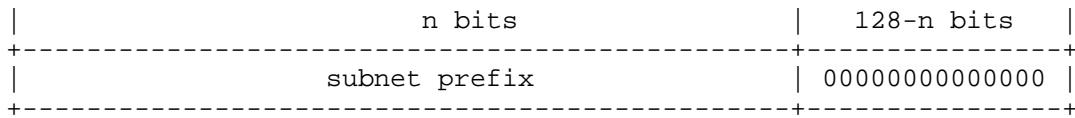
Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire Internet, which presents a severe scaling limit on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

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

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

2.5.1. Required Anycast Address

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



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

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

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

2.6. Multicast Addresses

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



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

flgs is a set of 4 flags: +---+---+
 |0|R|P|T|
 +---+---+

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.6.1. Pre-Defined Multicast Addresses

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

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

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

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

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

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

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

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

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

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

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

to

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

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

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

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

2.7. A Node's Required Addresses

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

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

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

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

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

3. IANA Considerations

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

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

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

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

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

4. Security Considerations

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

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

5. Acknowledgments

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

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

6. References

6.1. Normative References

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

6.2. Informative References

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

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

- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<http://www.rfc-editor.org/info/rfc4632>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC5942] Singh, H., Beebe, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <<http://www.rfc-editor.org/info/rfc5942>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<http://www.rfc-editor.org/info/rfc5952>>.
- [RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, DOI 10.17487/RFC6164, April 2011, <<http://www.rfc-editor.org/info/rfc6164>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<http://www.rfc-editor.org/info/rfc7421>>.
- [RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <<http://www.rfc-editor.org/info/rfc7428>>.

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

Appendix A. Modified EUI-64 Format Interface Identifiers

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

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

```

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

```

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

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

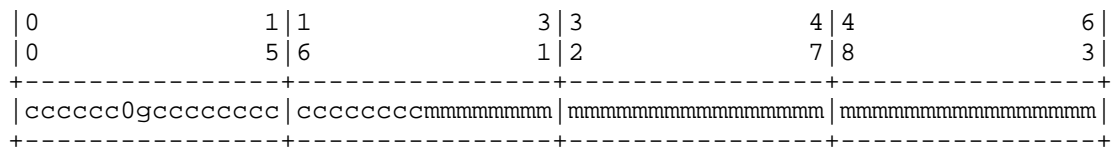
IPv6 nodes are not required to validate that interface identifiers created with modified EUI-64 tokens with the "u" bit set to universal are unique.

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

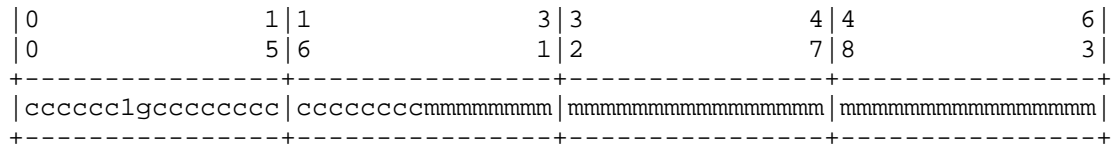
Depending on the characteristics of a specific link or node, there are a number of approaches for creating Modified EUI-64 format interface identifiers. This appendix describes some of these approaches.

Links or Nodes with IEEE EUI-64 Identifiers

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



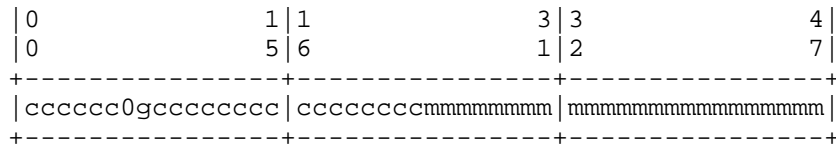
where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate universal scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The IPv6 interface identifier would be of the form:



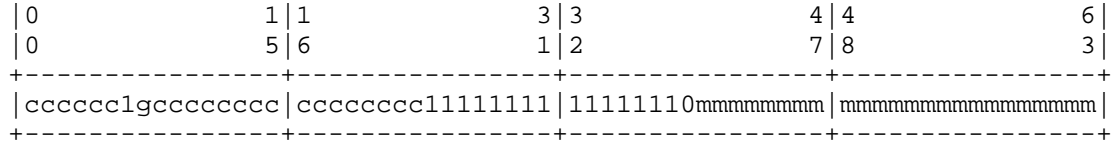
The only change is inverting the value of the universal/local bit.

Links or Nodes with IEEE 802 48-bit MACs

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



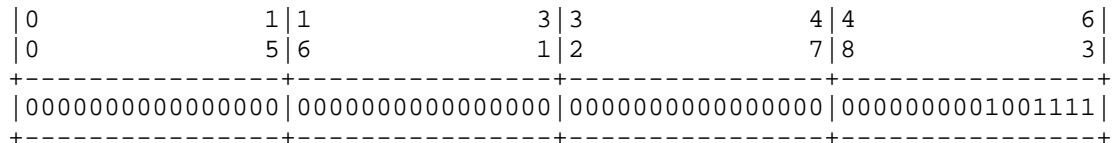
where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate Global scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The interface identifier would be of the form:



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

Links with Other Kinds of Identifiers

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



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

Links without Identifiers

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

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

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

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

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

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

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

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

Appendix B. CHANGES SINCE RFC 4291

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- o Editorial changes.

B.1. Change History Since RFC4291

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

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

Working Group Internet Drafts

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

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

00) Working Group Draft.

00) Editorial changes.

Individual Internet Drafts

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

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

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

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

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

03) Editorial changes.

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

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

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

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

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

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

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

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

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

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

- 02) Update references to current versions.
- 02) Editorial changes.
- 01) Incorporate the updates made by RFC5952 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case.
- 01) Revise this Section to document to show the changes from RFC4291.
- 01) Editorial changes.
- 00) Establish a baseline from RFC4291. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

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

Email: bob.hinden@gmail.com

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Internet Engineering Task Force
Internet-Draft
Obsoletes: 6434 (if approved)
Intended status: Best Current Practice
Expires: September 20, 2018

T. Chown
Jisc
J. Loughney
Intel
T. Winters
UNH-IOL
March 19, 2018

IPv6 Node Requirements
draft-ietf-6man-rfc6434-bis-08

Abstract

This document defines requirements for IPv6 nodes. It is expected that IPv6 will be deployed in a wide range of devices and situations. Specifying the requirements for IPv6 nodes allows IPv6 to function well and interoperate in a large number of situations and deployments.

This document obsoletes RFC 6434, and in turn RFC 4294.

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 September 20, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Scope of This Document	4
1.2.	Description of IPv6 Nodes	5
2.	Requirements Language	5
3.	Abbreviations Used in This Document	5
4.	Sub-IP Layer	5
5.	IP Layer	6
5.1.	Internet Protocol Version 6 - RFC 8200	6
5.2.	Support for IPv6 Extension Headers	7
5.3.	Protecting a node from excessive EH options	8
5.4.	Neighbor Discovery for IPv6 - RFC 4861	9
5.5.	SEcure Neighbor Discovery (SEND) - RFC 3971	10
5.6.	IPv6 Router Advertisement Flags Option - RFC 5175	11
5.7.	Path MTU Discovery and Packet Size	11
5.7.1.	Path MTU Discovery - RFC 8201	11
5.7.2.	Minimum MTU considerations	12
5.8.	ICMP for the Internet Protocol Version 6 (IPv6) - RFC 4443	12
5.9.	Default Router Preferences and More-Specific Routes - RFC 4191	12
5.10.	First-Hop Router Selection - RFC 8028	12
5.11.	Multicast Listener Discovery (MLD) for IPv6 - RFC 3810	12
5.12.	Explicit Congestion Notification (ECN) - RFC 3168	13
6.	Addressing and Address Configuration	13
6.1.	IP Version 6 Addressing Architecture - RFC 4291	13
6.2.	Host Address Availability Recommendations	13
6.3.	IPv6 Stateless Address Autoconfiguration - RFC 4862	14
6.4.	Privacy Extensions for Address Configuration in IPv6 - RFC 4941	15
6.5.	Stateful Address Autoconfiguration (DHCPv6) - RFC 3315	15
6.6.	Default Address Selection for IPv6 - RFC 6724	16
7.	DNS	16
8.	Configuring Non-Address Information	16
8.1.	DHCP for Other Configuration Information	16
8.2.	Router Advertisements and Default Gateway	17
8.3.	IPv6 Router Advertisement Options for DNS Configuration - RFC 8106	17
8.4.	DHCP Options versus Router Advertisement Options for Host Configuration	17
9.	Service Discovery Protocols	18

10. IPv4 Support and Transition	18
10.1. Transition Mechanisms	18
10.1.1. Basic Transition Mechanisms for IPv6 Hosts and Routers - RFC 4213	18
11. Application Support	18
11.1. Textual Representation of IPv6 Addresses - RFC 5952 . .	18
11.2. Application Programming Interfaces (APIs)	18
12. Mobility	19
13. Security	20
13.1. Requirements	21
13.2. Transforms and Algorithms	21
14. Router-Specific Functionality	22
14.1. IPv6 Router Alert Option - RFC 2711	22
14.2. Neighbor Discovery for IPv6 - RFC 4861	22
14.3. Stateful Address Autoconfiguration (DHCPv6) - RFC 3315 .	22
14.4. IPv6 Prefix Length Recommendation for Forwarding - BCP 198	23
15. Constrained Devices	23
16. Network Management	23
16.1. Management Information Base (MIB) Modules	23
16.1.1. IP Forwarding Table MIB	24
16.1.2. Management Information Base for the Internet Protocol (IP)	24
16.1.3. Interface MIB	24
16.2. YANG Data Models	24
16.2.1. IP Management YANG Model	24
16.2.2. Interface Management YANG Model	24
17. Security Considerations	24
18. IANA Considerations	24
19. Authors and Acknowledgments	25
19.1. Authors and Acknowledgments (Current Document)	25
19.2. Authors and Acknowledgments from RFC 6434	25
19.3. Authors and Acknowledgments from RFC 4294	25
20. Appendix: Changes from RFC 6434	25
21. Appendix: Changes from RFC 4294	27
22. References	28
22.1. Normative References	28
22.2. Informative References	35
Authors' Addresses	40

1. Introduction

This document defines common functionality required by both IPv6 hosts and routers. Many IPv6 nodes will implement optional or additional features, but this document collects and summarizes requirements from other published Standards Track documents in one place.

This document tries to avoid discussion of protocol details and references RFCs for this purpose. This document is intended to be an applicability statement and to provide guidance as to which IPv6 specifications should be implemented in the general case and which specifications may be of interest to specific deployment scenarios. This document does not update any individual protocol document RFCs.

Although this document points to different specifications, it should be noted that in many cases, the granularity of a particular requirement will be smaller than a single specification, as many specifications define multiple, independent pieces, some of which may not be mandatory. In addition, most specifications define both client and server behavior in the same specification, while many implementations will be focused on only one of those roles.

This document defines a minimal level of requirement needed for a device to provide useful internet service and considers a broad range of device types and deployment scenarios. Because of the wide range of deployment scenarios, the minimal requirements specified in this document may not be sufficient for all deployment scenarios. It is perfectly reasonable (and indeed expected) for other profiles to define additional or stricter requirements appropriate for specific usage and deployment environments. For example, this document does not mandate that all clients support DHCP, but some deployment scenarios may deem it appropriate to make such a requirement. For example, government agencies in the USA have defined profiles for specialized requirements for IPv6 in target environments (see [USGv6]).

As it is not always possible for an implementer to know the exact usage of IPv6 in a node, an overriding requirement for IPv6 nodes is that they should adhere to Jon Postel's Robustness Principle: "Be conservative in what you do, be liberal in what you accept from others" [RFC0793].

1.1. Scope of This Document

IPv6 covers many specifications. It is intended that IPv6 will be deployed in many different situations and environments. Therefore, it is important to develop requirements for IPv6 nodes to ensure interoperability.

This document assumes that all IPv6 nodes meet the minimum requirements specified here.

1.2. Description of IPv6 Nodes

From the Internet Protocol, Version 6 (IPv6) Specification [RFC8200], we have the following definitions:

IPv6 node - a device that implements IPv6.
IPv6 router - a node that forwards IPv6 packets not explicitly addressed to itself.
IPv6 host - any node that is not a router.

2. Requirements Language

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

3. Abbreviations Used in This Document

AH Authentication Header
DAD Duplicate Address Detection
ESP Encapsulating Security Payload
ICMP Internet Control Message Protocol
IKE Internet Key Exchange
MIB Management Information Base
MLD Multicast Listener Discovery
MTU Maximum Transmission Unit
NA Neighbor Advertisement
NBMA Non-Broadcast Multiple Access
ND Neighbor Discovery
NS Neighbor Solicitation
NUD Neighbor Unreachability Detection
PPP Point-to-Point Protocol

4. Sub-IP Layer

An IPv6 node must include support for one or more IPv6 link-layer specifications. Which link-layer specifications an implementation should include will depend upon what link-layers are supported by the hardware available on the system. It is possible for a conformant IPv6 node to support IPv6 on some of its interfaces and not on others.

As IPv6 is run over new layer 2 technologies, it is expected that new specifications will be issued. In the following, we list some of the layer 2 technologies for which an IPv6 specification has been developed. It is provided for informational purposes only and may not be complete.

- Transmission of IPv6 Packets over Ethernet Networks [RFC2464]
- Transmission of IPv6 Packets over Frame Relay Networks Specification [RFC2590]
- Transmission of IPv6 Packets over IEEE 1394 Networks [RFC3146]
- Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel [RFC4338]
- Transmission of IPv6 Packets over IEEE 802.15.4 Networks [RFC4944]
- Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks [RFC5121]
- IP version 6 over PPP [RFC5072]

In addition to traditional physical link-layers, it is also possible to tunnel IPv6 over other protocols. Examples include:

- Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs) [RFC4380]
- Section 3 of "Basic Transition Mechanisms for IPv6 Hosts and Routers" [RFC4213]

5. IP Layer

5.1. Internet Protocol Version 6 - RFC 8200

The Internet Protocol Version 6 is specified in [RFC8200]. This specification MUST be supported.

The node MUST follow the packet transmission rules in RFC 8200.

All conformant IPv6 implementations MUST be capable of sending and receiving IPv6 packets; forwarding functionality MAY be supported. Nodes MUST always be able to send, receive, and process fragment headers.

IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more of its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. See [RFC5722] for more information.

As recommended in [RFC8021], nodes MUST NOT generate atomic fragments, i.e., where the fragment is a whole datagram. As per

[RFC6946], if a receiving node reassembling a datagram encounters an atomic fragment, it should be processed as a fully reassembled packet, and any other fragments that match this packet should be processed independently.

To mitigate a variety of potential attacks, nodes SHOULD avoid using predictable fragment Identification values in Fragment Headers, as discussed in [RFC7739].

All nodes SHOULD support the setting and use of the IPv6 Flow Label field as defined in the IPv6 Flow Label specification [RFC6437]. Forwarding nodes such as routers and load distributors MUST NOT depend only on Flow Label values being uniformly distributed. It is RECOMMENDED that source hosts support the flow label by setting the Flow Label field for all packets of a given flow to the same value chosen from an approximation to a discrete uniform distribution.

5.2. Support for IPv6 Extension Headers

RFC 8200 specifies extension headers and the processing for these headers.

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.

Any unrecognized extension headers or options MUST be processed as described in RFC 8200. Note that where Section 4 of RFC 8200 refers to the action to be taken when a Next Header value in the current header is not recognized by a node, that action applies whether the value is an unrecognized Extension Header or an unrecognized upper layer protocol (ULP).

An IPv6 node MUST be able to process these headers. An exception is Routing Header type 0 (RH0), which was deprecated by [RFC5095] due to security concerns and which MUST be treated as an unrecognized routing type.

Further, [RFC7045] adds specific requirements for processing of Extension Headers, in particular that any forwarding node along an IPv6 packet's path, which forwards the packet for any reason, SHOULD do so regardless of any extension headers that are present.

As per RFC 8200, when a node fragments an IPv6 datagram, it MUST include the entire IPv6 Header Chain in the first fragment. The Per-Fragment headers must consist of the IPv6 header plus any extension

headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers. On reassembly, if the first fragment does not include all headers through an Upper-Layer header, then that fragment should be discarded and an ICMP Parameter Problem, Code 3, message should be sent to the source of the fragment, with the Pointer field set to zero. See [RFC7112] for a discussion of why oversized IPv6 Extension Header chains are avoided.

Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6 extension header can not be used for the desired new function, and in such cases need to follow the format described in Section 8 of RFC 8200. For further background reading on this topic, see [RFC6564].

5.3. Protecting a node from excessive EH options

As per RFC 8200, end hosts are expected to process all extension headers, destination options, and hop-by-hop options in a packet. Given that the only limit on the number and size of extension headers is the MTU, the processing of received packets could be considerable. It is also conceivable that a long chain of extension headers might be used as a form of denial-of-service attack. Accordingly, a host may place limits on the number and sizes of extension headers and options it is willing to process.

A host MAY limit the number of consecutive PAD1 options in destination options or hop-by-hop options to seven. In this case, if the more than seven consecutive PAD1 options are present the packet should be silently discarded. The rationale is that if padding of eight or more bytes is required than the PADN option should be used.

A host MAY limit number of bytes in a PADN option to be less than eight. In such a case, if a PADN option is present that has a length greater than seven then the packet should be silently discarded. The rationale for this guideline is that the purpose of padding is for alignment and eight bytes is the maximum alignment used in IPv6.

A host MAY disallow unknown options in destination options or hop-by-hop options. This should be configurable where the default is to accept unknown options and process them per [RFC8200]. If a packet with unknown options is received and the host is configured to disallow them, then the packet should be silently discarded.

A host MAY impose a limit on the maximum number of non-padding options allowed in a 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 eight. 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 silently discarded.

A host MAY impose a limit on the maximum length of destination options or hop-by-hop options extension 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 destination or hop-by-hop options extension header exceeds the length limit, then the packet should be silently discarded.

5.4. Neighbor Discovery for IPv6 - RFC 4861

Neighbor Discovery is defined in [RFC4861]; the definition was updated by [RFC5942]. Neighbor Discovery SHOULD be supported. RFC 4861 states:

Unless specified otherwise (in a document that covers operating IP over a particular link type) this document applies to all link types. However, because ND uses link-layer multicast for some of its services, it is possible that on some link types (e.g., Non-Broadcast Multi-Access (NBMA) links), alternative protocols or mechanisms to implement those services will be specified (in the appropriate document covering the operation of IP over a particular link type). The services described in this document that are not directly dependent on multicast, such as Redirects, next-hop determination, Neighbor Unreachability Detection, etc., are expected to be provided as specified in this document. The details of how one uses ND on NBMA links are addressed in [RFC2491].

Some detailed analysis of Neighbor Discovery follows:

Router Discovery is how hosts locate routers that reside on an attached link. Hosts MUST support Router Discovery functionality.

Prefix Discovery is how hosts discover the set of address prefixes that define which destinations are on-link for an attached link. Hosts MUST support Prefix Discovery.

Hosts MUST also implement Neighbor Unreachability Detection (NUD) for all paths between hosts and neighboring nodes. NUD is not required

for paths between routers. However, all nodes MUST respond to unicast Neighbor Solicitation (NS) messages.

[RFC7048] discusses NUD, in particular cases where it behaves too impatiently. It states that if a node transmits more than a certain number of packets, then it SHOULD use the exponential backoff of the retransmit timer, up to a certain threshold point.

Hosts MUST support the sending of Router Solicitations and the receiving of Router Advertisements. The ability to understand individual Router Advertisement options is dependent on supporting the functionality making use of the particular option.

[RFC7559] discusses packet loss resiliency for Router Solicitations, and requires that nodes MUST use a specific exponential backoff algorithm for RS retransmissions.

All nodes MUST support the sending and receiving of Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages. NS and NA messages are required for Duplicate Address Detection (DAD).

Hosts SHOULD support the processing of Redirect functionality. Routers MUST support the sending of Redirects, though not necessarily for every individual packet (e.g., due to rate limiting). Redirects are only useful on networks supporting hosts. In core networks dominated by routers, Redirects are typically disabled. The sending of Redirects SHOULD be disabled by default on backbone routers. They MAY be enabled by default on routers intended to support hosts on edge networks.

"IPv6 Host-to-Router Load Sharing" [RFC4311] includes additional recommendations on how to select from a set of available routers. [RFC4311] SHOULD be supported.

5.5. SEcure Neighbor Discovery (SEND) - RFC 3971

SEND [RFC3971] and Cryptographically Generated Addresses (CGAs) [RFC3972] provide a way to secure the message exchanges of Neighbor Discovery. SEND has the potential to address certain classes of spoofing attacks, but it does not provide specific protection for threats from off-link attackers.

There have been relatively few implementations of SEND in common operating systems and platforms since its publication in 2005, and thus deployment experience remains very limited to date.

At this time, support for SEND is considered optional. Due to the complexity in deploying SEND, and its heavyweight provisioning, its

deployment is only likely to be considered where nodes are operating in a particularly strict security environment.

5.6. IPv6 Router Advertisement Flags Option - RFC 5175

Router Advertisements include an 8-bit field of single-bit Router Advertisement flags. The Router Advertisement Flags Option extends the number of available flag bits by 48 bits. At the time of this writing, 6 of the original 8 single-bit flags have been assigned, while 2 remain available for future assignment. No flags have been defined that make use of the new option, and thus, strictly speaking, there is no requirement to implement the option today. However, implementations that are able to pass unrecognized options to a higher-level entity that may be able to understand them (e.g., a user-level process using a "raw socket" facility) MAY take steps to handle the option in anticipation of a future usage.

5.7. Path MTU Discovery and Packet Size

5.7.1. Path MTU Discovery - RFC 8201

"Path MTU Discovery for IP version 6" [RFC8201] SHOULD be supported. From [RFC8200]:

It is strongly recommended that IPv6 nodes implement Path MTU Discovery [RFC8201], in order to discover and take advantage of path MTUs greater than 1280 octets. However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of Path MTU Discovery.

The rules in [RFC8200] and [RFC5722] MUST be followed for packet fragmentation and reassembly.

As described in RFC 8201, nodes implementing Path MTU Discovery and sending packets larger than the IPv6 minimum link MTU are susceptible to problematic connectivity if ICMPv6 messages are blocked or not transmitted. For example, this will result in connections that complete the TCP three-way handshake correctly but then hang when data is transferred. This state is referred to as a black-hole connection [RFC2923]. Path MTU Discovery relies on ICMPv6 Packet Too Big (PTB) to determine the MTU of the path (and thus these should not be filtered, as per the recommendation in [RFC4890]).

An extension to Path MTU Discovery defined in RFC 8201 can be found in [RFC4821], which defines a method for Packetization Layer Path MTU Discovery (PLPMTUD) designed for use over paths where delivery of ICMPv6 messages to a host is not assured.

5.7.2. Minimum MTU considerations

While an IPv6 link MTU can be set to 1280 bytes, it is recommended that for IPv6 UDP in particular, which includes DNS operation, the sender use a large MTU if they can, in order to avoid gratuitous fragmentation-caused packet drops.

5.8. ICMP for the Internet Protocol Version 6 (IPv6) - RFC 4443

ICMPv6 [RFC4443] MUST be supported. "Extended ICMP to Support Multi-Part Messages" [RFC4884] MAY be supported.

5.9. Default Router Preferences and More-Specific Routes - RFC 4191

"Default Router Preferences and More-Specific Routes" [RFC4191] provides support for nodes attached to multiple (different) networks, each providing routers that advertise themselves as default routers via Router Advertisements. In some scenarios, one router may provide connectivity to destinations the other router does not, and choosing the "wrong" default router can result in reachability failures. In order to resolve this scenario IPv6 Nodes MUST implement [RFC4191] and SHOULD implement the Type C host role defined in RFC4191.

5.10. First-Hop Router Selection - RFC 8028

In multihomed scenarios, where a host has more than one prefix, each allocated by an upstream network that is assumed to implement BCP 38 ingress filtering, the host may have multiple routers to choose from.

Hosts that may be deployed in such multihomed environments SHOULD follow the guidance given in [RFC8028].

5.11. Multicast Listener Discovery (MLD) for IPv6 - RFC 3810

Nodes that need to join multicast groups MUST support MLDv2 [RFC3810]. MLD is needed by any node that is expected to receive and process multicast traffic and in particular MLDv2 is required for support for source-specific multicast (SSM) as per [RFC4607].

Previous versions of this document only required MLDv1 ([RFC2710]) to be implemented on all nodes. Since participation of any MLDv1-only nodes on a link require that all other nodes on the link then operate in version 1 compatibility mode, the requirement to support MLDv2 on all nodes was upgraded to a MUST. Further, SSM is now the preferred multicast distribution method, rather than ASM.

Note that Neighbor Discovery (as used on most link types -- see Section 5.4) depends on multicast and requires that nodes join Solicited Node multicast addresses.

5.12. Explicit Congestion Notification (ECN) - RFC 3168

An ECN-aware router may set a mark in the IP header in order to signal impending congestion, rather than dropping a packet. The receiver of the packet echoes the congestion indication to the sender, which can then reduce its transmission rate as if it detected a dropped packet.

Nodes that may be deployed in environments where they would benefit from such early congestion notification SHOULD implement [RFC3168]. In such cases, the updates presented in [RFC8311] may also be relevant.

6. Addressing and Address Configuration

6.1. IP Version 6 Addressing Architecture - RFC 4291

The IPv6 Addressing Architecture [RFC4291] MUST be supported.

The current IPv6 Address Architecture is based on a 64-bit boundary for subnet prefixes. The reasoning behind this decision is documented in [RFC7421].

Implementations MUST also support the Multicast flag updates documented in [RFC7371]

6.2. Host Address Availability Recommendations

Hosts may be configured with addresses through a variety of methods, including SLAAC, DHCPv6, or manual configuration.

[RFC7934] recommends that networks provide general-purpose end hosts with multiple global IPv6 addresses when they attach, and it describes the benefits of and the options for doing so. Routers SHOULD support [RFC7934] for assigning multiple address to a host. Host SHOULD support assigning multiple addresses as described in [RFC7934].

Nodes SHOULD support the capability to be assigned a prefix per host as documented in [RFC8273]. Such an approach can offer improved host isolation and enhanced subscriber management on shared network segments.

6.3. IPv6 Stateless Address Autoconfiguration - RFC 4862

Hosts MUST support IPv6 Stateless Address Autoconfiguration. It is recommended, as described in [RFC8064], that unless there is a specific requirement for MAC addresses to be embedded in an IID, nodes follow the procedure in [RFC7217] to generate SLAAC-based addresses, rather than using [RFC4862]. Addresses generated through RFC7217 will be the same whenever a given device (re)appears on the same subnet (with a specific IPv6 prefix), but the IID will vary on each subnet visited.

Nodes that are routers MUST be able to generate link-local addresses as described in [RFC4862].

From RFC 4862:

The autoconfiguration process specified in this document applies only to hosts and not routers. Since host autoconfiguration uses information advertised by routers, routers will need to be configured by some other means. However, it is expected that routers will generate link-local addresses using the mechanism described in this document. In addition, routers are expected to successfully pass the Duplicate Address Detection procedure described in this document on all addresses prior to assigning them to an interface.

All nodes MUST implement Duplicate Address Detection. Quoting from Section 5.4 of RFC 4862:

Duplicate Address Detection MUST be performed on all unicast addresses prior to assigning them to an interface, regardless of whether they are obtained through stateless autoconfiguration, DHCPv6, or manual configuration, with the following [exceptions noted therein].

"Optimistic Duplicate Address Detection (DAD) for IPv6" [RFC4429] specifies a mechanism to reduce delays associated with generating addresses via Stateless Address Autoconfiguration [RFC4862]. RFC 4429 was developed in conjunction with Mobile IPv6 in order to reduce the time needed to acquire and configure addresses as devices quickly move from one network to another, and it is desirable to minimize transition delays. For general purpose devices, RFC 4429 remains optional at this time.

[RFC7527] discusses enhanced DAD, and describes an algorithm to automate the detection of looped back IPv6 ND messages used by DAD. Nodes SHOULD implement this behaviour where such detection is beneficial.

6.4. Privacy Extensions for Address Configuration in IPv6 - RFC 4941

A node using Stateless Address Autoconfiguration [RFC4862] to form a globally unique IPv6 address using its MAC address to generate the IID will see that IID remain the same on any visited network, even though the network prefix part changes. Thus it is possible for 3rd party devices such nodes communicate with to track the activities of the node as it moves around the network. Privacy Extensions for Stateless Address Autoconfiguration [RFC4941] address this concern by allowing nodes to configure an additional temporary address where the IID is effectively randomly generated. Privacy addresses are then used as source addresses for new communications initiated by the node.

General issues regarding privacy issues for IPv6 addressing are discussed in [RFC7721].

RFC 4941 SHOULD be supported. In some scenarios, such as dedicated servers in a data center, it provides limited or no benefit, or may complicate network management. Thus devices implementing this specification MUST provide a way for the end user to explicitly enable or disable the use of such temporary addresses.

Note that RFC4941 can be used independently of traditional SLAAC, or of RFC7217-based SLAAC.

Implementers of RFC 4941 should be aware that certain addresses are reserved and should not be chosen for use as temporary addresses. Consult "Reserved IPv6 Interface Identifiers" [RFC5453] for more details.

6.5. Stateful Address Autoconfiguration (DHCPv6) - RFC 3315

DHCPv6 [RFC3315] can be used to obtain and configure addresses. In general, a network may provide for the configuration of addresses through SLAAC, DHCPv6, or both. There will be a wide range of IPv6 deployment models and differences in address assignment requirements, some of which may require DHCPv6 for stateful address assignment. Consequently, all hosts SHOULD implement address configuration via DHCPv6.

In the absence of observed Router Advertisement messages, IPv6 nodes MAY initiate DHCP to obtain IPv6 addresses and other configuration information, as described in Section 5.5.2 of [RFC4862].

Where devices are likely to be carried by users and attached to multiple visited networks, DHCPv6 client anonymity profiles SHOULD be supported as described in [RFC7844] to minimise the disclosure of

identifying information. Section 5 of RFC7844 describes operational considerations on the use of such anonymity profiles.

6.6. Default Address Selection for IPv6 - RFC 6724

IPv6 nodes will invariably have multiple addresses configured simultaneously, and thus will need to choose which addresses to use for which communications. The rules specified in the Default Address Selection for IPv6 [RFC6724] document MUST be implemented. [RFC8028] updates rule 5.5 from [RFC6724]; implementations SHOULD implement this rule.

7. DNS

DNS is described in [RFC1034], [RFC1035], [RFC3363], and [RFC3596]. Not all nodes will need to resolve names; those that will never need to resolve DNS names do not need to implement resolver functionality. However, the ability to resolve names is a basic infrastructure capability on which applications rely, and most nodes will need to provide support. All nodes SHOULD implement stub-resolver [RFC1034] functionality, as in [RFC1034], Section 5.3.1, with support for:

- AAAA type Resource Records [RFC3596];
- reverse addressing in ip6.arpa using PTR records [RFC3596];
- Extension Mechanisms for DNS (EDNS0) [RFC6891] to allow for DNS packet sizes larger than 512 octets.

Those nodes are RECOMMENDED to support DNS security extensions [RFC4033] [RFC4034] [RFC4035].

A6 Resource Records, which were only ever defined with Experimental status in [RFC3363], are now classified as Historic, as per [RFC6563].

8. Configuring Non-Address Information

8.1. DHCP for Other Configuration Information

DHCP [RFC3315] Specifies a mechanism for IPv6 nodes to obtain address configuration information (see Section 6.5) and to obtain additional (non-address) configuration. If a host implementation supports applications or other protocols that require configuration that is only available via DHCP, hosts SHOULD implement DHCP. For specialized devices on which no such configuration need is present, DHCP may not be necessary.

An IPv6 node can use the subset of DHCP (described in [RFC3736]) to obtain other configuration information.

If an IPv6 node implements DHCP it MUST implement the DNS options [RFC3646] as most deployments will expect these options are available.

8.2. Router Advertisements and Default Gateway

There is no defined DHCPv6 Gateway option.

Nodes using the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) are thus expected to determine their default router information and on-link prefix information from received Router Advertisements.

8.3. IPv6 Router Advertisement Options for DNS Configuration - RFC 8106

Router Advertisement Options have historically been limited to those that are critical to basic IPv6 functionality. Originally, DNS configuration was not included as an RA option, and DHCP was the recommended way to obtain DNS configuration information. Over time, the thinking surrounding such an option has evolved. It is now generally recognized that few nodes can function adequately without having access to a working DNS resolver, and thus a Standards Track document has been published to provide this capability [RFC8106].

Implementations MUST include support for the DNS RA option [RFC8106].

8.4. DHCP Options versus Router Advertisement Options for Host Configuration

In IPv6, there are two main protocol mechanisms for propagating configuration information to hosts: Router Advertisements (RAs) and DHCP. RA options have been restricted to those deemed essential for basic network functioning and for which all nodes are configured with exactly the same information. Examples include the Prefix Information Options, the MTU option, etc. On the other hand, DHCP has generally been preferred for configuration of more general parameters and for parameters that may be client-specific. Generally speaking, however, there has been a desire to define only one mechanism for configuring a given option, rather than defining multiple (different) ways of configuring the same information.

One issue with having multiple ways of configuring the same information is that interoperability suffers if a host chooses one mechanism but the network operator chooses a different mechanism. For "closed" environments, where the network operator has significant influence over what devices connect to the network and thus what

configuration mechanisms they support, the operator may be able to ensure that a particular mechanism is supported by all connected hosts. In more open environments, however, where arbitrary devices may connect (e.g., a WIFI hotspot), problems can arise. To maximize interoperability in such environments, hosts would need to implement multiple configuration mechanisms to ensure interoperability.

9. Service Discovery Protocols

[RFC6762] and [RFC6763] describe multicast DNS (mDNS) and DNS-Based Service Discovery (DNS-SD) respectively. These protocols, collectively commonly referred to as the 'Bonjour' protocols after their naming by Apple, provide the means for devices to discover services within a local link and, in the absence of a unicast DNS service, to exchange naming information.

Where devices are to be deployed in networks where service discovery would be beneficial, e.g., for users seeking to discover printers or display devices, mDNS and DNS-SD SHOULD be supported.

The IETF dnssd WG is defining solutions for DNS-based service discovery in multi-link networks.

10. IPv4 Support and Transition

IPv6 nodes MAY support IPv4.

10.1. Transition Mechanisms

10.1.1. Basic Transition Mechanisms for IPv6 Hosts and Routers - RFC 4213

If an IPv6 node implements dual stack and tunneling, then [RFC4213] MUST be supported.

11. Application Support

11.1. Textual Representation of IPv6 Addresses - RFC 5952

Software that allows users and operators to input IPv6 addresses in text form SHOULD support "A Recommendation for IPv6 Address Text Representation" [RFC5952].

11.2. Application Programming Interfaces (APIs)

There are a number of IPv6-related APIs. This document does not mandate the use of any, because the choice of API does not directly relate to on-the-wire behavior of protocols. Implementers, however,

would be advised to consider providing a common API or reviewing existing APIs for the type of functionality they provide to applications.

"Basic Socket Interface Extensions for IPv6" [RFC3493] provides IPv6 functionality used by typical applications. Implementers should note that RFC3493 has been picked up and further standardized by the Portable Operating System Interface (POSIX) [POSIX].

"Advanced Sockets Application Program Interface (API) for IPv6" [RFC3542] provides access to advanced IPv6 features needed by diagnostic and other more specialized applications.

"IPv6 Socket API for Source Address Selection" [RFC5014] provides facilities that allow an application to override the default Source Address Selection rules of [RFC6724].

"Socket Interface Extensions for Multicast Source Filters" [RFC3678] provides support for expressing source filters on multicast group memberships.

"Extension to Sockets API for Mobile IPv6" [RFC4584] provides application support for accessing and enabling Mobile IPv6 [RFC6275] features.

12. Mobility

Mobile IPv6 [RFC6275] and associated specifications [RFC3776] [RFC4877] allow a node to change its point of attachment within the Internet, while maintaining (and using) a permanent address. All communication using the permanent address continues to proceed as expected even as the node moves around. The definition of Mobile IP includes requirements for the following types of nodes:

- mobile nodes
- correspondent nodes with support for route optimization
- home agents
- all IPv6 routers

At the present time, Mobile IP has seen only limited implementation and no significant deployment, partly because it originally assumed an IPv6-only environment rather than a mixed IPv4/IPv6 Internet. Recently, additional work has been done to support mobility in mixed-mode IPv4 and IPv6 networks [RFC5555].

More usage and deployment experience is needed with mobility before any specific approach can be recommended for broad implementation in all hosts and routers. Consequently, [RFC6275], [RFC5555], and associated standards such as [RFC4877] are considered a MAY at this time.

IPv6 for 3GPP [RFC7066] lists a snapshot of required IPv6 Functionalities at the time the document was published that would need to be implemented, going above and beyond the recommendations in this document. Additionally a 3GPP IPv6 Host MAY implement [RFC7278] for delivering IPv6 prefixes on the LAN link.

13. Security

This section describes the specification for security for IPv6 nodes.

Achieving security in practice is a complex undertaking. Operational procedures, protocols, key distribution mechanisms, certificate management approaches, etc., are all components that impact the level of security actually achieved in practice. More importantly, deficiencies or a poor fit in any one individual component can significantly reduce the overall effectiveness of a particular security approach.

IPsec either can provide end-to-end security between nodes or or can provide channel security (for example, via a site-to-site IPsec VPN), making it possible to provide secure communication for all (or a subset of) communication flows at the IP layer between pairs of internet nodes. IPsec has two standard operating modes, Tunnel-mode and Transport-mode. In Tunnel-mode, IPsec provides network-layer security and protects an entire IP packet by encapsulating the original IP packet and then pre-pending a new IP header. In Transport-mode, IPsec provides security for the transport-layer (and above) by encapsulating only the transport-layer (and above) portion of the IP packet (i.e., without adding a 2nd IP header).

Although IPsec can be used with manual keying in some cases, such usage has limited applicability and is not recommended.

A range of security technologies and approaches proliferate today (e.g., IPsec, Transport Layer Security (TLS), Secure SHell (SSH), SSL VPNS, etc.) No one approach has emerged as an ideal technology for all needs and environments. Moreover, IPsec is not viewed as the ideal security technology in all cases and is unlikely to displace the others.

Previously, IPv6 mandated implementation of IPsec and recommended the key management approach of IKE. This document updates that

recommendation by making support of the IPsec Architecture [RFC4301] a SHOULD for all IPv6 nodes. Note that the IPsec Architecture requires (e.g., Section 4.5 of RFC 4301) the implementation of both manual and automatic key management. Currently, the default automated key management protocol to implement is IKEv2 [RFC7296].

This document recognizes that there exists a range of device types and environments where approaches to security other than IPsec can be justified. For example, special-purpose devices may support only a very limited number or type of applications, and an application-specific security approach may be sufficient for limited management or configuration capabilities. Alternatively, some devices may run on extremely constrained hardware (e.g., sensors) where the full IPsec Architecture is not justified.

Because most common platforms now support IPv6 and have it enabled by default, IPv6 security is an issue for networks that are ostensibly IPv4-only; see [RFC7123] for guidance on this area.

13.1. Requirements

"Security Architecture for the Internet Protocol" [RFC4301] SHOULD be supported by all IPv6 nodes. Note that the IPsec Architecture requires (e.g., Section 4.5 of [RFC4301]) the implementation of both manual and automatic key management. Currently, the default automated key management protocol to implement is IKEv2. As required in [RFC4301], IPv6 nodes implementing the IPsec Architecture MUST implement ESP [RFC4303] and MAY implement AH [RFC4302].

13.2. Transforms and Algorithms

The current set of mandatory-to-implement algorithms for the IPsec Architecture are defined in "Cryptographic Algorithm Implementation Requirements For ESP and AH" [RFC8221]. IPv6 nodes implementing the IPsec Architecture MUST conform to the requirements in [RFC8221]. Preferred cryptographic algorithms often change more frequently than security protocols. Therefore, implementations MUST allow for migration to new algorithms, as RFC 8221 is replaced or updated in the future.

The current set of mandatory-to-implement algorithms for IKEv2 are defined in "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)" [RFC8247]. IPv6 nodes implementing IKEv2 MUST conform to the requirements in [RFC8247] and/or any future updates or replacements to [RFC8247].

14. Router-Specific Functionality

This section defines general host considerations for IPv6 nodes that act as routers. Currently, this section does not discuss detailed routing-specific requirements. For the case of typical home routers, [RFC7084] defines basic requirements for customer edge routers.

14.1. IPv6 Router Alert Option - RFC 2711

The IPv6 Router Alert Option [RFC2711] is an optional IPv6 Hop-by-Hop Header that is used in conjunction with some protocols (e.g., RSVP [RFC2205] or Multicast Listener Discovery (MLDv2) [RFC3810]). The Router Alert option will need to be implemented whenever such protocols that mandate its use are implemented. See Section 5.11.

14.2. Neighbor Discovery for IPv6 - RFC 4861

Sending Router Advertisements and processing Router Solicitations MUST be supported.

Section 7 of [RFC6275] includes some mobility-specific extensions to Neighbor Discovery. Routers SHOULD implement Sections 7.3 and 7.5, even if they do not implement Home Agent functionality.

14.3. Stateful Address Autoconfiguration (DHCPv6) - RFC 3315

A single DHCP server ([RFC3315] or [RFC4862]) can provide configuration information to devices directly attached to a shared link, as well as to devices located elsewhere within a site. Communication between a client and a DHCP server located on different links requires the use of DHCP relay agents on routers.

In simple deployments, consisting of a single router and either a single LAN or multiple LANs attached to the single router, together with a WAN connection, a DHCP server embedded within the router is one common deployment scenario (e.g., [RFC7084]). There is no need for relay agents in such scenarios.

In more complex deployment scenarios, such as within enterprise or service provider networks, the use of DHCP requires some level of configuration, in order to configure relay agents, DHCP servers, etc. In such environments, the DHCP server might even be run on a traditional server, rather than as part of a router.

Because of the wide range of deployment scenarios, support for DHCP server functionality on routers is optional. However, routers targeted for deployment within more complex scenarios (as described above) SHOULD support relay agent functionality. Note that "Basic

Requirements for IPv6 Customer Edge Routers" [RFC7084] requires implementation of a DHCPv6 server function in IPv6 Customer Edge (CE) routers.

14.4. IPv6 Prefix Length Recommendation for Forwarding - BCP 198

Forwarding nodes MUST conform to BCP 198 [RFC7608] and thus IPv6 implementations of nodes that may forward packets MUST conform to the rules specified in Section 5.1 of [RFC4632].

15. Constrained Devices

The target for this document is general IPv6 nodes. In this Section, we briefly discuss considerations for constrained devices.

In the case of constrained nodes, with limited CPU, memory, bandwidth or power, support for certain IPv6 functionality may need to be considered due to those limitations. While the requirements of this document are RECOMMENDED for all nodes, including constrained nodes, compromises may need to be made in certain cases. Where such compromises are made, the interoperability of devices should be strongly considered, particularly where this may impact other nodes on the same link, e.g., only supporting MLDv1 will affect other nodes.

The IETF 6LowPAN (IPv6 over Low Power LWPAN) WG defined six RFCs, including a general overview and problem statement ([RFC4919], the means by which IPv6 packets are transmitted over IEEE 802.15.4 networks [RFC4944] and ND optimisations for that medium [RFC6775].

If an IPv6 node is concerned about the impact of IPv6 message power consumption, it SHOULD want to implement the recommendations in [RFC7772].

16. Network Management

Network management MAY be supported by IPv6 nodes. However, for IPv6 nodes that are embedded devices, network management may be the only possible way of controlling these nodes.

Existing network management protocols include SNMP [RFC3411], NETCONF [RFC6241] and RESTCONF [RFC8040].

16.1. Management Information Base (MIB) Modules

[RFC8096] clarifies the obsoleted status of various IPv6-specific MIB modules.

The following two MIB modules SHOULD be supported by nodes that support a Simple Network Management Protocol (SNMP) agent.

16.1.1. IP Forwarding Table MIB

The IP Forwarding Table MIB [RFC4292] SHOULD be supported by nodes that support an SNMP agent.

16.1.2. Management Information Base for the Internet Protocol (IP)

The IP MIB [RFC4293] SHOULD be supported by nodes that support an SNMP agent.

16.1.3. Interface MIB

The Interface MIB [RFC2863] SHOULD be supported by nodes the support an SNMP agent.

16.2. YANG Data Models

The following YANG data models SHOULD be supported by nodes that support a NETCONF or RESTCONF agent.

16.2.1. IP Management YANG Model

The IP Management YANG Model [I-D.ietf-netmod-rfc7277bis] SHOULD be supported by nodes that support NETCONF or RESTCONF.

16.2.2. Interface Management YANG Model

The Interface Management YANG Model [I-D.ietf-netmod-rfc7223bis] SHOULD be supported by nodes that support NETCONF or RESTCONF.

17. Security Considerations

This document does not directly affect the security of the Internet, beyond the security considerations associated with the individual protocols.

Security is also discussed in Section 13 above.

18. IANA Considerations

This document does not require any IANA actions.

19. Authors and Acknowledgments

19.1. Authors and Acknowledgments (Current Document)

For this version of the IPv6 Node Requirements document, the authors would like to thank Brian Carpenter, Dave Thaler, Tom Herbert, Erik Kline, Mohamed Boucadair, and Michayla Newcombe for their contributions.

19.2. Authors and Acknowledgments from RFC 6434

Ed Jankiewicz and Thomas Narten were named authors of the previous iteration of this document, RFC6434.

For this version of the document, the authors thanked Hitoshi Asaeda, Brian Carpenter, Tim Chown, Ralph Droms, Sheila Frankel, Sam Hartman, Bob Hinden, Paul Hoffman, Pekka Savola, Yaron Sheffer, and Dave Thaler.

19.3. Authors and Acknowledgments from RFC 4294

The original version of this document (RFC 4294) was written by the IPv6 Node Requirements design team, which had the following members: Jari Arkko, Marc Blanchet, Samita Chakrabarti, Alain Durand, Gerard Gastaud, Jun-ichiro Itojun Hagino, Atsushi Inoue, Masahiro Ishiyama, John Loughney, Rajiv Raghunarayan, Shoichi Sakane, Dave Thaler, and Juha Wiljakka.

The authors would like to thank Ran Atkinson, Jim Bound, Brian Carpenter, Ralph Droms, Christian Huitema, Adam Machalek, Thomas Narten, Juha Ollila, and Pekka Savola for their comments. Thanks to Mark Andrews for comments and corrections on DNS text. Thanks to Alfred Hoenes for tracking the updates to various RFCs.

20. Appendix: Changes from RFC 6434

There have been many editorial clarifications as well as significant additions and updates. While this section highlights some of the changes, readers should not rely on this section for a comprehensive list of all changes.

1. Restructured sections
2. Added 6LoWPAN to link layers as it has some deployment.
3. Removed DOD IPv6 Profile as it hasn't been updated.

4. Updated to MLDv2 support to a MUST since nodes are restricted if MLDv1 is used.
5. Require DNS RA Options so SLAAC-only devices can get DNS, RFC8106 is a MUST.
6. Require RFC3646 DNS Options for DHCPv6 implementations.
7. Added RESTCONF and NETCONF as possible options to Network management.
8. Added section on constrained devices.
9. Added text on RFC7934, address availability to hosts (SHOULD).
10. Added text on RFC7844, anonymity profiles for DHCPv6 clients.
11. mDNS and DNS-SD added as updated service discovery.
12. Added RFC8028 as a SHOULD as a method for solving multi-prefix network
13. Added ECN RFC3168 as a SHOULD, since recent reports have shown this as useful, and added a note on RFC8311, which is related.
14. Added reference to RFC7123 for Security over IPv4-only networks
15. Removed Jumbograms RFC2675 as they aren't deployed.
16. Updated Obseleted RFCs to the new version of the RFC including 2460, 1981, 7321, 4307
17. Added RFC7772 for power consumptions considerations
18. Added why /64 boundries for more detail - RFC 7421
19. Added a Unique IPv6 Prefix per Host to support currently deployed IPv6 networks
20. Clarified RFC7066 was snapshot for 3GPP
21. Updated 4191 as a MUST, SHOULD for Type C Host as it helps solve multi-prefix problem
22. Removed IPv6 over ATM since there aren't many deployments
23. Added a note in Section 6.6 for RFC6724 Section 5.5/

24. Added MUST for BCP 198 for forwarding IPv6 packets
 25. Added reference to RFC8064 for stable address creation.
 26. Added text on protection from excessive EH options
 27. Added text on dangers of 1280 MTU UDP, esp. wrt DNS traffic
 28. Added text to clarify RFC8200 behaviour for unrecognized EHs or unrecognized ULPs
 29. Removed dated email addresses from design team acknowledgements for RFC 4294.
21. Appendix: Changes from RFC 4294

There have been many editorial clarifications as well as significant additions and updates. While this section highlights some of the changes, readers should not rely on this section for a comprehensive list of all changes.

1. Updated the Introduction to indicate that this document is an applicability statement and is aimed at general nodes.
2. Significantly updated the section on Mobility protocols, adding references and downgrading previous SHOULDs to MAYs.
3. Changed Sub-IP Layer section to just list relevant RFCs, and added some more RFCs.
4. Added section on SEND (it is a MAY).
5. Revised section on Privacy Extensions [RFC4941] to add more nuance to recommendation.
6. Completely revised IPsec/IKEv2 section, downgrading overall recommendation to a SHOULD.
7. Upgraded recommendation of DHCPv6 to SHOULD.
8. Added background section on DHCP versus RA options, added SHOULD recommendation for DNS configuration via RAs (RFC6106), and cleaned up DHCP recommendations.
9. Added recommendation that routers implement Sections 7.3 and 7.5 of [RFC6275].
10. Added pointer to subnet clarification document [RFC5942].

11. Added text that "IPv6 Host-to-Router Load Sharing" [RFC4311] SHOULD be implemented.
 12. Added reference to [RFC5722] (Overlapping Fragments), and made it a MUST to implement.
 13. Made "A Recommendation for IPv6 Address Text Representation" [RFC5952] a SHOULD.
 14. Removed mention of "DNAME" from the discussion about [RFC3363].
 15. Numerous updates to reflect newer versions of IPv6 documents, including [RFC4443], [RFC4291], [RFC3596], and [RFC4213].
 16. Removed discussion of "Managed" and "Other" flags in RAs. There is no consensus at present on how to process these flags, and discussion of their semantics was removed in the most recent update of Stateless Address Autoconfiguration [RFC4862].
 17. Added many more references to optional IPv6 documents.
 18. Made "A Recommendation for IPv6 Address Text Representation" [RFC5952] a SHOULD.
 19. Added reference to [RFC5722] (Overlapping Fragments), and made it a MUST to implement.
 20. Updated MLD section to include reference to Lightweight MLD [RFC5790].
 21. Added SHOULD recommendation for "Default Router Preferences and More-Specific Routes" [RFC4191].
 22. Made "IPv6 Flow Label Specification" [RFC6437] a SHOULD.
22. References
- 22.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<https://www.rfc-editor.org/info/rfc2710>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<https://www.rfc-editor.org/info/rfc2711>>.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <<https://www.rfc-editor.org/info/rfc2863>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <<https://www.rfc-editor.org/info/rfc3411>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, DOI 10.17487/RFC3736, April 2004, <<https://www.rfc-editor.org/info/rfc3736>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <<https://www.rfc-editor.org/info/rfc4213>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292, DOI 10.17487/RFC4292, April 2006, <<https://www.rfc-editor.org/info/rfc4292>>.
- [RFC4293] Routhier, S., Ed., "Management Information Base for the Internet Protocol (IP)", RFC 4293, DOI 10.17487/RFC4293, April 2006, <<https://www.rfc-editor.org/info/rfc4293>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC4311] Hinden, R. and D. Thaler, "IPv6 Host-to-Router Load Sharing", RFC 4311, DOI 10.17487/RFC4311, November 2005, <<https://www.rfc-editor.org/info/rfc4311>>.

- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<https://www.rfc-editor.org/info/rfc4632>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <<https://www.rfc-editor.org/info/rfc5095>>.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <<https://www.rfc-editor.org/info/rfc5453>>.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <<https://www.rfc-editor.org/info/rfc5722>>.
- [RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <<https://www.rfc-editor.org/info/rfc5790>>.

- [RFC5942] Singh, H., Beebee, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <<https://www.rfc-editor.org/info/rfc5942>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<https://www.rfc-editor.org/info/rfc5952>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <<https://www.rfc-editor.org/info/rfc6564>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.

- [RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments", RFC 6946, DOI 10.17487/RFC6946, May 2013, <<https://www.rfc-editor.org/info/rfc6946>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/info/rfc7045>>.
- [RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, DOI 10.17487/RFC7048, January 2014, <<https://www.rfc-editor.org/info/rfc7048>>.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <<https://www.rfc-editor.org/info/rfc7112>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.
- [I-D.ietf-netmod-rfc7277bis]
Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-rfc7277bis-03 (work in progress), January 2018.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7527] Asati, R., Singh, H., Beebee, W., Pignataro, C., Dart, E., and W. George, "Enhanced Duplicate Address Detection", RFC 7527, DOI 10.17487/RFC7527, April 2015, <<https://www.rfc-editor.org/info/rfc7527>>.

- [RFC7559] Krishnan, S., Anipko, D., and D. Thaler, "Packet-Loss Resiliency for Router Solicitations", RFC 7559, DOI 10.17487/RFC7559, May 2015, <<https://www.rfc-editor.org/info/rfc7559>>.
- [RFC7608] Boucadair, M., Petrescu, A., and F. Baker, "IPv6 Prefix Length Recommendation for Forwarding", BCP 198, RFC 7608, DOI 10.17487/RFC7608, July 2015, <<https://www.rfc-editor.org/info/rfc7608>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [RFC8021] Gont, F., Liu, W., and T. Anderson, "Generation of IPv6 Atomic Fragments Considered Harmful", RFC 8021, DOI 10.17487/RFC8021, January 2017, <<https://www.rfc-editor.org/info/rfc8021>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <<https://www.rfc-editor.org/info/rfc8064>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <<https://www.rfc-editor.org/info/rfc8311>>.

22.2. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/info/rfc2464>>.
- [RFC2491] Armitage, G., Schulter, P., Jork, M., and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, DOI 10.17487/RFC2491, January 1999, <<https://www.rfc-editor.org/info/rfc2491>>.
- [RFC2590] Conta, A., Malis, A., and M. Mueller, "Transmission of IPv6 Packets over Frame Relay Networks Specification", RFC 2590, DOI 10.17487/RFC2590, May 1999, <<https://www.rfc-editor.org/info/rfc2590>>.
- [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <<https://www.rfc-editor.org/info/rfc2923>>.

- [RFC3146] Fujisawa, K. and A. Onoe, "Transmission of IPv6 Packets over IEEE 1394 Networks", RFC 3146, DOI 10.17487/RFC3146, October 2001, <<https://www.rfc-editor.org/info/rfc3146>>.
- [RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, DOI 10.17487/RFC3363, August 2002, <<https://www.rfc-editor.org/info/rfc3363>>.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <<https://www.rfc-editor.org/info/rfc3493>>.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <<https://www.rfc-editor.org/info/rfc3542>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, DOI 10.17487/RFC3678, January 2004, <<https://www.rfc-editor.org/info/rfc3678>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<https://www.rfc-editor.org/info/rfc6275>>.
- [RFC3776] Arkko, J., Devarapalli, V., and F. Dupont, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents", RFC 3776, DOI 10.17487/RFC3776, June 2004, <<https://www.rfc-editor.org/info/rfc3776>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<https://www.rfc-editor.org/info/rfc3972>>.

- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4338] DeSanti, C., Carlson, C., and R. Nixon, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel", RFC 4338, DOI 10.17487/RFC4338, January 2006, <<https://www.rfc-editor.org/info/rfc4338>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4584] Chakrabarti, S. and E. Nordmark, "Extension to Sockets API for Mobile IPv6", RFC 4584, DOI 10.17487/RFC4584, July 2006, <<https://www.rfc-editor.org/info/rfc4584>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.
- [RFC4877] Devarapalli, V. and F. Dupont, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture", RFC 4877, DOI 10.17487/RFC4877, April 2007, <<https://www.rfc-editor.org/info/rfc4877>>.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <<https://www.rfc-editor.org/info/rfc4884>>.
- [RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <<https://www.rfc-editor.org/info/rfc4890>>.

- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <<https://www.rfc-editor.org/info/rfc4919>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <<https://www.rfc-editor.org/info/rfc5014>>.
- [RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, DOI 10.17487/RFC5072, September 2007, <<https://www.rfc-editor.org/info/rfc5072>>.
- [RFC5121] Patil, B., Xia, F., Sarikaya, B., Choi, JH., and S. Madanapalli, "Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks", RFC 5121, DOI 10.17487/RFC5121, February 2008, <<https://www.rfc-editor.org/info/rfc5121>>.
- [RFC5555] Soliman, H., Ed., "Mobile IPv6 Support for Dual Stack Hosts and Routers", RFC 5555, DOI 10.17487/RFC5555, June 2009, <<https://www.rfc-editor.org/info/rfc5555>>.
- [RFC6563] Jiang, S., Conrad, D., and B. Carpenter, "Moving A6 to Historic Status", RFC 6563, DOI 10.17487/RFC6563, March 2012, <<https://www.rfc-editor.org/info/rfc6563>>.
- [RFC7066] Korhonen, J., Ed., Arkko, J., Ed., Savolainen, T., and S. Krishnan, "IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts", RFC 7066, DOI 10.17487/RFC7066, November 2013, <<https://www.rfc-editor.org/info/rfc7066>>.
- [RFC7084] Singh, H., Beebe, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<https://www.rfc-editor.org/info/rfc7084>>.
- [RFC7123] Gont, F. and W. Liu, "Security Implications of IPv6 on IPv4 Networks", RFC 7123, DOI 10.17487/RFC7123, February 2014, <<https://www.rfc-editor.org/info/rfc7123>>.

- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7371] Boucadair, M. and S. Venaas, "Updates to the IPv6 Multicast Addressing Architecture", RFC 7371, DOI 10.17487/RFC7371, September 2014, <<https://www.rfc-editor.org/info/rfc7371>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.
- [RFC7772] Yourtchenko, A. and L. Colitti, "Reducing Energy Consumption of Router Advertisements", BCP 202, RFC 7772, DOI 10.17487/RFC7772, February 2016, <<https://www.rfc-editor.org/info/rfc7772>>.
- [RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <<https://www.rfc-editor.org/info/rfc7844>>.
- [RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi, "Host Address Availability Recommendations", BCP 204, RFC 7934, DOI 10.17487/RFC7934, July 2016, <<https://www.rfc-editor.org/info/rfc7934>>.
- [RFC8096] Fenner, B., "The IPv6-Specific MIB Modules Are Obsolete", RFC 8096, DOI 10.17487/RFC8096, April 2017, <<https://www.rfc-editor.org/info/rfc8096>>.
- [RFC8273] Brzozowski, J. and G. Van de Velde, "Unique IPv6 Prefix per Host", RFC 8273, DOI 10.17487/RFC8273, December 2017, <<https://www.rfc-editor.org/info/rfc8273>>.
- [POSIX] IEEE, "IEEE Std. 1003.1-2008 Standard for Information Technology -- Portable Operating System Interface (POSIX), ISO/IEC 9945:2009", <<http://www.ieee.org>>.

[USGv6] National Institute of Standards and Technology, "A Profile for IPv6 in the U.S. Government - Version 1.0", July 2008, <<http://www.antd.nist.gov/usgv6/usgv6-v1.pdf>>.

Authors' Addresses

Tim Chown
Jisc
Lumen House, Library Avenue
Harwell Oxford, Didcot OX11 0SG
United Kingdom

Email: tim.chown@jisc.ac.uk

John Loughney
Intel
Santa Clara, CA
USA

Email: john.loughney@gmail.com

Timothy Winters
University of New Hampshire, Interoperability Lab (UNH-IOL)
Durham, NH
United States

Email: twinters@iol.unh.edu

6man WG
Internet-Draft
Updates: 4861,6059 (if approved)
Intended status: Standards Track
Expires: May 4, 2017

E. Nordmark
Arista Networks
A. Yourtchenko
Cisco
S. Krishnan
Ericsson
October 31, 2016

IPv6 Neighbor Discovery Optional RS/RA Refresh
draft-ietf-6man-rs-refresh-02

Abstract

IPv6 Neighbor Discovery relies on periodic multicast Router Advertisement messages to update timer values and to distribute new information (such as new prefixes) to hosts. On some links the use of periodic multicast messages to all host becomes expensive, and in some cases it results in hosts waking up frequently. Many implementations of RFC 4861 also use multicast for solicited Router Advertisement messages, even though that behavior is optional.

This specification provides an optional mechanism for hosts and routers where instead of periodic multicast Router Advertisements the hosts are instructed (by the routers) to use Router Solicitations to request refreshed Router Advertisements. This mechanism is enabled by configuring the router to include a new option in the Router Advertisement in order to allow the network administrator to choose host behavior based on whether periodic multicast are more efficient on their link or not. The routers can also tell whether the hosts are capable of the new behavior through a new flag in the Router Solicitations.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Goals and Requirements	4
3. Definition Of Terms	4
4. Protocol Overview	4
5. New Neighbor Discovery Flags and Options	5
5.1. Introducing a Router Solicitation Flag	5
5.2. Refresh Time option	6
6. Conceptual Data Structures	6
7. Host Behavior	7
7.1. Sleep and Wakeup	8
7.2. Movement	8
8. Router Behavior	8
8.1. Router and/or Interface Initialization	9
8.2. Periodic Multicast RA for unmodified hosts	9
8.3. Unsolicited RAs to share new information	9
9. Router Advertisement Consistency	10
10. Security Considerations	10
11. IANA Considerations	10
12. Acknowledgements	10
13. Change Log	10
14. References	11
14.1. Normative References	11
14.2. Informative References	11
Authors' Addresses	12

1. Introduction

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

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

There are still links, such a satellite links, where periodic multicast advertisements is the most efficient and reliable approach to keep the hosts up to date. However other links have different performance and reliability for multicast than for unicast (see for instance [I-D.vyncke-6man-mcast-not-efficient] which discusses WiFi links). On some of those links the performance and reliability is dependent on the direction e.g., with host to network multicast having the same characteristics as unicast, but network to host being different. Cellular networks which employ paging and support sleeping hosts have different issues (see e.g., [I-D.garneij-6man-nd-m2m-issues] that would benefit from having the hosts wake up and request information from the routers instead of the routers periodically multicasting the information.

Since different links types and deployments have different needs, this specification provides mechanism by which the routers can determine whether all the hosts support the RS refresh, and the hosts only employ the RS refresh when instructed by the routers using an option in the Router Advertisement.

The operator retains the option to use unsolicited multicast Router Advertisement to announce new or removed information. That can be useful for uncommon cases while allowing using a higher refresh time for normal network operations.

Hosts that sleep without waking up due to multicast Router Advertisements need to send a RS refresh when they wake up in order to receive configuration changes that took place while the host was sleeping.

The specification does not assume that all hosts on the link implement the new capability. As soon as there are router(s) on a link which supports these optimizations, then the updated hosts on the link can sleep better, while co-existing on the same link with unmodified hosts.

2. Goals and Requirements

The key goal is to allow the operator to choose whether RS refresh is more efficient than periodic multicast RAs, while preserving the timely and scalable reconfiguration capabilities that a periodic RA model provides.

The approach should allow for hosts that sleep on a schedule i.e., that do not wake up due to unsolicited RA messages.

In general a link can have multiple routers hence the RS messages should be multicast to find new routers. But for networks which do not there operator should be able to choose unicast RS behavior.

In addition, an operator might want to be notified whether the link includes hosts that do not support the new mechanism. Potential router implementations can react dynamically to that information, or can log events to system management when hosts appear which do not implement this new capability.

The assumption is that host which implement this specification also implement [I-D.ietf-6man-resilient-rs] as that ensures resiliency to packet loss.

3. Definition Of Terms

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

4. Protocol Overview

The hosts include a new flag in the Router Solicitation message, which allows the routers to report to system management whether there are hosts that do not support the RS refresh on the link.

If the network administrator has configured the routers to send the new Refresh Time option, then the option will be included in all the Router Advertisements. This option includes the time interval when the hosts should send Router Solicitations refresh messages.

The host maintains the value of the Refresh Time option (RTO) by recording it in the default router list. A value of zero can be used to indicate that a router did not include a Refresh Time option.

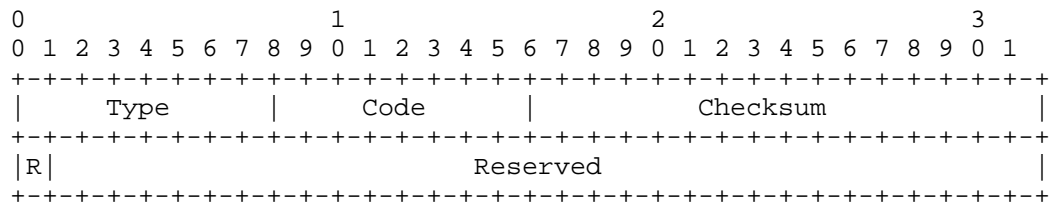
The host calculates a timeout after it has received a RTO - either per router or per link. If it is maintained per link then the host SHOULD use the minimum Refresh Time it has received from the routers on the link. The timeout is a random value uniformly distributed between 0.5 and 1.5 times the Refresh Time value (in order to avoid synchronization of the timers across hosts [SYNC].) When this timer fires the host sends one Router Solicitation.

5. New Neighbor Discovery Flags and Options

This specification introduces a new option used in the RAs which both indicates that the router can handle RS refresh by immediately responding with a unicast RA, and a flag for the RS that indicates to the router that the host will do RS refresh if the router so wishes.

5.1. Introducing a Router Solicitation Flag

A node which implements this specification sets the R flag in all the Router Solicitation messages it sends. That allows the router to determine whether there are legacy hosts on the link.

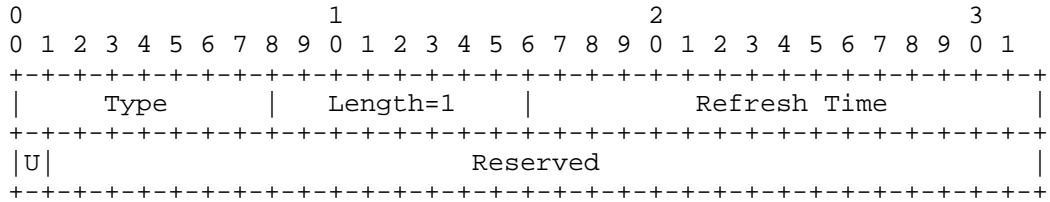


New fields:

- R-flag: When set indicates that the sending node is capable of doing unicast RS refresh.
- Reserved: Field is reduced from 32 bits to 31 bits. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

5.2. Refresh Time option

A router which implements this specification can be configured to instruct hosts to use RS refresh. When the operator configures this mode of operation, then the router MUST include this new option in the RA. If the operator has a single router (or single VRRP router) on the link, then the operator MAY set the Unicast flag in the option.



Fields:

- Type: TBD ND option code value (IANA)
- Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. The value 0 is invalid. Value is 1 for this option.
- Refresh Time: 16-bit unsigned integer. Units is seconds. The value zero is invalid and make the receiver ignore the option.
- U-flag: 1 bit flag to indicate that the host should unicast the RS refresh.
- Reserved: 31 bits. This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

6. Conceptual Data Structures

In addition to the Conceptual Data structures in [RFC4861] a host records the received Refresh Time value and the Unicast flag in the default router list. It also maintains a timeout - either per link or per default router. If the timeout is per link it is set to the minimum of the received Refresh Time values.

7. Host Behavior

See Protocol Overview section above.

A host implementing this specification SHOULD also implement [I-D.ietf-6man-resilient-rs]. That ensures that if there is packet loss and/or the periodic router advertisements are very infrequent, the host will always receive a timely RA as part of its initialization.

If there is no RTO in the received Router Advertisements or there is an RTO with a zero Refresh Time, then the host behavior does not change. However, if RTOs start appearing in RAs after the initial RAs, the host SHOULD start performing RS refresh. As the last router that included RTO options time out from the default router list, the host SHOULD stop sending RS refresh messages.

The host MUST join the all-nodes multicast address as in [RFC4861] since the routers MAY send multicast RAs for important changes.

Some links might have routers with different configuration where some router includes RTO in the RA and others do not. Hosts MAY make the simplifying assumption that if any router on the link includes RTO then the host can use RS refresh to all the routers in the default router list. Also, the routers might advertise different Refresh Time, and hosts MAY use the minimum of the time received from any router that remains in the default router list, or use a separate timer for each router in the default router list. Note that Section 9 says that routers SHOULD report such inconsistencies to system management.

A RTO option with a Refresh Time value of zero is silently ignored, that is, the RA is handled the same way as if it did not contain an RTO option.

If the U-flag is zero for at least one of the routers in the default router list, then the host will send each refresh RS to the all-routers multicast address. Otherwise the host will unicast the RS refresh to each router in the default router list. The host can either maintain the Refresh Time and Unicast flag per router or per link. If they are maintained per router then the host MUST NOT multicast an RS for every default router list entry but instead multicast once when the minimum (across the default router list for the interface) Refresh Time expires. If they are maintained per link, then the host would determine an effective Unicast bit for the link; set if all the routers which sent RTO set the Unicast bit.

If there is no response to a refresh RS, the host follows the same retransmit behavior as in resilient-rs [I-D.ietf-6man-resilient-rs].

7.1. Sleep and Wakeup

The protocol allows the sleepy nodes to complete its sleep schedule without waking up due to multicast Router Advertisement messages and the host is not required to wake up solely for the purposes of performing RS refresh. Such a host SHOULD send a RS refresh upon wakeup even if the Refresh Time has not yet expired, in order to receive any updated RA information.

Hosts that do wake up due to multicast RAs only needs to perform a refresh on wakeup if the Refresh timeout has expired while the host was sleeping.

7.2. Movement

When a host wakes up or thinks it might have moved to a different link (new L2 association, lost and required L2 connectivity, etc) it can combine DNA (Detecting Network Attachment - DNA [RFC6059]), NUD, and refreshing its prefixes etc by sending a unicast RS to each of its existing RTO default router(s). If it receives unicast RA from a router, then it can mark the router as REACHABLE.

Note that DNA specifies using NS messages since many IPv6 routers delay (and multicast) solicited RAs and DNA wants to avoid that delay. Routers which implement this specification and send RTO SHOULD unicast solicited RAs, hence if a router included the RTO then the host can use RS for DNA without incurring additional delay. Thus the host would not need to use a unicast NS as part of DNA for RTO routers. For non-RTO routers the host MAY choose to use NS for DNA as in [RFC6059].

8. Router Behavior

See Protocol Overview section.

A router implementing this specification (and including the RTO in the RAs) SHOULD also respond to unicast RS messages (that do not have an unspecified source address) with unicast RAs. If a RS message has an unspecified source address then the router MAY respond with a RA unicast at layer 2 (sent to the link-layer source address of the RS), or it MAY follow the rate-limited multicast RA procedure in [RFC4861].

The RECOMMENDED default configuration for routers is to have RTO disabled. When RTO is enabled the RECOMMENDED default configuration is to have the Unicast flag disabled.

8.1. Router and/or Interface Initialization

This specification does not change the initialization procedure. Thus a router multicasts some initial Router Advertisements (`MAX_INITIAL_RTR_ADVERTISEMENTS`) at system startup or interface initialization as specified in [RFC4861] and its updates.

8.2. Periodic Multicast RA for unmodified hosts

By default a router MUST send periodic multicast RAs as specified in [RFC4861]. A router can be configured to omit those, which can be used in particular deployments. If they are omitted, then there MUST be a mechanism to prevent or detect the existence of unmodified hosts on the link. That could be performed at deployment time (e.g., only hosts which are known to support RTO are configured with the layer 2 security keys), or the routers could either detect any RSs which do not include the R-flag and report this to system management or dynamically enable periodic multicast RAs when observing at least one RS without the R-flag.

Note that such dynamic detection of "legacy" hosts is not bullet proof, in particular when there is packet loss on the link. If a host does not implement resilient RS [I-D.ietf-6man-resilient-rs], then the host might receive a multicast RA (from router initialization or the periodic multicast RAs) without the router ever receiving a RS from the host. Such a host would function as long as the routers are sending periodic multicast RAs. However, hosts without resilient RS do not operate well in the presence of packet loss. They might be without service (no default router and no prefixes) for one or more multiples of the RA advertisement interval (`MaxRtrAdvInterval`), which currently can be as high as 30 minutes.

8.3. Unsolicited RAs to share new information

When a router has new information to share (new prefixes, prefixes that should be immediately deprecated, etc) it MAY multicast up to `MAX_INITIAL_RTR_ADVERTISEMENTS` number of Router Advertisements.

On links where multicast is expensive the router MAY instead unicast up to `MAX_INITIAL_RTR_ADVERTISEMENTS` number of Router Advertisements to the hosts in its neighbor cache.

Note that such new information is not likely to reach hosts sleeping on a schedule until those hosts refresh by sending a RS. However, as

such hosts are recommended to send a RS refresh when they wake up, they will receive the updated information and not use the potentially stale information to send packets.

9. Router Advertisement Consistency

The routers follows section 6.2.7 in [RFC4861] by receiving RAs from other routers on the link. In addition to the checks in that section, the routers SHOULD verify that the RTO have the same Refresh Time, and report to system management if they differ. While the host will pick the lowest time and operate correctly, it is not useful to use different Refresh Times for different routers.

10. Security Considerations

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

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

The mechanisms in this document work with SeND [RFC3971].

11. IANA Considerations

A new flag (R-flag) in the Router Solicitation message has been introduced by carving out a bit from the Reserved field. There is currently no IANA registry for RS flags. Perhaps one should be created?

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

12. Acknowledgements

The original idea came up in a discussion with Suresh Krishnan. Comments from Samita Chakrabarti, Lorenzo Colitti, and Erik Kline have helped improve the document.

This document has been discussed in the efficient-nd design team.

13. Change Log

Changes since the draft-nordmark-6man-rs-refresh-00 version of the draft:

- o Removed any suggestion that periodic RAs would not be needed. The remain required.

- o Made Refresh Time zero be reserved and RTOs with this value ignored by the receiver.
- o Removed notion that all-ones refresh time means infinite lifetime. It now means 65535 seconds.
- o Changed default to be multicast RS refresh, with the option to specify unicast in the RTO. This enables discovering new routers on the link.
- o Clarified the normative behavior for hosts that sleep on a schedule.
- o Clarified the updated DNA behavior.
- o Editorial fixes.

14. References

14.1. Normative References

- [I-D.ietf-6man-resilient-rs]
Krishnan, S., Anipko, D., and D. Thaler, "Packet loss resiliency for Router Solicitations", draft-ietf-6man-resilient-rs-06 (work in progress), April 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.

14.2. Informative References

- [I-D.garneij-6man-nd-m2m-issues]
Garneij, F., Chakrabarti, S., and S. Krishnan, "Impact of IPv6 Neighbor Discovery on Cellular M2M Networks", draft-garneij-6man-nd-m2m-issues-00 (work in progress), July 2014.

- [I-D.vyncke-6man-mcast-not-efficient]
Vyncke, E., Thubert, P., Levy-Abegnoli, E., and A. Yourtchenko, "Why Network-Layer Multicast is Not Always Efficient At Datalink Layer", draft-vyncke-6man-mcast-not-efficient-01 (work in progress), February 2014.
- [RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, DOI 10.17487/RFC3756, May 2004, <<http://www.rfc-editor.org/info/rfc3756>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<http://www.rfc-editor.org/info/rfc3971>>.
- [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, DOI 10.17487/RFC6059, November 2010, <<http://www.rfc-editor.org/info/rfc6059>>.
- [SYNC] Floyd, S. and V. Jacobson, "The Synchronization of Periodic Routing Messages", IEEE/ACM Transactions on Networking , April 1994, <http://ee.lbl.gov/papers/sync_94.pdf>.

Authors' Addresses

Erik Nordmark
Arista Networks
Santa Clara, CA
USA

Email: nordmark@acm.org

Andrew Yourtchenko
Cisco
7a de Kleetlaan
Diegem, 1831
Belgium

Phone: +32 2 704 5494
Email: ayourtch@cisco.com

Suresh Krishnan
Ericsson
8400 Decarie Blvd.
Town of Mount Royal, QC
Canada

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.com

6man WG
Internet-Draft
Updates: 6724 (if approved)
Intended status: Standards Track
Expires: October 1, 2017

J. Linkova
Google
March 30, 2017

Default Address Selection and Subnet Renumbering
draft-linkova-6man-default-addr-selection-update-00

Abstract

This document discusses some scenarios when IPv6 hosts might not be able to properly detect the fact the network they are connected to has changed IPv6 addressing. It proposes changes to the Default Address Selection algorithm defined in [RFC6724] to mitigate the impact of the abovementioned failure scenarios as well as provides recommendations for sending Prefix Information Options (PIO). It updated [RFC6724].

Status of This Memo

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

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

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

This Internet-Draft will expire on October 1, 2017.

Copyright Notice

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

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

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

Table of Contents

1. Introduction 2
 1.1. Requirements Language 3
 2. Failure Scenarios 3
 3. Proposed Solition 4
 3.1. Default Address Selection Algorithm Update 4
 4. IANA Considerations 6
 5. Security Considerations 6
 6. Acknowledgements 6
 7. Normative References 6
 Appendix A. Change Log 7
 Author's Address 7

1. Introduction

When an IPv6 host configures an address using Stateless Address Autoconfiguration (SLAAC) as described in [RFC4862], the configured address stays preferred (and therefore can be used for new communications) until one of the following happens:

- o its preferred lifetime expires
- o the hosts receives an router advertisement (RA) with the corresponding Prefix Information Option (PIO) with preferred lifetime set to zero
- o the network interface changes its status

In other words once a host get connected to a network and an IPv6 address is configured that address may be used for quite long time (the default value of preferred lifetime is 7 days) or until the host received an explicit notification from a router that the particular SLAAC prefix is not valid anymore.

A host might need to stop using addresses from a particular prefix in the following scenarios:

- o the host has moved to another layer 2 domain (e.g. VLAN or LAN)
- o the layer 2 domain the host is connected to has been renumbered to another /64

In the ideal world the first scenario (a host moving to another layer 2 domain) would trigger the interface status change and as a result all network settings being reset. In the second scenario (network renumbering) it is expected that the router is sending an RA with the "old" PIO preferred lifetime set to zero and then a new POI is sent so hosts can use that POI for SLAAC. In either case the host receives an explicit notification about the addressing change. The preferred lifetime value is acting as a "safety net", with the default value being 604800 seconds (7 days) ([RFC4861]) and the realistic minimal value at least 12 seconds in the best case scenario being too long to rely on to detect the address change.

Unfortunately in practice there are some scenarios when a failure (or misconfiguration) on the host or the network level leads to a situation when a host is using addresses from a prefix which should be deprecated as it is not assigned to that layer 2 domain anymore. This results in a host using a "wrong" IPv6 address for initiating the connection and, as the returning packets can not reach the host, broken IPv6 connectivity and unsatisfactory user experience. Therefore it would be desirable to explore the feasibility of updating hosts and routers behavior to minimize the impact and make IPv6 implementations more robust to such failures/misconfigurations.

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

2. Failure Scenarios

Scenarios when a host might not receive an explicit notification leading to a prefix deprecation include but are not limited to:

- o A switchport the host is connected to is moved to another subnet (VLAN) as a result of manual switchport reconfiguration or 802.1x re-authentication. In particular there have been evidence that some 802.1x supplicants do not reset network setting after successful 802.1x authentication. So if a host had failed 802.1x authentication for some reasons, was placed in a "quarantine" vlan and then got successfully authenticated later on, it might end up having IPv6 addresses from both old ("quarantine") and new vlans.
- o A router which had received a prefix via DHCP-PD and sent RAs with the corresponding PIOs to hosts in LAN segments got rebooted/ crashed. After coming back up the router received a new DHCP-PD prefix so all connected hosts received RAs with a new POI.

- o During the planned network renumbering a router was configured to send an RA with preferred lifetime for the "old" POI set to zero and the new POI having non-zero preferred lifetime. However due to unsolicited RAs being sent as all-hosts multicast and the multicast being rather unreliable on busy wifi network, that RA was not received by a host
- o Automated device config management system performs periodical config push to network devices. If such a push results in changing /64 subnet configured on a particular network, hosts attached to that network would not get notified about the subnet change and their addresses from the "old" prefix are not deprecated. The related case is incorrectly performed renumbering when a network administrator is renumbering a network by simply removing the "old" prefix from the configuration and configuring a new prefix instead.

All those (and others) scenarios result in a situation when the host has addresses from two different prefixes, "old" and "new". As both addresses are preferred and allowed to be used for communication the host relies on the default address selection algorithm ([RFC6724]) to choose a source address. If the address from the "old" prefix is selected as source address, then even if the packet reaches its destination (not being dropped due to antispoofing or any other type of filtering), the return traffic would not be delivered to the host.

3. Proposed Solution

3.1. Default Address Selection Algorithm Update

The Default Address Selection algorithm defined in [RFC6724] describes 8 rules to choose a single source address for use with a given IPv6 destination address. In the abovementioned scenario when the host has preferred addresses from two GUA prefixes, the first 7 rules can not act as a tie breaker. In theory when the host moves from one network segment from another its default router link-local address would change and the rule 5.5, "Prefer addresses in a prefix advertised by the next-hop" can lead to selecting a source address from the "new" prefix. However there are two reasons why the rule 5.5 can not reliably ensure that the "new" prefix is preferred over the "old" one:

1. The link-local address of the router in the new layer 2 domain might be the same as the link-local address of the "old" router (it's quite common to have link-local address on routers to be explicitly configured, especially in VRRP-enabled environments)

2. Until recently ([RFC8028]) IPv6 implementations were not required to track what next hop advertized what PIO and therefore the rule 5.5 was not applicable for such implementations.

The last rule, the rule 8, instructs the host to use the longest matching prefix and according to [RFC6724] that rule MAY be superseded if the implementation has other means of choosing among source addresses. In all scenarios described above it seems to be beneficial to prefer an address from the most recently received PIO. It would ensure that if the network subnet has been changed and the host has addresses from both "old" and "new" prefixes, it would prefer the new prefix. In generic case choosing an address from the most recent PIO if none of the first seven source address selection rules can be a tie breaker is harmless. If all POIs were received in the same time (the same RA) then the rule 8 (or any other means) can be used to choose the source address.

Therefore this document proposes the following changes to the Section 5 of [RFC6724]:

OLD TEXT:

Rule 8: Use longest matching prefix.

If $\text{CommonPrefixLen}(\text{SA}, \text{D}) > \text{CommonPrefixLen}(\text{SB}, \text{D})$, then prefer SA. Similarly, if $\text{CommonPrefixLen}(\text{SB}, \text{D}) > \text{CommonPrefixLen}(\text{SA}, \text{D})$, then prefer SB.

Rule 8 MAY be superseded if the implementation has other means of choosing among source addresses. For example, if the implementation somehow knows which source address will result in the "best" communications performance.

NEW TEXT:

Rule 8: Use the address from the most recently refreshed prefix.

If SA's PIO was received more recently than SB's POI, then prefer SA. Similarly, if SB's POI was received more recently than SA's POI, then prefer SB. If the implementation does not keep track of when the particular POI was received, then the addresses preferred lifetime SHOULD be considered instead: if $\text{preferred lifetime}(\text{SA}) > \text{preferred lifetime}(\text{SB})$, then prefer SA. Similarly, if $\text{preferred lifetime}(\text{SB}) > \text{preferred lifetime}(\text{SA})$, then prefer SB.

Rule 9: Use longest matching prefix.

If `CommonPrefixLen(SA, D) > CommonPrefixLen(SB, D)`, then prefer SA. Similarly, if `CommonPrefixLen(SB, D) > CommonPrefixLen(SA, D)`, then prefer SB.

Rules 8 and 9 MAY be superseded if the implementation has other means of choosing among source addresses. For example, if the implementation somehow knows which source address will result in the "best" communications performance.

To make the proposed solution work for the implementations which do not record when an RA with the PIO was most recently received, both old and new POI need to be advertised with same (or reasonably similar) preferred lifetime value. Otherwise it is possible that even the new POI was received after the old POI, the preferred lifetime of the old prefix might be still higher than one of the new prefix (if the preferred lifetime field value for the old prefix was much higher than the corresponding value for the new prefix). Despite such a limitation it seems reasonable to assume that in most scenarios described in Section 2 the PIOs preferred lifetime values would not vary much.

4. IANA Considerations

This memo asks the IANA for no new parameters.

5. Security Considerations

This memo has no direct security considerations.

6. Acknowledgements

The authors thank Erik Kline for input and contributions.

7. Normative References

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<http://www.rfc-editor.org/info/rfc8028>>.

Appendix A. Change Log

Initial Version: March 2017

Author's Address

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

Internet Engineering Task Force
Internet-Draft
Updates: 4861,6275 (if approved)
Intended status: Experimental
Expires: September 28, 2017

E. Kline
Google Japan KK
M. Abrahamsson
T-Systems Nordic
March 27, 2017

IPv6 Router Advertisement Prefix Information Option eXclusive Flag
draft-pioxfolks-6man-pio-exclusive-bit-02

Abstract

This document defines a new control bit in the IPv6 RA PIO flags octet that indicates that the node receiving this RA is the exclusive receiver of all traffic destined to any address within that prefix.

Termed the eXclusive flag (or "X flag"), nodes that recognize this can perform some optimizations to save time and traffic (e.g. disable ND and DAD for addresses within this prefix) and more immediately pursue the benefits of being provided multiple addresses (vis. [RFC7934] section 3). Additionally, network infrastructure nodes (routers, switches) can benefit by minimizing the number of {link layer, IP} address pairs required to offer network connectivity (vis. [RFC7934] section 9.3).

Use of the X flag is backward compatible with existing IPv6 standards compliant implementations.

Status of This Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

1.	Introduction	3
2.	Motivation	3
2.1.	Efficiency improvements	4
2.2.	New architectural possibilities	4
3.	Applicability statement	5
4.	Terminology	6
4.1.	Requirements Language	6
4.2.	Abbreviations	6
4.2.1.	PIO-X	6
4.2.2.	PIO-X RA	6
4.2.3.	Host	6
5.	Updated Prefix Information Option	6
5.1.	Updated format description	6
5.2.	Receiver processing	7
5.2.1.	PIO R flag	7
5.2.2.	(Re)Interpretation of other flags	7
5.2.2.1.	PIO L flag	8
5.2.2.2.	PIO A flag	8
5.3.	Sender requirements	8
5.4.	Comparison with DHCPv6 PD	9
6.	Host behavior	9
6.1.	PIO-X processing	9
6.2.	Neighbor Discovery implications	9
6.2.1.	Duplicate Address Detection (DAD)	9
6.2.2.	Router Solicitations (RSes)	10
6.3.	Link-local address behavior	10
6.4.	Source address selection	10
6.5.	Next hop router selection	11
6.6.	Implications for Detecting Network Attachment	11
6.7.	Additional guidance	11
7.	Router behavior	11
7.1.	PIO-X RA destination address	11
7.2.	Detecting hosts to send PIO-X RAs to	12

7.3. Binding table requirements	12
7.4. Preparations before sending a PIO-X RA	13
7.5. Implementation considerations	13
8. Acknowledgements	13
9. IANA Considerations	13
10. Security Considerations	13
11. References	14
11.1. Normative References	14
11.2. Informative References	14
Authors' Addresses	15

1. Introduction

This document defines a new control flag in the Internet Protocol version 6 (IPv6) Router Advertisement (RA) Prefix Information Option (PIO) flags octet that indicates that the node receiving this RA is the exclusive receiver of all traffic destined to any address with that prefix. Subject to the lifetime constraints within the PIO, the receiving node effectively has exclusive use of the prefix, and will be the next hop destination for the sending router, and possibly other routers, for all traffic destined toward the prefix.

Termed the eXclusive flag (or "X flag"), nodes that recognize this can perform some optimizations to save time and traffic (e.g. disable Neighbor Discovery (ND) and Duplicate Address Detection (DAD) for addresses within this prefix) and more immediately pursue the benefits of being provided multiple addresses (vis. [RFC7934] section 3).

Additionally, network infrastructure nodes (routers, switches) can benefit by minimizing the number of {link layer, IP} address pairs required to offer network connectivity (vis. [RFC7934] section 9.3). A router, for example, need not create any {link layer, IP} address pair entries for IP address within a proffered exclusive-use prefix-- it can reliably forward all traffic to the network node to which it advertised the prefix. This solves one potential link layer state exhaustion problem, i.e excessive number of {link layer, IP address pairs}, using IP layer forwarding.

Use of the X flag is backward compatible with existing IPv6 standards compliant implementations. [RFC4861]-compliant nodes that do not understand the X flag are not negatively impacted. They must ignore it, and can process the PIO under existing standards, making use of the information exactly as if the X flag were not set.

2. Motivation

This work is motivated by the pursuit of two categories of benefits: modest host and network side improvements in efficiency, and support for new deployment architectures and address space use models.

2.1. Efficiency improvements

If a host knows it has exclusive use of a prefix it can perform some optimizations to save time and traffic. It can avoid ND on the receiving interface for addresses within these prefixes. Network interfaces can even drop Neighbor Solicitations for these addresses on the receiving interface to save power by not waking up more power-hungry CPUs.

Additionally, a host can save time by not performing DAD for addresses within an exclusive-use prefix on the receiving interface. A host that wanted, for example, to use 2^{64} unique IPv6 source addresses for DNS queries in order to improve resilience against forged answers (as recommended in section 9.2 of [1]), could do so without delaying each query from a newly formed address. A node could in theory implement the same strategy using Optimistic Duplicate Address Detection [2], but it could be very unfriendly to the network infrastructure (in terms of {link-layer, IP address} pair state) to do so without this kind of explicit signal.

A host that recognizes the X flag might perform other traffic-saving optimizations, like not attempt Multicast DNS in some cases, or avoid trying to register addresses with sleep proxies. Being the only host on this link these may be of little benefit.

2.2. New architectural possibilities

There are several initiatives that propose network side practices that provide customer isolation, enhanced operational scalability, power efficiency, security and other benefits in IPv6 network deployments. Some of these involve isolating a host (or RA accepting client node) so that the host is the only node to receive a specific prefix, including

- o DHCPv6 Prefix Delegation to hosts ([3]), and
- o advertising a unique prefix per host via unique RAs. ([4]).

Some architectures further isolate the host layers below IPv6, for improved client node security.

Regardless of the specific level of isolation, the host can best make choices about its use of a prefix exclusively forwarded to itself if the host can be informed of the exclusivity. (In the case of a

DHCPv6 Prefix Delegation the prefix can be assumed to be of exclusive use by the requesting node, in accordance with the model in [RFC3633].) An implementation can, for example, safely "bind to an IPv6 subnet" in the style of [5], or start 64sharing [6] (given a prefix of sufficient size).

This memo documents an additional flag in the IPv6 RA PIO that makes this information explicit to receiving node.

3. Applicability statement

Use of the X flag in PIOs is only applicable to networks where the architecture (i.e. serving infrastructure like routers, link-layer equipment, et cetera) can collectively guarantee the following criteria are met:

1. an RA containing a PIO with the X flag set MUST be delivered to one and only target node (host) such that no two nodes can reasonably expect exclusive access to the same prefix at the same time
2. any router advertising an RA containing a PIO with the X flag set SHOULD be notified quickly when a node leaves the network

The first criterion ensures that the same exclusive use prefix is not advertised to more than one host at a time (and hence no longer "exclusive"). This implies that an allocated exclusive-use prefix must be tracked by the issuing router for at least the minimum of (a) the lifetime of the recipient node's continuous attachment to the network and (b) the lifetime of the prefix itself in the PIO, if not longer.

The second criterion aims to help the prefix allocation infrastructure reclaim unused prefixes quickly while also helping routers drop (possibly with appropriate ICMPv6 errors) traffic that can no longer be delivered.

It is expected that in practice this primarily describes networks where the IPv6 infrastructure and the link-layer have a tight integration. All point-to-point links meet these criteria (e.g. PPPoE and VPNs), as does the 3GPP architecture [RFC7066] and some IEEE 802.11 deployment architectures ([7]).

4. Terminology

4.1. Requirements Language

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

4.2. Abbreviations

Throughout this document the following terminology is used purely for the sake of brevity.

4.2.1. PIO-X

The term "PIO-X" is used to refer to a Prefix Information Option (PIO) that has the X flag set.

4.2.2. PIO-X RA

The phrase "PIO-X RA" is used to refer to an IPv6 Router Advertisement (RA) that contains one or more PIO-X entries (the same RA may also contain one or more PIOs without the X flag set).

4.2.3. Host

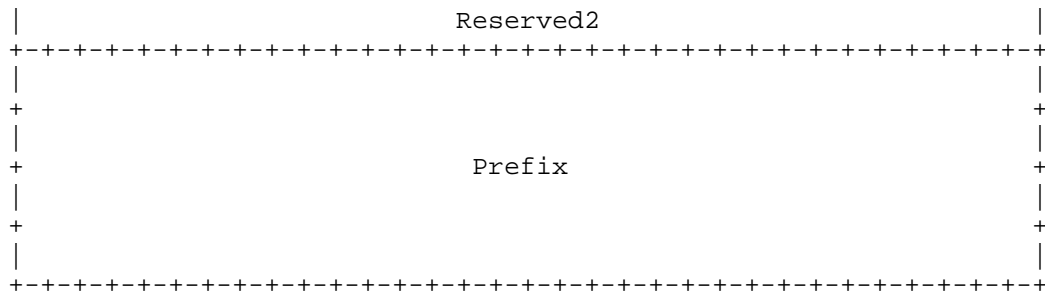
The term "host" may be used interchangeably throughout this document to mean a network node receiving and processing an RA. The receiving node may itself be a router, or may temporarily become one by routing all or a portion of an exclusive use prefix.

5. Updated Prefix Information Option

This document updates the Prefix Information Option specification in RFC 4861 [8] section 4.6.2 and RFC 6275 [9] section 7.2 with the definition of a flag from the former Reserved1 field as follows.

5.1. Updated format description

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Type										Length										Prefix Length										L A R X Rsvrd1									
Valid Lifetime																																							
Preferred Lifetime																																							



Fields:

X The eXclusive use indicator flag, defined by this document. When set, the receiving node can be assured that all traffic destined to any address within the specified Prefix will be forwarded to itself by, at a minimum, the router from which the encapsulating RA was received, but possibly other routers as well.

When not set, the receiving node MUST NOT make any assumptions of exclusive use of the specified Prefix, i.e. processing is unchanged from previous standards behavior.

Rsrvd1 Retains the same meaning as Reserved1 from [10] section 4.6.2.

All other fields Retain their same meaning from [11] section 4.6.2.

5.2. Receiver processing

Nodes compliant with this specification perform the following additional processing of RAs and PIO-X options when a PIO-X option is present.

5.2.1. PIO R flag

If the R flag is set then the X flag MUST be ignored. The R flag indicates that the PIO includes an address the router has selected for itself from the prefix. Logically, the prefix cannot exclusively be used by the receiving node if the router has allocated any addresses for itself from the prefix.

5.2.2. (Re)Interpretation of other flags

Nodes compliant with this specification, i.e. those that understand the X-flag, MUST, when the X-flag is set, ignore the actual values of the L and A flags and instead interpret them as follows:

- o interpret the L flag as if it were 0 (L=0)
- o interpret the A flag as if it were 1 (A=1)

The rationale for this is as follows.

5.2.2.1. PIO L flag

Because a PIO-X aware node will know that it has exclusive use of a prefix with non-zero valid lifetime, the prefix itself cannot be considered to be on-link with respect to the link on which the PIO-X RA was received.

Note that a given address from within the prefix may be considered on-link according to the definition in [12] section 4, item 1, should the receiving node chose to configure that address on said link, but this is in no way synonymous with the entire prefix being considered on-link.

5.2.2.2. PIO A flag

Because a PIO-X aware node will know that it has exclusive use of a prefix with non-zero valid lifetime, autoconfiguration of addresses according to any desired scheme, e.g. [13], [14], et cetera, is implicit in the setting of the X flag.

Accordingly, the A flag can be interpreted as having been set, should the host choose to apply standard address generation schemes that require the flag to be set. It is free to assign any address formed from an exclusive prefix to any available interface; it is not required to configure the address on the link over which the PIO-X RA was received (i.e. it is under no obligation to form addresses such that they would be classified as on-link (according to the definition in [15] section 4, item 1).

5.3. Sender requirements

When a router transmits an RA containing one or more PIO-X options it SHOULD unicast the PIO-X RA to its intended recipient at the IPv6 layer and, if applicable, at the link-layer.

It is RECOMMENDED that a PIO with the X-flag set also have the PIO flags L=0 and A=1 explicitly configured, for backward compatibility (i.e. use by non X-flag aware nodes).

A router transmitting a PIO-X RA MUST NOT configure for itself any address from with the PIO-X prefix. (If it did, the prefix would logically no longer be of exclusive use for the receiving node.)

5.4. Comparison with DHCPv6 PD

There exists a key difference in semantics between PIO-X and DHCPv6 PD: with PIO-X the network keeps the client refreshed with its prefix whereas with DHCPv6 PD the client is responsible for refreshing its prefix from the server. This is one reason it is important for the data link layer to be able to quickly inform routers of client detachment.

Another difference is that [16] section 12.1 states:

... the requesting router MUST NOT assign any delegated prefixes or subnets from the delegated prefix(es) to the link through which it received the DHCP message from the delegating router.

In contrast, a node receiving a PIO-X RA is explicitly free to treat the entire prefix as on-link with respect to the interface via which it was received.

6. Host behavior

TODO: This section needs some work.

6.1. PIO-X processing

A receiving node compliant with this document processes an RA with a PIO entry with the X flag set according the requirements in previous standards documents (chiefly [17] section 6.3.4) subject to the additional requirements documented in Section 5.2.

6.2. Neighbor Discovery implications

6.2.1. Duplicate Address Detection (DAD)

Whatever use the host makes of the exclusive prefix during its valid lifetime, it SHOULD NOT perform Duplicate Address Detection ("DAD", [18] section 5.4) on any address it configures from within the prefix if that address is configured on either the interface over which the PIO-X RA was received or on a loopback interface. Note that this does not absolve the host from performing DAD in all scenarios; if, for example, the host uses the prefix for 64sharing [19] it MUST at a minimum defend via DAD any addresses it has configured for itself as documented in Requirement 2 of [20] section 3.

6.2.2. Router Solicitations (RSes)

Routers announcing PIO-X RAs do so via IPv6 unicast to the intended receiving node and may note the IPv6 unicast destination address of an RS as the next hop for the exclusive prefix. As such, hosts compliant with this SHOULD NOT use the unspecified address (::) when sending RSes; they SHOULD prefer issuing Router Solicitations from a link-local address.

It is possible for a node to receive multiple RAs with a mix of exclusive and non-exclusive PIOs and even non-zero and zero default router lifetimes. While it is not possible for a host (receiving node) to be sure it has received all the RA information available to it, hosts compliant with this specification SHOULD implement Packet-Loss Resiliency for Router Solicitations [RFC7559] so that the host continues to transmit Router Solicitations at least until an RA with a non-zero default router lifetime has been seen.

6.3. Link-local address behavior

Routers announcing PIO-X RAs may record the source (link-local) address of an RS as the next hop for the exclusive prefix. A node compliant with this specification MUST continue to respond to Neighbor Solicitations for the source address used to send RSes (alternatively: the destination address of unicast PIO-X RAs received). Hosts that deprecate or even remove this address may experience a loss of connectivity.

6.4. Source address selection

No change to existing source address selection behavior is required or specified by this document.

6.5. Next hop router selection

No change to existing next hop router selection behavior is required or specified by this document.

6.6. Implications for Detecting Network Attachment

TODO: Describe implications for Detecting Network Attachment in IPv6 [21] (DNav6). Probably the best that can be done is (a) no change to RFC6059 coupled with (b) a host MAY send a test packet (e.g. ICMPv6 Echo Request) with a source and destination address from within the PIO-X prefix to the PIO-X RA issuing router and verify the packet is delivered back to itself. Consistent failure to receive such traffic MAY be considered a signal that the exclusive prefix should no longer be used by the host.

6.7. Additional guidance

The intent of networks that use PIO-X RAs is not to enable sophisticated routing architectures that could be far better handled by an actual routing protocol but rather to propagate a prefix's exclusive use information to enable the receiving node to make better use of the available addresses. As such:

A PIO-X receiving node SHOULD NOT issue ICMPv6 Redirects ([RFC4861] section 4.5) for any address within an exclusive use prefix via the link over which the PIO-X RA was received. Redirecting portions of exclusive prefixes to other "upstream" on-link nodes is not a supported configuration.

A PIO-X receiving node SHOULD NOT transmit RAs with any subset of its exclusive prefixes via the same interface through which the exclusive prefix was learned.

7. Router behavior

TODO: This section needs some work.

7.1. PIO-X RA destination address

Since the host will not perform DAD for addresses within prefix announced via PIO-X, it's very important that only a single host receives the PIO-X RA. Therefore, the router MUST only include PIO-X in RAs that are sent using unicast RAs to destination unicast link-layer address and IPv6 link-local unicast address for a specific host. For point-to-point media without link-layer addresses or where there is guaranteed to only be single host that will receive the PIO-X RA (e.g. as enforced by link layer mechanisms), the router MAY

send PIO-X RA with multicast destination IPv6 address. Under all circumstances the router MUST maintain a binding table of state information as discussed in Section 7.3.

7.2. Detecting hosts to send PIO-X RAs to

When the host starts using a network connection it normally sends out an RS (Router Solicitation) packet. This is one way for the router to detect that a new host is connected to the network and detects its link-local address. If the router is configured to use PIO-X, it can now perform necessary processing/configuration and then send the PIO-X RA.

For some networks, the host information regarding link-layer and link-local address might be available through other mechanism(s). Examples of this are PPP, 802.1x and 3GPP mobile networks. In that case this information MAY be used instead of relying on the host to send RS. It is however RECOMMENDED that these networks also provide indication whether the host is no longer connected to the network so that the router can invalidate the prefix binding prior to binding expiration (timeout).

7.3. Binding table requirements

Routers transmitting PIO-X RAs have state maintenance and operational requirements similar to delegating routers in networks where DHCPv6 Prefix Delegation [RFC3633] is used. The state maintained is describe here in terms of a conceptual binding table.

- R1 The router SHOULD keep track of which PIO-X prefix has been issued to each node.
- R2 The router SHOULD keep the binding between prefix and link-local address for the advertised valid lifetime, plus some operationally determined delay prior to reissuing a prefix ("grace period"), of the prefix.
- R3 The router MUST monitor the reachability of each node in the binding table via Neighbor Unreachability Detection ("NUD", [22] section 7.3) or an equivalent link-layer mechanism.
- R4 The binding SHOULD be considered refreshed every time a periodic PIO-X RA is sent to a node.

R5 If the router is informed by some other mechanism (link-layer indication for instance) that a node is no longer connected to the link, it MAY immediately invalidate the prefix binding. (DISCUSS: Is this the correct approach? Do we want to point to some definition somewhere else?)

7.4. Preparations before sending a PIO-X RA

When the router intends to send a PIO-X RA, it SHOULD before sending the PIO-X RA, complete any and all necessary processing for the host to start using the PIO-X prefix to communicate through the router to other networks. This is so that the host can start using PIO-X based addresses without delay or error after receipt of the PIO-X RA.

7.5. Implementation considerations

TODO: Out of scope things that are worth careful consideration include...

Routers SHOULD NOT announce the same prefix to two different nodes within the valid lifetime of the earlier of the two PIO-X announcements.

A link may operate in a mode where routers announce RAs to all nodes, possibly with non-exclusive PIO data, and non-zero default router lifetimes. Separately, one or more other nodes on the link may announce exclusive PIO information to nodes along with zero default router lifetimes. Except in the presence of a non-expired more specific route, e.g. learning from an [23] Route Information Option (RIO), the receiving node should send exclusive use prefix originated or forwarded traffic destined off-link through routers with non-zero default router lifetimes.

8. Acknowledgements

9. IANA Considerations

This memo contains no requests of IANA.

10. Security Considerations

This document fundamentally introduces no new protocol or behavior substantively different from existing behavior on a link which guarantees a unique /64 prefix to every attached host. It only describes a mechanism to convey that topological reality, allowing the host to make certain optimizations as well as share the exclusive prefix as it sees fit with other nodes according to its capabilities and policies.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<http://www.rfc-editor.org/info/rfc6275>>.
- [RFC7559] Krishnan, S., Anipko, D., and D. Thaler, "Packet-Loss Resiliency for Router Solicitations", RFC 7559, DOI 10.17487/RFC7559, May 2015, <<http://www.rfc-editor.org/info/rfc7559>>.

11.2. Informative References

- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<http://www.rfc-editor.org/info/rfc4429>>.
- [RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <<http://www.rfc-editor.org/info/rfc5452>>.
- [RFC7066] Korhonen, J., Ed., Arkko, J., Ed., Savolainen, T., and S. Krishnan, "IPv6 for Third Generation Partnership Project

(3GPP) Cellular Hosts", RFC 7066, DOI 10.17487/RFC7066,
November 2013, <<http://www.rfc-editor.org/info/rfc7066>>.

[RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi,
"Host Address Availability Recommendations", BCP 204, RFC
7934, DOI 10.17487/RFC7934, July 2016,
<<http://www.rfc-editor.org/info/rfc7934>>.

Authors' Addresses

Erik Kline
Google Japan KK
6-10-1 Roppongi
Mori Tower, 44th floor
Minato, Tokyo 106-6126
JP

Email: ek@google.com

Mikael Abrahamsson
T-Systems Nordic
Kistagangen 26
Stockholm
SE

Email: Mikael.Abrahamsson@t-systems.se

Network Working Group
Internet-Draft
Updates: rfc4191, rfc4861 (if approved)
Intended status: Standards Track
Expires: June 7, 2018

F. Templin, Ed.
Boeing Research & Technology
December 04, 2017

The AERO Address
draft-templin-6man-aeroaddr-01.txt

Abstract

IPv6 interfaces are required to have a link-local address that is unique on the link. Nodes normally derive a link local address through the use of IPv6 Stateless Address Autoconfiguration (SLAAC) along with Duplicate Address Detection (DAD). This document presents a method for a node to construct a link-local address that is assured to be unique on the link when the node has already received a delegated prefix. This is through the construction of a link-local address format known as the AERO address.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 June 7, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<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
2. Terminology	2
3. The AERO Address	2
4. Intended Use Cases	3
5. Implementation Status	3
6. IANA Considerations	4
7. Security Considerations	4
8. Acknowledgements	4
9. References	4
9.1. Normative References	4
9.2. Informative References	4
Author's Address	5

1. Introduction

IPv6 interfaces are required to have a link-local address that is unique on the link [RFC2460][RFC4861]. Nodes normally derive a link local address through the use of IPv6 Stateless Address Auto Configuration (SLAAC) along with Duplicate Address Detection (DAD) [RFC4862]. This document presents a method for a node to construct a link-local address that is assured to be unique on the link when the node has already received a delegated prefix. This is through the construction of a link-local address format known as the AERO address.

2. Terminology

The terminology in the normative references applies.

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]. Lower case uses of these words are not to be interpreted as carrying RFC2119 significance.

3. The AERO Address

An AERO address is an IPv6 link-local address with an interface identifier based on a prefix that has been delegated to a node for its own exclusive use. AERO addresses begin with the prefix fe80::/64 and include in the interface identifier (i.e., the lower 64

bits) a 64-bit prefix taken from one of the node's delegated prefixes. For example, if the node receives the IPv6 prefix:

```
2001:db8:1000:2000::/64
```

it constructs its corresponding AERO addresses as:

```
fe80::2001:db8:1000:2000
```

After constructing the AERO address, the node can assign the address to the interface over which it received the prefix delegation. Since the prefix delegation is already known to be unique, the node need not use Duplicate Address Detection (DAD) to test the AERO address for uniqueness since no other node on the link will configure the same address.

AERO addresses can be constructed for any IPv6 prefix that is no longer than /64. For prefixes shorter than /64, the AERO address is constructed based on the lowest-numbered /64 prefix taken from the shorter prefix. For example, if the node received the IPv6 prefix:

```
2001:db8:1000:2000::/56
```

it constructs its corresponding AERO addresses as:

```
fe80::2001:db8:1000:2000
```

4. Intended Use Cases

The AERO address is intended for use by mobile networks that comprise a mobile router and a tethered network of "Internet of Things" devices that travel together with the router as a single unit. The mobile router assigns the AERO address to its upstream interface over which it receives a prefix delegation from a delegating router. The manner for receiving the delegated prefix could be through static configuration or some automated prefix delegation service.

Many other use case scenarios are possible (e.g., home networks) but the above case extends to multitudes of applications, e.g., a cell phone and its associated devices, an airplane and its on-board network, etc.

5. Implementation Status

Public domain implementations exist that use the AERO address format as described in this document.

6. IANA Considerations

This document introduces no IANA considerations.

7. Security Considerations

TBD

8. Acknowledgements

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program and the Boeing Research & Technology (BR&T) enterprise autonomy program.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.

9.2. Informative References

[I-D.templin-aerolink]

Templin, F., "Asymmetric Extended Route Optimization
(AERO)", draft-templin-aerolink-76 (work in progress),
November 2017.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Network Working Group
Internet-Draft
Updates: rfc4191 (if approved)
Intended status: Standards Track
Expires: May 3, 2018

F. Templin, Ed.
Boeing Research & Technology
J. Woodyatt
Google
October 30, 2017

Route Information Options in IPv6 Neighbor Discovery
draft-templin-6man-rio-redirect-05.txt

Abstract

The IPv6 Neighbor Discovery (ND) protocol allows nodes to discover neighbors on the same link. Router Advertisement (RA) messages can also convey routing information by including a non-zero (default) Router Lifetime, and/or Route Information Options (RIOs). This document specifies backward-compatible extensions that permit nodes to include RIOs in other IPv6 ND messages to support the discovery of more-specific routes.

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 May 3, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<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
2. Terminology	3
3. Motivation	3
4. Route Information Options (RIOs) in IPv6 Neighbor Discovery Messages	5
4.1. RIO Update	5
4.2. RIO Requirements	7
4.3. Classic Redirection Scenario	7
4.4. RIO Redirection Scenario	9
4.4.1. Router Specification	9
4.4.2. Source Specification	10
4.4.3. Target Specification	11
4.5. Operation Without Redirects	11
4.6. Multiple RIOs	12
4.7. Multicast	12
4.8. Why NS/NA?	12
5. Implementation Status	13
6. IANA Considerations	13
7. Security Considerations	14
8. Acknowledgements	15
9. References	15
9.1. Normative References	15
9.2. Informative References	16
Appendix A. Link-layer Address Changes	17
Appendix B. Interfaces with Multiple Link-Layer Addresses	17
Appendix C. Change Log	17
Authors' Addresses	18

1. Introduction

"Neighbor Discovery for IP version 6 (IPv6)" [RFC4861] (IPv6 ND) provides a Router Solicitation (RS) function allowing nodes to solicit a Router Advertisement (RA) response from an on-link router, a Neighbor Solicitation (NS) function allowing nodes to solicit a Neighbor Advertisement (NA) response from an on-link neighbor, and a Redirect function allowing routers to inform nodes of a better next hop neighbor on the link toward the destination. Further guidance for processing Redirect messages is given in "First-Hop Router Selection by Hosts in a Multi-Prefix Network" [RFC8028].

"Default Router Preferences and More-Specific Routes" [RFC4191] specifies a Route Information Option (RIO) that routers can include

in RA messages to inform recipients of more-specific routes (section 1 of that document provides rationale for the use of RA messages instead of an adjunct routing protocol). This document specifies a backward-compatible and incrementally-deployable extension to allow nodes to include RIOs in other IPv6 ND messages to support the dynamic discovery of more-specific routes. This allows nodes to discover a better neighbor for more-specific routes to both increase performance and reduce the workload on default routers.

This approach applies to any link type on which there may be many nodes that provision delegated prefixes on their downstream interfaces and do not provide transit services between upstream networks. These nodes can either be routers that forward packets on behalf of their downstream networks, or hosts that use a delegated prefix for their own multi-addressing purposes
[I-D.templin-v6ops-pdhost][RFC7934].

This work benefits from the experience of [RFC6706] - an experimental protocol that uses UDP-based "pseudo-ND" messages instead of actual ICMPv6 message codes. That experience has shown that using synthesized UDP messages in addition to the IPv6 ND messaging already present on the link is inefficient. Furthermore, the UDP approach is neither backward-compatible nor incrementally-deployable, since sending UDP messages blindly to a node that does not have the port open could be mis-interpreted as a port scan attack. This specification avoids these issues by using the already-present and natural IPv6 ND messaging available on the link, as specified in this document.

2. Terminology

The terminology in the normative references applies.

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]. Lower case uses of these words are not to be interpreted as carrying RFC2119 significance.

3. Motivation

An example of a good application for RIO is the local-area subnets served by the routers described in "Basic Requirements for IPv6 Customer Edge Routers" [RFC7084]. While many customer edge routers are capable of operating in a mode with a dynamic routing protocol operating in the local-area network, the default mode of operation is typically designed for unmanaged operation without any dynamic routing protocol. On these networks, the only means for any node to

learn about routers on the link is by using the Router Discovery protocol described in [RFC4861].

Nevertheless, hosts on unmanaged home subnets may use "IPv6 Prefix Options for DHCPv6" [RFC3633] (DHCPv6 PD) to receive IPv6 routing prefixes for additional subnets allocated from the space provided by the service provider, and operate as routers for other links where hosts in delegated subnets are attached. Hosts may even learn about more specific routes than the default route by processing RIOs in RA messages as described in [RFC4191].

However, due to perceptions of the security considerations for hosts in processing RIOs on unmanaged networks, the default configuration for common host IPv6 implementations is to ignore RIOs. Accordingly, on typical home networks the forwarding path from hosts on one subnet to destinations on every off-link local subnet always passes through the customer edge router, even when a shorter path would otherwise be available through an on-link router. This adds costs for retransmission on shared LAN media, often adding latency and jitter with queuing delay and delay variability. This is not materially different under the scenarios described in "IPv6 Home Networking Architecture Principles" [RFC7368] except that routers may use an interior dynamic routing protocol to coordinate sending of RIOs in RA messages, which as explained above, are not processed by typical hosts.

In increasingly common practice, a node that receives a prefix delegation can use the prefix for its own multi-addressing purposes or can connect an entourage of "Internet of Things (IoT)" back end devices (an approach sometimes known as "tethering" [RFC7934]). On many link types, the number of such nodes may be quite large which would make running a dynamic routing protocol between the nodes impractical. Example use cases include:

- o IETF conference, airport, and hotel WiFi networks, where large numbers of nodes on the link could receive IPv6 prefix delegations. Using the extensions described in this document, the nodes could dynamically discover more-specific routes to enable direct neighbor-to-neighbor communications.
- o Mobile enterprise devices that connect into a corporate network via VPN links. Using the extensions described in this document, mobile devices could dynamically establish pair-wise VPN links between themselves without having to use the enterprise network as transit.
- o Civil aviation networks where an aircraft holds an IPv6 prefix derived from the identification value assigned to it by the

International Civil Aviation Organization (ICAO). Using the extensions described in this document, direct paths between the aircraft and Air Traffic Control (ATC) can be established to provide a more direct route for communications.

- o Unmanned Air System (UAS) networks where each UAS receives an IPv6 prefix delegation for operation with in the Unmanned Air Traffic Management (UTM) service under development within NASA and the FAA. Using the extensions described in this document, very large numbers of UAS can be accommodated by the UTM service for both vehicle-to-infrastructure and vehicle-to-vehicle communications.

By using RIOs in IPv6 ND messages, the forwarding path between subnets can be shortened while accepting a much narrower opening of attack surfaces on general purpose hosts related to the Router Discovery protocol. The basic idea is simple: hosts normally send packets for off-link destinations to their default router unless they receive ND Redirect messages designating another on-link node as the target. This document allows ND Redirects additionally to suggest another on-link node as the target for one or more routing prefixes, including one with the destination. Hosts that receive RIOs in ND Redirect messages then send NS messages to the target containing those RIOs, and process the NA messages the target sends in reply. If hosts only process RIOs in NA messages when they have previously sent them in NS messages to the targets of received ND Redirect messages, then hosts only process RIO at the initiative of routers they already accept as authoritative.

4. Route Information Options (RIOs) in IPv6 Neighbor Discovery Messages

The RIO is specified for inclusion in RA messages in Section 2.3 of [RFC4191], while the neighbor discovery functions are specified in [RFC4861]. This specification permits routers to include RIOs in other IPv6 ND messages so that recipients can discover a better next hop for a destination *prefix* instead of just a specific destination address. This specification therefore updates [RFC4191] as discussed in the following sections.

4.1. RIO Update

The RIO format given in Section 2.3 of [RFC4191] is updated by this specification as shown in Figure 1:

This document defines the NULL Attribute with Type '0'. Other Attribute Types are assigned through IANA action.

When Type is '0', Length MUST be set to the total number of 8-octet blocks in the Attribute, and the Attribute body MUST include a corresponding number of '0' octets. For example, for Lengths of 1, 2, 3, etc., the Attribute body includes 6, 14, 22, etc. '0' octets, respectively.

Receivers ignore any NULL, unknown or malformed Attributes and continue to process any other Attributes in the RIO that follow.

4.2. RIO Requirements

This specification updates [RFC4191] by allowing RIOs to appear in any IPv6 ND messages with the following requirements:

- o Redirect, NA and RA messages MUST NOT include RIOs with the S flag set to '1'; any RIOs received in Redirect, NA and RA messages with S set to '1' MUST be silently ignored.
- o NS and RS messages MAY include some RIOs with S set to '1' and others with S set to '0'.
- o NA/RA responses to RIOs in NS/RS messages with S set to '1' MUST include RIOs with the solicited route information and with S set to '0'. (If the route information solicited by the NS/RS message is incorrect or unrecognized, however, the RIO MUST be silently ignored.)
- o Asserted route information in any RIOs received with S set to '0' SHOULD be considered as "unconfirmed" until the assertion can be verified. Assertion verification can be through a trust anchor such as a trusted on-link router, through a static routing table, or through some other means outside the scope of this document. Any route information that cannot be verified SHOULD be ignored.

The following sections present the classic redirection scenario illustrating an exchange where a trusted on-link router is used to verify RIO assertions. Other IPv6 ND messaging scenarios that can employ some other means of verifying RIO assertions are also acceptable.

4.3. Classic Redirection Scenario

In the classical redirection scenario there are three actors, namely the Source, Router and Target as shown in Figure 3:

4.4. RIO Redirection Scenario

In the RIO redirection scenario, the Source sends initial packets via the Router the same as in the classical scenario. When the Router receives the packets, it searches its routing tables for a route that is assigned to the Target and that covers the destination address of the packet. The Router then includes the route in an RIO in a Redirect message to send back to the Source. The Router sets the S flag in the RIO to '0' to indicate that a prefix is being asserted.

When the Source receives the Redirect message, it prepares an NS message that includes the route information received in the RIO from the Redirect message and with S set to '1' to indicate that route information is being solicited. At the same time, if the Source needs to assert any route information to the Target, it includes the information in RIOs with S set to '0'. The Source then sends the NS message to the Target.

When the Target receives the NS message, it records any route information in RIOs with S set to '0' as unconfirmed route information for the Source pending verification. At the same time, it determines whether the route information included in any RIOs with S set to '1' matches one of its own routes. If so, the Target includes the route information in an RIO with S set to '0' to return in an NA message reply to the Source.

When the Source receives the NA message it can install any RIO information that matches the Redirect RIOs in its routing table. The following sections present more detailed specifications for the Router, Source and Target.

4.4.1. Router Specification

When the Router receives a packet from the Source it searches its routing table for a prefix that covers the destination address (e.g., 2001:db8::/N as depicted in Figure 1), where prefix could be populated in the routing table during DHCPv6 Prefix Delegation [RFC3633], via manual configuration, etc. If the next hop for the prefix is on-link (i.e., a "Target" in the terms of [RFC4861]), the Router then prepares a Redirect message with the Destination Address field set to the packet's IPv6 destination address, with the Target Address field set to the link-local address of the Target, with a TLLAO set to the link-layer address of the Target, and with an RIO that includes route information for the prefix with Route Lifetime, Prf, and S set to 0. The Router then sends the Redirect message to the Source (subject to rate limiting).

4.4.2. Source Specification

According to [RFC4861], a Source that receives a valid Redirect message updates its destination cache per the Destination Address and its neighbor cache per the Target Address. According to [RFC4191], Sources can be classified as Type "A", "B" or "C" based on how they process RIOs, where a Type "C" Source updates its routing table per any RIO elements included in an RA message. Finally, according to [RFC8028], a Type "C" Source operating on a Multi-Prefix Network with multiple default routes can make source address selection decisions based on information in its routing table decorated with information derived from the source of the RIO element.

In light of these considerations, this document introduces a new Type "D" behavior for Sources with the same behavior as a Type "C" Source, but which also process RIO elements in other IPv6 ND messages. Type "D" Sources process Redirect messages with RIO elements by first verifying that the Prefix in the first RIO matches the Destination Address. If the Destination Address does not match the Prefix, the Source discards the Redirect message. Otherwise, the Source updates its neighbor cache per the Target Address and its destination cache per the Destination Address the same as for classical redirection. Next, the Source MAY send an NS message to the Target containing an RIO with the Prefix and Prefix Length and with S set to '1' to elicit an NA response (at the same time, the Source MAY include RIOs with S set to '0' if it needs to assert any route information to the Target).

When the Type 'D' Source receives the solicited NA message from the Target, if the NA includes an RIO with S set to '0' and with a Prefix corresponding to the one received in the Redirect message, the Source installs the route information in its routing table with the Target's address as the next hop. (Note that the Prefix Length received in the NA message MAY be different than the Prefix Length received in the Redirect message. If the Prefix Length in the NA is the same or longer, the Source accepts the Prefix as verified by the Router; if the Prefix Length is shorter, the Source considers the Prefix as unconfirmed.)

After the Source installs the route information in its routing table, it MAY begin sending packets with destination addresses that match the Prefix directly to the Target Instead of sending them to the Router. The Source SHOULD decrement the Route Lifetime and MAY send new NS messages to receive a fresh Route Lifetime (if the Route Lifetime decrements to 0, the Source instead deletes the route information from its routing table). The Source MAY furthermore delete the route information at any time and again allow packets to flow through the Router which may send a fresh Redirect. The Source

SHOULD then again test the route by performing an NS/NA exchange with the Target the same as described above.

After updating its routing table, the Source may receive an unsolicited NA message from the Target with an RIO with new route information. If the RIO Prefix is in its routing table, and if the RIO Route Lifetime value is 0, the Source deletes the corresponding route.

After updating its routing table, the Source may subsequently receive a Destination Unreachable message from the Target with Code '0' ("No route to destination"). If so, the Source again deletes the corresponding route information from its routing table.

4.4.3. Target Specification

When the Target receives an NS message from the Source containing an RIO with S set to '1', it examines the Prefix and Prefix Length to see if it matches one of the prefixes in its routing table. If so, the Target prepares an NA message with an RIO including a Prefix and Prefix Length, any necessary route information, and with S set to '0'. The Target then sends the NA message back to the Source.

If the NS included any RIO options with S set to '0', the Target SHOULD employ a suitable means to verify the asserted route information, and SHOULD reject any route information that cannot be verified.

At some later time, the Target may either alter or deprecate one of its routes. If the Target has asserted route information in RIOs to one or more Sources, the Target SHOULD send unsolicited NA messages with RIOs that assert new route information to alter the route, where a new Route Lifetime value of '0' deprecates the route. If the Target receives a packet with a destination addresses for which there is no matching route for one of its downstream networks, the Target sends a Destination Unreachable message to the Source with Code '0' ("No route to destination"), subject to rate limiting.

4.5. Operation Without Redirects

If the Source has some way to determine the Target's link-local address without receiving a Redirect message from the Router, the Source MAY send an NS message with an RIO directly to the Target with S set to 1, Prefix set to the destination address of an IPv6 packet, Prefix Length set to 128 and all other route information is set to 0.

When the Target receives the NS message, it prepares an NA response with an RIO that includes route information for one of its prefixes

that covers the destination address. The Target then sends the NA message to the Source.

When the Source receives the NA message, it SHOULD consider the route information asserted in the RIO as unconfirmed until it can verify the Target's claim (i.e., as described in Section 4.2).

Any node may also assert route information at any time by sending IPv6 ND messages with RIOs with S set to 0. Recipients of such messages SHOULD consider the route information as unconfirmed until the information can be verified.

4.6. Multiple RIOs

If a Redirect includes multiple RIOs, the Source only checks the destination address for a match against the Prefix in the first RIO.

If an NS/RS message includes multiple RIOs with S set to '1', the neighbor responds to those RIOs which match entries in its routing table.

If an NS/NA/RS/RA message includes multiple RIOs with S set to '0', the neighbor considers all of the route information as unconfirmed until the information can be verified.

4.7. Multicast

Nodes MAY send IPv6 ND messages with RIOs to link-scoped multicast destination addresses including All Nodes, All Routers, and Solicited-Node multicast (see: [RFC4291]). As an example, a node could send unsolicited NA messages to the All Nodes multicast address to alter or deprecate a route it had previously asserted to one or more neighbors.

Nodes MUST be conservative in their use of multicast IPv6 ND messaging to avoid unnecessarily disturbing other nodes on the link.

4.8. Why NS/NA?

Since [RFC4191] already specifies the inclusion of RIOs in RA messages, a natural question is why use NS/NA instead of RS/RA?

First, RA messages are only sent over advertising interfaces [RFC4861]. Source and Target nodes typically connect only downstream networks; hence, they configure their upstream interfaces as non-advertising interfaces.

Second, NS/NA exchanges used by the IPv6 Neighbor Unreachability Detection (NUD) procedure are unicast-based whereas RA responses to RS messages are typically sent as multicast. Since this mechanism must support unicast operation, the use of unicast NS/NA exchanges is required.

Third, the IPv6 ND specification places restrictions on minimum delays between RA messages. Since this mechanism expects an immediate advertisement from the Target in response to the Source's solicitation, only the NS/NA exchange can satisfy this property.

Fourth, the RA message is the "swiss army knife" of the IPv6 ND protocol. RA messages carry numerous configuration parameters for the link, including Cur Hop Limit, M/O flags, Router Lifetime, Reachable Time, Retrans Time, Prefix Information Options, MTU options, etc. The Target must not advertise any of this information to the soliciting Source.

Fifth, RIOs in legacy RA messages cannot encode attributes and therefore may be limited in the route information they can carry.

Finally, operators are deeply concerned about the security of RA messages - so much so that they deploy link-layer security mechanisms that drop RA messages originating from nodes claiming to be an authoritative router for the link [RFC6105].

5. Implementation Status

The IPv6 ND functions and RIOs are widely deployed in IPv6 implementations, however these implementations do not currently include RIOs in IPv6 ND messages other than RAs.

An experimental implementation of [RFC6706] exists, and demonstrates how the Redirect function can be used to carry route information.

6. IANA Considerations

IANA is instructed to create a registry for "RIO Attributes" as discussed in Section 4.1. The registry includes the following initial entry:

0 - the NULL Attribute [draft-templin-6man-rio-redirect]

Other Attribute types are defined through standards action or expert review.

7. Security Considerations

The Redirect message validation rules in Section 8.1 of [RFC4861] require recipients to verify that the IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address. Recipients therefore naturally reject any Redirect message with an incorrect source address.

Other security considerations for IPv6 ND messages that include RIOs are the same as specified in Section 11 of [RFC4861]. Namely, the protocol must take measures to secure IPv6 ND messages on links where spoofing attacks are possible.

A spoofed Redirect message containing no RIOs could cause corruption in the recipient's destination cache, while a spoofed Redirect message containing RIOs could corrupt the host's routing tables. While the latter would seem to be a more onerous result, the possibility for corruption is unacceptable in either case.

"IPv6 ND Trust Models and Threats" [RFC3756] discusses spoofing attacks, and states that: "This attack is not a concern if access to the link is restricted to trusted nodes". "SEcure Neighbor Discovery (SEND)" [RFC3971] provides one possible mitigation for other cases. In some scenarios, it may be sufficient to include only the Timestamp and Nonce options defined for SEND without implementing other aspects of the protocol.

"IPv6 Router Advertisement Guard" [RFC6105] ("RA Guard") describes a layer-2 filtering technique intended for network operators to use in protecting hosts from receiving RA messages sent by nodes that are not among the set of routers regarded as legitimate by the network operator.

Nodes must have some form of trust basis for knowing that the sender of an ND message is authoritative for the prefixes it asserts in RIOs. For example, when an NS/NA exchange is triggered by the receipt of a Redirect, the soliciting node can verify that the RIOs in the NA message match the ones it received in the Redirect message (which originally came from a trusted router).

Nodes that do not wish to provide transit services for upstream networks may also receive IPv6 packets via an upstream interface that do not match any of the their delegated prefixes. In that case, the node drops the packets and observes the "Destination Unreachable - No route to destination" procedures discussed in [RFC4443]. Dropping the packets is necessary to avoid a reflection attack that would cause the node to forward packets received from an upstream interface via the same or a different upstream interface.

8. Acknowledgements

Joe Touch suggested a standalone draft to document this approach in discussions on the intarea list. The work was subsequently transferred to the 6man list, where the following individuals provided valuable feedback: Mikael Abrahamsson, Zied Bouziri, Brian Carpenter, Steinar Haug, Christian Huitema, Tatuya Jinmei, Tomoyuki Sahara.

Discussion with colleagues during the "bits-and-bites" session at IETF98 helped shape this document. Those colleagues are gratefully acknowledged for their contributions.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program and the Boeing Research & Technology (BR&T) enterprise autonomy program.

9. References

9.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.

- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.

9.2. Informative References

- [I-D.templin-v6ops-pdhost] Templin, F., "IPv6 Prefix Delegation for Hosts", draft-templin-v6ops-pdhost-15 (work in progress), October 2017.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<https://www.rfc-editor.org/info/rfc3633>>.
- [RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, DOI 10.17487/RFC3756, May 2004, <<https://www.rfc-editor.org/info/rfc3756>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SECure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <<https://www.rfc-editor.org/info/rfc6706>>.

- [RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<https://www.rfc-editor.org/info/rfc7084>>.
- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <<https://www.rfc-editor.org/info/rfc7368>>.
- [RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi, "Host Address Availability Recommendations", BCP 204, RFC 7934, DOI 10.17487/RFC7934, July 2016, <<https://www.rfc-editor.org/info/rfc7934>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.

Appendix A. Link-layer Address Changes

Type "D" hosts send unsolicited NAs to announce link-layer address changes per standard neighbor discovery [RFC4861]. Link-layer address changes may be due to localized factors such as hot-swap of an interface card, but could also occur during movement to a new point of attachment on the same link.

Appendix B. Interfaces with Multiple Link-Layer Addresses

Type "D" host interfaces may have multiple connections to the link; each with its own link-layer address. Type "D" nodes can therefore include multiple link-layer address options in IPv6 ND messages. Neighbors that receive these messages can cache and select link-layer addresses in a manner outside the scope of this specification.

Appendix C. Change Log

-04 to -05:

- o Removed "Ver" field and version numbers.
- o Included reference to 'draft-templin-v6ops-pdhost'
- o Changed "MAY" to "may" in two places
- o Added text on advertising interfaces

- o Added UAS use case

Authors' Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

James Woodyatt
Google
3400 Hillview Ave
Palo Alto, CA 94304
USA

Email: jhw@google.com