Internet Engineering Task Force
Internet-Draft
Obsoletes: 6434 (if approved)
Intended status: Informational
Expires: September 14, 2017

T. Chown
Jisc
J. Loughney
Nokia
T. Winters
University of New Hampshire

March 13, 2017

IPv6 Node Requirements
draft-clw-rfc6434-bis-01

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 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 14, 2017.

Table of Contents

1. Introduction

This document defines common functionality required from 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 [RFC2460], 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.

**BIS We will need to refer to 2460-bis, as well as 1981-bis and 4291-bis, throughout this document. These are still in flux, but we

will know the final versions of these documents before this -bis is published, so can adapt text here once those updates are complete.**

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

ATM	Asynchronous Transfer Mode
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]

- IPv6 over ATM Networks [RFC2492]

- 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]

- IPv6 over IEEE 802.15.4 Networks [RFC4944]

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]

BIS Do we want a small section somewhere on UDP IPv6 tunneling, and issues like RFC 6935, or 6936?

5. IP Layer

5.1. Internet Protocol Version 6 - RFC 2460

The Internet Protocol Version 6 is specified in [RFC2460]. This specification MUST be supported.

**BIS Again, update for RFC 2460 -bis **

Any unrecognized extension headers or options MUST be processed as described in RFC 2460.

The node MUST follow the packet transmission rules in RFC 2460.

Nodes MUST always be able to send, receive, and process fragment headers. All conformant IPv6 implementations MUST be capable of sending and receiving IPv6 packets; the forwarding functionality MAY be supported. Overlapping fragments MUST be handled as described in [RFC5722].

[RFC6946] discusses IPv6 atomic fragments, and recommends that IPv6 atomic fragments are processed independently of any other fragments,

to protect against fragmenttation-based attacks. [RFC8021] goes further and recommends the deprecation of atomic fragments. Nodes thus MUST not generate atomic fragments.

To mitigate a variety of potential attacks, nodes SHOULD avoid using predictable fragment Identification values in Fragment Headers, as discussed in [RFC7739].

RFC 2460 specifies extension headers and the processing for these headers.

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.

Should a new type of Extension Header need to be defined, its format MUST follow the consistent format described in Section 4 of [RFC6564].

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.

[RFC7112] discusses issues with oversized IPv6 Extension Header chains, and states that when a node fragments an IPv6 datagram, it MUST include the entire IPv6 Header Chain in the First Fragment.

**BIS Wait to see outcome of insertion of EHs issue in 2460-bis, and re-state here? **

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. 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 resliency 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.

5.3. 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 such cases, RFC 4191 can help.

Small Office/Home Office (SOHO) deployments supported by routers adhering to [RFC7084] use RFC 4191 to advertise routes to certain local destinations. Consequently, nodes that will be deployed in SOHO environments SHOULD implement RFC 4191.

5.4. 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. It requires relatively heavyweight provisioning, so is only likely to be used in scenarios where security considerations are particularly important.

There have been relatively few implementations of SEND in common operating systems and platforms, and thus deployment experience has been limited to date.

At this time, SEND is considered optional. Due to the complexity in deploying SEND, its deployment is only likely to be considered where nodes are operating in a particularly strict security environment.

5.5. 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.

5.6. Path MTU Discovery and Packet Size

5.6.1. Path MTU Discovery - RFC 1981

"Path MTU Discovery for IP version 6" [RFC1981] SHOULD be supported. From [RFC2460]:

> 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.

The rules in [RFC2460] and [RFC5722] MUST be followed for packet fragmentation and reassembly.

One operational issue with Path MTU Discovery occurs when firewalls block ICMP Packet Too Big messages. Path MTU Discovery relies on such messages to determine what size messages can be successfully sent. "Packetization Layer Path MTU Discovery" [RFC4821] avoids having a dependency on Packet Too Big messages.

**BIS Add note about 1280 MTU and UDP, as per Mark Andrews’ comments in Berlin? **

5.7. IPv6 Jumbograms - RFC 2675

IPv6 Jumbograms [RFC2675] are an optional extension that allow the sending of IP datagrams larger than 65.535 bytes. IPv6 Jumbograms make use of IPv6 hop-by-hop options and are only suitable on paths in which every hop and link are capable of supporting Jumbograms (e.g., within a campus or datacenter). To date, few implementations exist, and there is essentially no reported experience from usage. Consequently, IPv6 Jumbograms [RFC2675] remain optional at this time.

**BIS Are these used? Do we need to modify the text for that? **

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. Addressing

5.9.1. IP Version 6 Addressing Architecture - RFC 4291

The IPv6 Addressing Architecture [RFC4291] MUST be supported.

**BIS Update to 4291-bis **

**BIS Add note on Why /64? RFC 7421, after the conclusion of the RFC4291-bis (lengthy!!!) discussions on the 64-bit IID topic. But no need for /127 p2p text RFC 6164. And no need for note on IID significance, as per RFC 7136. **

5.9.2. 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. There are, for example, benefits to multiple addresses for privacy reasons, or to assigning hosts a whole /64 to avoid the need for host-based NAT.

5.9.3. IPv6 Stateless Address Autoconfiguration - RFC 4862

Hosts MUST support IPv6 Stateless Address Autoconfiguration as defined in either [RFC4862] or [RFC7217]. It is recommended 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. 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.

5.9.4. 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.

[RFC7721] discusses general privacy issues with IPv6 addressing.

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.

5.9.5. 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.

5.9.6. 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 Router Advertisements, 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 a router, IPv6 nodes using DHCP for address assignment MAY initiate DHCP to obtain IPv6 addresses and other configuration information, as described in Section 5.5.2 of [RFC4862].

5.10. Multicast Listener Discovery (MLD) for IPv6

**BIS MLDv2 only?

Nodes that need to join multicast groups MUST support MLDv1 [RFC2710]. MLDv1 is needed by any node that is expected to receive and process multicast traffic. Note that Neighbor Discovery (as used on most link types -- see Section 5.2) depends on multicast and requires that nodes join Solicited Node multicast addresses.

MLDv2 [RFC3810] extends the functionality of MLDv1 by supporting Source-Specific Multicast. The original MLDv2 protocol [RFC3810] supporting Source-Specific Multicast [RFC4607] supports two types of "filter modes". Using an INCLUDE filter, a node indicates a

multicast group along with a list of senders for the group from which it wishes to receive traffic. Using an EXCLUDE filter, a node indicates a multicast group along with a list of senders from which it wishes to exclude receiving traffic. In practice, operations to block source(s) using EXCLUDE mode are rarely used but add considerable implementation complexity to MLDv2. Lightweight MLDv2 [RFC5790] is a simplified subset of the original MLDv2 specification that omits EXCLUDE filter mode to specify undesired source(s).

Nodes SHOULD implement either MLDv2 [RFC3810] or Lightweight MLDv2 [RFC5790]. Specifically, nodes supporting applications using Source-Specific Multicast that expect to take advantage of MLDv2's EXCLUDE functionality [RFC3810] MUST support MLDv2 as defined in [RFC3810], [RFC4604], and [RFC4607]. Nodes supporting applications that expect to only take advantage of MLDv2's INCLUDE functionality as well as Any-Source Multicast will find it sufficient to support Lightweight MLDv2 as defined in [RFC5790].

If a node only supports applications that use Any-Source Multicast (i.e, they do not use Source-Specific Multicast), implementing MLDv1 [RFC2710] is sufficient. In all cases, however, nodes are strongly encouraged to implement MLDv2 or Lightweight MLDv2 rather than MLDv1, as the presence of a single MLDv1 participant on a link requires that all other nodes on the link operate in version 1 compatibility mode.

When MLDv1 is used, the rules in the Source Address Selection for the Multicast Listener Discovery (MLD) Protocol [RFC3590] MUST be followed.

6. DHCP versus Router Advertisement Options for Host Configuration

**BIS this section probably needs rewriting **

In IPv6, there are two main protocol mechanisms for propagating configuration information to hosts: Router Advertisements (RAs) and DHCP. Historically, 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. That said, identifying the exact line on whether a particular option should be configured via DHCP versus an RA option has not always been easy. 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.

7. DNS and DHCP

7.1. 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) [RFC2671] 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].

**BIS Add DNS-SD? **

7.2. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) - RFC 3315

7.2.1. Other Configuration Information

IPv6 nodes use DHCP [RFC3315] to obtain address configuration information (see Section 5.9.6) 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.

7.2.2. Use of Router Advertisements in Managed Environments

Nodes using the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) are expected to determine their default router information and on-link prefix information from received Router Advertisements. There is no defined DHCPv6 Gateway option.

7.3. IPv6 Router Advertisement Options for DNS Configuration - RFC 6106

Router Advertisements have historically limited options to those that are critical to basic IPv6 functioning. 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. [RFC5006] was published as an Experimental document in 2007, and recently, a revised version was placed on the Standards Track [RFC6106].

Implementations SHOULD implement the DNS RA option [RFC6106].

8. IPv4 Support and Transition

IPv6 nodes MAY support IPv4.

8.1. Transition Mechanisms

8.1.1. Basic Transition Mechanisms for IPv6 Hosts and Routers - RFC 4213

If an IPv6 node implements dual stack and tunneling, then [RFC4213] MUST be supported.

9. Application Support

9.1. 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].

9.2. 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.

10. Cellular Host

IPv6 for 3GPP [RFC7066] lists IPv6 Functionalities that need to be implemented above and beyond the recommendations in this document. Additionally a 3GPP IPv6 Host MAY implement [RFC7278] for delivering IPv6 prefixes on the LAN link.

11. 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 provides channel security at the Internet layer, 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 provides sufficient flexibility and granularity that individual TCP connections can (selectively) be protected, etc.

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), 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 [RFC5996].

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.

**BIS Add note on security in IPv4-only networks? RFC 7123? Relevant? **

11.1. 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].

11.2. 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" [RFC4835]. IPv6 nodes implementing the IPsec Architecture MUST conform to the requirements in [RFC4835]. Preferred cryptographic algorithms often change more frequently than security protocols. Therefore, implementations MUST allow for migration to new algorithms, as RFC 4835 is replaced or updated in the future.

BIS update to 7321bis

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)" [RFC4307]. IPv6 nodes implementing IKEv2 MUST conform to the requirements in [RFC4307] and/or any future updates or replacements to [RFC4307].

BIS update to 4307bis

12. Router-Specific Functionality

This section defines general host considerations for IPv6 nodes that act as routers. Currently, this section does not discuss routing-specific requirements; for the case of typical home routers, [RFC7084] defines basic requirements for customer edge routers.

BIS Sync here with work by John Brzozowski et al. in draft-ali-ipv6rtr-reqs-02

12.1. 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 (MLD) [RFC2710]). The Router Alert option will need to be implemented whenever protocols that mandate its usage (e.g., MLD) are implemented. See Section 5.10.

12.2. 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.

12.3. 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]). However, 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.

13. 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.

**BIS This is a little thin. Add Netconf, restconf, yang models? **

BIS add the network polling/syslod nd for none DHCPv6 network tracking.

13.1. Management Information Base (MIB) Modules

**BIS Address MIB Obsolete draft

The following two MIB modules SHOULD be supported by nodes that support a Simple Network Management Protocol (SNMP) agent.

13.1.1. IP Forwarding Table MIB

The IP Forwarding Table MIB [RFC4292] SHOULD be supported by nodes that support an SNMP agent.

13.1.2. Management Information Base for the Internet Protocol (IP)

The IP MIB [RFC4293] SHOULD be supported by nodes that support an SNMP agent.

14. Constrained Devices

**BIS Should we add notes on constrained devices, and power efficiency here in a new section? Talk about resource management in nodes. Low power operation.

15. 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 11 above.

16. Authors and Acknowledgments

16.1. Authors and Acknowledgments (Current Document)

For this version of the IPv6 Node Requirements document, the authors would like to thank **BIS Add new acknowledgements for significant comments ** for their contributions.

16.2. 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.

16.3. Authors and Acknowledgments from RFC 4294

The original version of this document (RFC 4294) was written by the IPv6 Node Requirements design team:

Jari Arkko
jari.arkko@ericsson.com

Marc Blanchet
marc.blanchet@viagenie.qc.ca

Samita Chakrabarti
samita.chakrabarti@eng.sun.com

Alain Durand
alain.durand@sun.com

Gerard Gastaud
gerard.gastaud@alcatel.fr

Jun-ichiro Itojun Hagino
itojun@iijlab.net

Atsushi Inoue
inoue@isl.rdc.toshiba.co.jp

Masahiro Ishiyama
masahiro@isl.rdc.toshiba.co.jp

John Loughney
john.loughney@nokia.com

Rajiv Raghunarayan
raraghun@cisco.com
Shoichi Sakane
shouichi.sakane@jp.yokogawa.com

Dave Thaler
dthaler@windows.microsoft.com

Juha Wiljakka
juha.wiljakka@Nokia.com

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.

17. 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. Added 6LoWPAN to link layers

2. Removed DOD IPv6 Profile updates

3. Removed IPv6 Mobility RFC6275

18. 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.

19. References

19.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[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>.

[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>.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, DOI 10.17487/RFC2671, August 1999, <http://www.rfc-editor.org/info/rfc2671>.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <http://www.rfc-editor.org/info/rfc2710>.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <http://www.rfc-editor.org/info/rfc2711>.

[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, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3590] Haberman, B., "Source Address Selection for the Multicast Listener Discovery (MLD) Protocol", RFC 3590, DOI 10.17487/RFC3590, September 2003, <http://www.rfc-editor.org/info/rfc3590>.

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, DOI 10.17487/RFC3596, October 2003, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4291>.

[RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292, DOI 10.17487/RFC4292, April 2006, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, DOI 10.17487/RFC4307, December 2005, <http://www.rfc-editor.org/info/rfc4307>.

[RFC4311] Hinden, R. and D. Thaler, "IPv6 Host-to-Router Load Sharing", RFC 4311, DOI 10.17487/RFC4311, November 2005, <http://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", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <http://www.rfc-editor.org/info/rfc4604>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <http://www.rfc-editor.org/info/rfc4607>.

[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>.

[RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, DOI 10.17487/RFC4835, April 2007, <http://www.rfc-editor.org/info/rfc4835>.

[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>.

[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>.

[RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <http://www.rfc-editor.org/info/rfc5453>.

[RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <http://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, <http://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, <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>.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, DOI 10.17487/RFC5996, September 2010, <http://www.rfc-editor.org/info/rfc5996>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

[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>.

[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>.

[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>.

[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>.

[RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, DOI 10.17487/RFC7048, January 2014, <http://www.rfc-editor.org/info/rfc7048>.

[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>.

[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>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7559>.

[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>.

19.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://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, <http://www.rfc-editor.org/info/rfc2205>.

[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>.

[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, <http://www.rfc-editor.org/info/rfc2491>.

[RFC2492] Armitage, G., Schulter, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <http://www.rfc-editor.org/info/rfc2492>.

[RFC2590] Conta, A., Malis, A., and M. Mueller, "Transmission of IPv6 Packets over Frame Relay Networks Specification", RFC 2590, DOI 10.17487/RFC2590, May 1999, <http://www.rfc-editor.org/info/rfc2590>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <http://www.rfc-editor.org/info/rfc2675>.

[RFC3146] Fujisawa, K. and A. Onoe, "Transmission of IPv6 Packets over IEEE 1394 Networks", RFC 3146, DOI 10.17487/RFC3146, October 2001, <http://www.rfc-editor.org/info/rfc3146>.

[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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc3542>.

[RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, DOI 10.17487/RFC3678, January 2004, <http://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, <http://www.rfc-editor.org/info/rfc6275>.

[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>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <http://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, <http://www.rfc-editor.org/info/rfc4191>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4380>.

[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>.

[RFC4584] Chakrabarti, S. and E. Nordmark, "Extension to Sockets API for Mobile IPv6", RFC 4584, DOI 10.17487/RFC4584, July 2006, <http://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, <http://www.rfc-editor.org/info/rfc4821>.

[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, <http://www.rfc-editor.org/info/rfc4884>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <http://www.rfc-editor.org/info/rfc4944>.

[RFC5006] Jeong, J., Ed., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Option for DNS Configuration", RFC 5006, DOI 10.17487/RFC5006, September 2007, <http://www.rfc-editor.org/info/rfc5006>.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <http://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, <http://www.rfc-editor.org/info/rfc5072>.

[RFC5121] Patil, B., Xia, F., Sarikaya, B., Choi, JH., and S. Madanapalli, "Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks", RFC 5121, DOI 10.17487/RFC5121, February 2008, <http://www.rfc-editor.org/info/rfc5121>.

[RFC6563] Jiang, S., Conrad, D., and B. Carpenter, "Moving A6 to Historic Status", RFC 6563, DOI 10.17487/RFC6563, March 2012, <http://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, <http://www.rfc-editor.org/info/rfc7066>.

[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, <http://www.rfc-editor.org/info/rfc7084>.

[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, <http://www.rfc-editor.org/info/rfc7278>.

[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>.

[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>.

[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
Nokia
200 South Mathilda Ave.
Sunnyvale, CA 94086
USA

Phone: +1 650 283 8068
Email: john.loughney@nokia.com

Tim Winters
University of New Hampshire
InterOperability Laboratory
Durham NH
United States

Email: twinters@iol.unh.edu

DOTS
Internet-Draft
Intended status: Standards Track
Expires: November 4, 2017

J. Francois
Inria
A. Lahmadi
University of Lorraine - LORIA
M. Davids
G. Moura
SIDN Labs

May 3, 2017

IPv6 DOTS Signal Option
draft-francois-dots-ipv6-signal-option-02

Abstract

DOTS client signal using original signal communication channel can expect service degradation and even service disruption as any other service over Internet but in more severe conditions because the signal may have to be transmitted over congested paths due to the denial-of-service attack.

This document specifies a fall-back asynchronous mechanism using an intermediate agent to store DOTS signal information during a limited period of time. This mechanism allows a DOTS server to request a signal information stored by a DOTS client when no heartbeat is received from the DOTS client. This intermediate agent called DOTS Signal Repository have to be connected to the DOTS client and server independently. The repository must be located and/or reached through one or multiple network paths, preferably as most as possible disjoint from regular signal channel, in order to increase its reachability. The document introduces a set of support protocols to build the asynchronous communication between the DOTS cient, server and the repository.

This Internet-Draft will expire on November 4, 2017.

Table of Contents

1. Introduction

1.1. Overview

A distributed denial-of-service (DDoS) attack aims at rendering machines or network resources unavailable. These attacks have grown in frequency, intensity and target diversity [I-D.ietf-dots-requirements]. Moreover, several protocols have been utilized to amplify the intensity of the attacks [kuhrer2014exit], peaking at several hundred gigabits per second.

DDoS Open Threat Signaling (DOTS) aims at defining a common and open protocol to signal DDoS attacks to facilitate a coordinated response to these attacks. This document specifies an asynchronous signaling method that MAY be used between a DOTS client and server instead of relying on purely synchronous communication as specified in [I-D.ietf-dots-signal-channel]. Indeed initial signaling should be done in real-time through connections between DOTS clients and severs such that a client can forward signal information as soon as the attack is detected. However, the signaling messages may have to be forwarded through paths impacted by the attack itself, i.e. highly congested. Asynchronous signaling in this document is an additional mechanism which MAY propagate signal information in a less reactive manner due to the use of an asynchronous communication channel but through alternative paths in the network. It increases the reachability of the DOTS server which will then in charge of requesting the mitigation.

The proposed mechanism constitutes an additional signaling channel but MUST NOT replace the original signaling channel used between DOTS client and servers as the one defined in [I-D.ietf-dots-signal-channel].

To perform asynchronous communication, this document introduces DOTS Signals Repository (DSR) which represents datastores where DOTS clients can send signal information. This information is then stored and the DOTS server can request it. In addition to provide a general process for achieving asynchronous signaling, this document introduces also a set of protocols which can support it.

1.2. Terminology

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

The terms DOTS client, DOTS server, DOTS gateway, DOTS agents refers to the terminology introduced in [I-D.ietf-dots-architecture].

The following terms are introduced:

- DSR (DOTS signal repository): intermediate agent able to store signal from clients during a limited period of time and which can be requested by DOTS servers.

- Opportunistic DOTS signal: an IPv6 packet containing the signaling attributes of an attack within the Hop-by-Hop extension header. The purpose is the same as the DOTS signal. It is used to request help for mitigating the attack.

- DOTS opportunistic-capable router: a router with the capacity to decode the opportunistic DOTS signal and re-embed such an information in other IPv6 packets.

- All DOTS opportunistic-capable agents are defined as the DOTS agents supporting the opportunistic DOTS signal processing.

2. Asynchronous DOTS signaling

2.1. Motivation

The traffic generated by a DDoS can be characterized according to various parameters, such as the layer (IP/ICMP or application), maximum and instant throughput, among others. Regardless its nature, we assume that for most cases, a DOTS client will be able to signal back one or few messages, during the attack, to the DOTS phase.

We have the same behavior in other DDoS attacks. For instance, on November 30th and December 1st, 2015, the Root DNS system was hit by an application layer (DNS) attack [rootops-ddos]. Each one of the 13 root server letters (A-M) was hit by attacks peaking at 5 million queries per second. By utilizing the RIPE Atlas DNSMON infrastructure, we can see that during the DDoS attacks, most of the root server letters remained reachable and able to respond to the DNS request sent by the probes employed by the DNSMON [ripe-dnsmon-ddos]. Few letters, however, had a packet loss rate of more than 99%. The DNSMON probes, however, experience mostly delays in their DNS requests instead.

As regular signaling from the DOTS client to the DOTS server or the DOTS gateway might be affected by the attack traffic, it is important to maximize the delivering success of the signals by using alternative packets and/or paths to deliver it. As a result, it forces to have intermediate agents, DSRs, able to catch DOTS signals delivered through those auxiliary mechanisms. However, those agents MUST not always forward DOTS signal to the server in order to limit the induced overhead. Only if the regular signal is not received by

the server, retrieving the signal from the DSRs is required, which has thus initiated by the DOTS server itself.

2.2. Architecture

DSR (DOTS Signal Repository) are additional REQUIRED datastores. They are integrated in the DOTS architecture [I-D.ietf-dots-architecture]as highlighted in Figure 1. (1) refers to the regular signaling. (2) and (3) refers to the proposed auxiliary mechanism. As shown in this figure, DSRs act as synchronizing agent where the DOTS client drops signal information (2) (attack details). On the contrary, the server will retrieve this information (3).

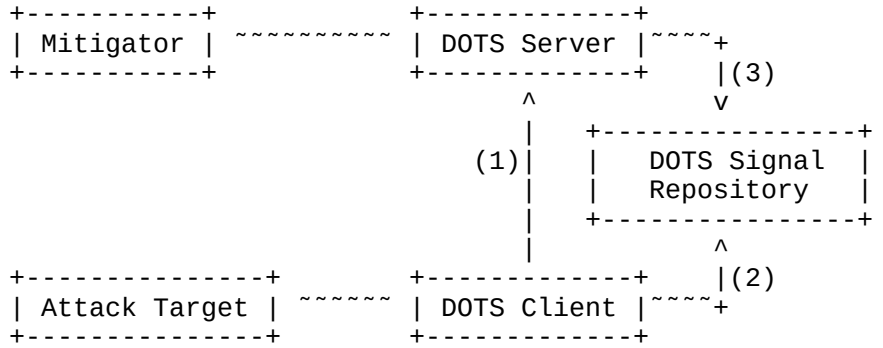


Figure 1: Asynchronous signaling

2.3. Asynchronous process

This section describes the process of asynchronous DOTS signal processing. In Figure Figure 1, there are two main communication channels which are not synchronized:

- Between DOTS client and DSR: the client sends DOTS signal information when a condition is satisfied. The condition MUST be configured by the user but SHOULD be linked to information provided by Attack target. For instance, when an attack is detected, the client connects to DOTS server and in parallel sends signal to one or more DSRs.

- Between DOTS server and DSR: the server will retrieve signal information when a specific condition occurs. Such a condition is linked with the probable occurrence of an attack. The server can infer this condition when the client is not responsive anymore. In [I-D.ietf-dots-signal-channel], an heartbeat mechanism is defined. Hence, when no heartbeat is received from the client, the server MUST try to get signal information by the asynchronous communication channel.

Each communication channel can implement its own protocol. They are NOT REQUIRED to be the same.

We have to note that the client condition to provide signals to the DSR can be weaker than regular synchronous signaling between DOTS client and server. Indeed, a client can signal to the DSR some suspect activities for which no mitigation is required yet. However, when the supposed attack is stronger provoking client disruption, the latter is not able to provide any type of signaling anymore and the server can thus retrieve information on prior stored signals.

2.4. Protocol requirements

DOTS signaling requirements are documented in [I-D.ietf-dots-requirements].

GEN-003 (Bidirectionality) requires that signal channel MUST enable asynchronous communications between DOTS agent by allowing unsolicited messages. Asynchronous signaling described in the current document allows DOTS client to provide signals, which can be retrieved later by DOTS server(s).

Because of this mechanism, there are requirements which are not supported: OP-002 (Session Health Monitoring), OP-003 (Session Redirection). Therefore, the fall-back asynchronous DOTS signaling is an additionnal mechanism and MUST NOT replace regular signaling as described in [I-D.ietf-dots-signal-channel].

It is particularly designed to fulfill GEN-002 (Resilience and Robustness) by increasing the signal delivery success even under the severely constrained network conditions imposed by particular attack traffic.

In addition, the fall-back asynchronous DOTS signal MUST specify a TTL (Time-to-Live) used by DSRs to store received signal in a limited period of time. It is different from mitigation lifetime but MUST be lower or equals.

3. DOTS Signal Repository

DSRs have to be deployed and distributed in order to enhance its reachability by the DOTS client. It is NOT REQUIRED that all DOTS agents use the same set of DSRs and a DOTS client and server SHOULD define their own set regarding their particular context, e.g. the network topology. The Data channel [I-D.ietf-dots-data-channel] between client and server MUST be used to configure it. As an example in the case of the inter-domain scenario, the DOTS server can inform the DOTS client to use DSRs scattered in multiple domains.

There is no restriction on the environment where DSRs can be deployed. Two types of DSRs are mainly considered:

- Routers: they have low capacity to process and store received signals but they are well distributed by nature in the network.
- Servers or stations: they provide higher computational power and storage but are less distributed

Selection of protocols to use for asynchronous signaling MUST take into account those specificities.

4. Protocol

4.1. Between DSR and DOTS server

To retrieve signals, the DOTS server MUST request the DSRs. Standard protocols can be used. Requests from the DOTS server MUST specify a client identifier and the DSRs returns all stored signal related to this client. The protocol MUST provide integrity and authenticity and SHOULD guarantee confidentiality. To limit entailed overhead lightweight protocol SHOULD be used. COAP [RFC7252] over DTLS [RFC6347] is RECOMMENDED when DSRs are servers. In the case of routers acting as DSR, network management protocol such as SNMP [RFC1157] or NETCONF [RFC6241] SHOULD be leveraged.

4.2. Between DSR and DOTS client

The signal sent by the DOTS client to the DSR is more prone to be affected by attack traffic due to its proximity to the attack victim. Similar protocol as between DSR and DOTS server can be used but it is RECOMMENDED to convey the signal over multiple paths to increase the reachability sucess

This document introduces two mechanisms to deliver the signal from the DOTS client to the DSR: IPv6 opportunistic signaling using Hop-by-Hop Option header; source routing using IPv6 Segment Routing Header (SRH).

4.2.1. Opportunistic DOTS signaling

This section specifies a signalling mechanism that instead of designing a new application-layer protocol, it utilizes the IPv6 Hop-by-Hop header [RFC2460]. This header SHOULD be processed by intermediate devices and it MUST be the first header in IPv6 extension headers [RFC7045].

In such a particular scenario, DSRs are intermediate routers capable of processing the option. DOTS server MAY also receive IPV6 packets with the Hop-by-Hop option and could thus process it directly. It is till considered as asynchronous since the server MAY NOT receive the initial packet emitted by the client but a copy of the signal in another packet (done by an intermediate DSR/router). Otherwise, it MUST connect to the DSR (section Section 4.1)

The new option containing the attributes of the signalling message is included in an opportunistic way in available IPv6 packets leaving a network element. The DOTS client will thus embed the signalling attributes into outgoing IPv6 packets not necessarily going to the DOTS server. Intermediate routers receiving such a packet will examine it and embed the same information into other IPv6 packets. domain in this opportunistic way to increase the probability that such a packet will be finally forwarded to a DOTS gateway or Server, but also in controlled way to avoid that the mechanism is exploited for a malicious purposes.

Only the Hop-by-Hop options header allows such behavior and using Destination options header is not enough to make the DOTS signal going through the network in an opportunistic way. Each network element recognizing this new option will select the best fitted IPv6 packets to deliver the signal to the DOTS DSRs. For this reason the Hop-by-Hop header option is essential to make such behavior compared to other existing IPv6 extension headers [RFC6564].

The goal is to provide an efficient mechanism where nodes in a IPv6 network facing a DDoS attack can deliver a DOTS signal message sent by a DOTS client to the DOTS server. The specified mechanism does not generate transport packets to carry the DOST signal message but it only relies on existing IPv6 packets in the network to include inside them a hop-by-hop extension header which contains an encoded DOTS signal message. The solution defines a new IPv6 Hop-by-Hop header option with the semantic that the network node SHOULD include

the option content within one or multiple outgoing IPv6 packets available in that network node.

4.2.1.1. Hop-by-Hop option encoding

According to [RFC2460], options encoded into the IPv6 Hop-by-Hop header are formatted as Type-Length-Values (TLVs). The option for opportunistic DOTS signal is thus defined as described in Figure 2

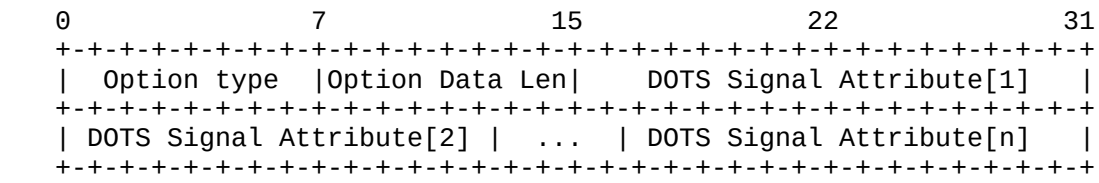


Figure 2: Hop-by-Hop option encoding

The first byte defines the Hop-by-Hop Option type number allocated to the DOTS opportunistic signalling. This number is not yet fixed but the first three bits MUST be set to 0. The first two zero bits indicate that routers which cannot handle the DOTS signal option will continue to process other options. The third 0 bit means that the option processing will not change the packet's final destination [RFC2460].

The second byte contains the length of the option content. The content of the DOTS Signal option is a variable-length field that contains one or more type-length-values (TLV) encoded DOTS signal attributes, and has the format described in Figure 3.

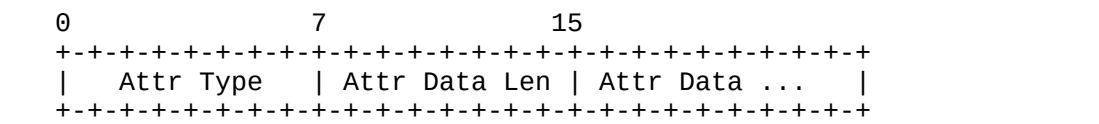


Figure 3: Hop-by-Hop option encoding

The Attr Type is 8-bit identifier of a DOTS signal attribute.

The Attr Data Len is 8-bit unsigned integer which is the length of Attr Data in bytes.

The Attr Data is variable-length field that contains the data of the attribute.

Since using TLVs in Hop-by-Hop options is known to be a factor of attacks [I-D.krishnan-ipv6-hopbyhop], DOTS attributes are encoded with fixed length when possible.

4.2.1.2. DOTS signal Option attributes

The first attribute embedded into the opportunistic DOTS signal is a TTL (Time-to-Live) field which indicates the maximum number of retransmission of the signal into another IPv6 packets until it MUST be discarded. Remaining attributes are similar to the header fields described in [I-D.ietf-dots-signal-channel] used to convey a DOTS signal through a HTTP POST.

The sequence of attributes to be inserted within the header MUST start with fixed-length attributes which are defined in the following order:

- TTL: Time-to-Live. This is a mandatory attribute encoded in one byte.
- Flags: one byte is reserved for flags.
  The first bit indicates the type of the IP address of the host: 0 for IPv4, 1 for IPv6. The second bit indicate if the protocol to use is TCP (1) or UDP (0). The third bit indicates if the message is signed. The remaining bit are not used yet.
- host: the IP address of the DOTS server where the signal option SHOULD be delivered. Depending on the flags, this field is encoded in 4 or 16 bytes.
- port: the listening port of the DOTS server. It is encoded in 2 bytes.

The remaining attributes MUST be TLV encoded, and they are defined in the following order:

- policy-id: defined in [I-D.ietf-dots-signal-channel].
- target-ip: defined in [I-D.ietf-dots-signal-channel].
  However, each address or prefix is encoded in its own TLV element. The distinction between IPv4 and IPv6 is done over the length of the value.
- target-port: defined in [I-D.ietf-dots-signal-channel].
  However, each target port is encoded in its own TLV element.
- target-protocol: defined in [I-D.ietf-dots-signal-channel]
  However each target protocol is encoded in its own TLV element.

o lifetime (lt): lifetime of the mitigation request defined in [I-D.ietf-dots-signal-channel]

The encoded attributes MUST be included in the option header in the order defined above.

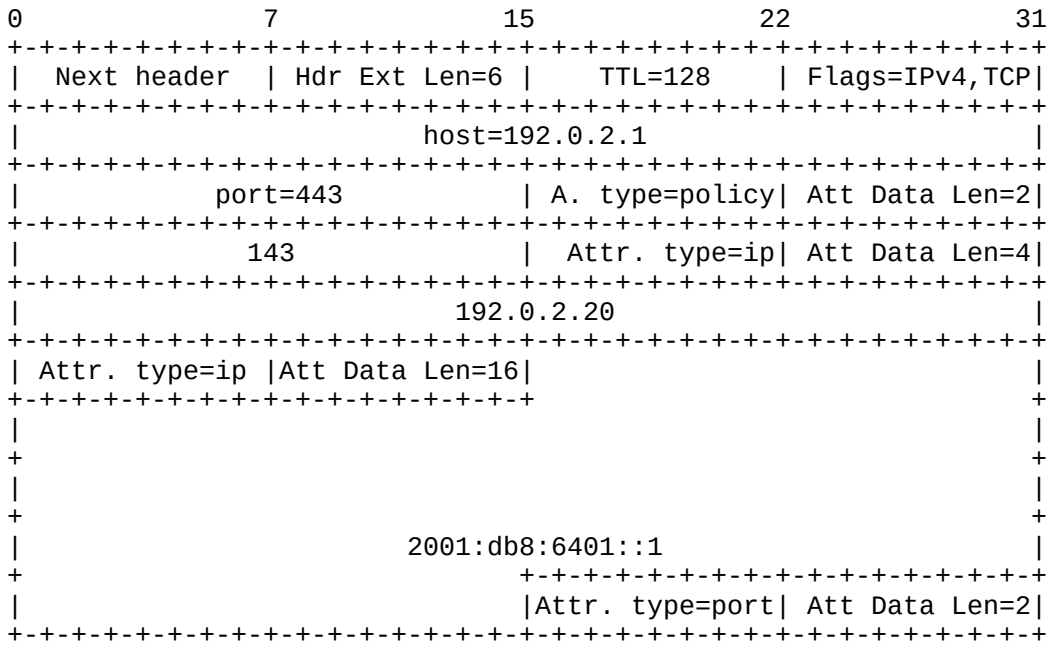
Table 1 provides the value of types that are used by the TLV encoded attributes.

Attribute type	Value
policy-id	0
target-ip	1
target-port	2
target-protocol	3
lifetime	4

Table 1: TLV encoded attributes types

4.2.1.3. Example

Following is an example of an encoded Hop-by-Hop Option header to signal that a web service is under attack.



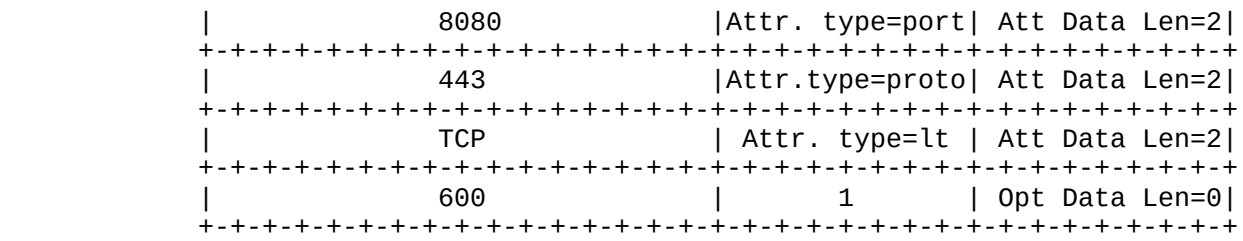


Figure 4: Example of Hop-by-Hop DOTS signal encoding

In the previous example, the message is not signed and terminates with padding. If it is the case, then the signature MUST BE added at the end such that the integrity and authenticity can be checked by the DOTS server or gateway. The TTL attributes MUST be excluded from the signature calculation.

4.2.1.4. Option Processing

4.2.1.4.1. Opportunistic DOTS signal initialization by a DOTS client

When a DOTS client needs to inform the DOTS server that it is under attack, it firstly makes a connection attempt and applies the mechanisms described in [I-D.ietf-dots-signal-channel].

In addition, it MAY activates an opportunistic mechanism to include the Hop-by-Hop header option specified in this document in one or multiple available IPv6 packets leaving the node. Because the DOTS client location is independent of the signalling, it can be positionned in a part of the network where there is no passing-by traffic which can serve for opportunistic signalling. DOTS client MAY also create and emit IPv6 datagrams without payload but with the signal encoded in the Hop-by-Hop option header.

Otherwise, the selection of packets has to be configured a priori. The configuration is composed of a sequence of rules defined in a hierarchical order such that they are triggered in a sequential manner.

The selection of packets has to be configured a priori. The configuration is composed of a sequence of rules defined in a hierarchical order such that they are triggered in a sequential manner.

Each rule is defined by:

- a set of filters over the IPv6 packet headers. Only packets matching those filters are selected for opportunistic signalling.

For instance, only packets heading to a given subnetwork or to specific address close to a DOTS server can be selected to increase the chance to reach the latter.

- o a ratio to select only a proportion of packets matching the filters in order to limit the induced overhead of the opportunistic signalling.

- o a timeout until the rule is active and selected IPv6 packets embed the DOTS opportunistic signal.

The objective is to apply each ordered rule after another according to their timeouts. The first rule is triggered immediately after the opportunistic signalling is activated.

In all cases (embedding information into an exsiting packet or creating an new pakket with no payload), the client MUST avoid fragmentation.

Although the definition of rules MUST be configured by the user. It is RECOMMENDED to order them inversely related to the number of packets that would be selected. This can be approximated regarding the definition of filters. The core idea is to benefit from the first instants of the attack before losing connectivity by using a maximum number of outgoing packets to include the DOTS signalling option. It is thus RECOMMENDED to define the first as matching all IPv6 packets with a ratio equals one to rapidly disseminate the information but with a short timeout to limit the implied overhead.

Here is the an example of rules:

1. all outgoing IPv6 packets with a 10 second timeout

2. all outgoing IPv6 packets with a ratio of 10% and a 1 minute timeout

3. all outgoing multicast IPv6 packets with a ratio of 10% and a 1 minute timeout

4. all outgoing IPv6 packets heading to the DOTS server with a ratio of 100% and a one hour timeout

4.2.1.4.2. Processing by a non DOTS opportunistic-capable router

When receiving an opportunistic DOTS signal encoded in a IPv6 packet, a non DOT opportunistic capable router simply skips the Hop-by-Hop option and continue the normal processing of the IPv6 packet because the option type MUST start with three zero bits.

4.2.1.4.3. Processing by a DOTS opportunistic-capable router

A DOTS opportunistic-capable router MUST store DOTS signalling information whose it is aware of. If a router processes an IPv6 DOTS opportunistic signal and supports this option, it first checks if it has already stored the associated information. In that case, the router simply skips the option and continues the normal processing otherwise it stores the encoded information in order to embed it again in other IPv6 packets similarly to the DOTS client. Hence, a set of rules are also defined in advance and are triggered upon the reception of a new opportunistic DOTS signal. Once all rule have been applied, signalling information MUST be discarded by the router. When embedding the information into other IPv6 packets, the router MUST decrease the TTL by one since opportunistic signalling does not prevent loops in the dissemination of signalling.

4.2.1.4.4. Processing by a DOTS opportunistic-capable gateway

If a DOTS gateway has DOTS capabilities, it will apply the same strategy as a DOTS client by making attempts of direct connections to the DOST server and in addition it inserts the Hop-by-Hop header DOTS signalling option in leaving IPv6 packets using the strategy specified above.

4.2.1.4.5. Processing by a DOTS opportunistic-capable server

When the IP layer of the host where the DOTS server is running receives an IPv6 packet carrying a Hop-by-Hop DOTS signal option header it MUST extracts the content of the option and provides the attributes data to the server program.

4.2.1.5. Deployment considerations

This mechanism will be potentially used by networks with IPv6 capable elements and requires that of IPv6 traffic exist in the network during the attack. The existing IPv6 traffic to be used could be of any type from management or user levels. It is also important to emphasize that while our mechanism utilizes an IPv6 header field, it can also be used to signal IPv4 attacks as well - given that the network devices are dual stacked.

IPv6 extension headers are often rate-limited or dropped entirely. To be able to use the mechanism specified in this document, network operators need to avoid discarding packets or ignoring the processing of the hop-by-hop option on their deployed network elements. However, instead of dropping or ignoring packets with hop-by-hop option carrying DOTS signal, they need to assign these packets to slow forwarding path, and be processed by the router’s CPU. This

behavior will not affect the performance of the network devices since the network is already facing a DDoS attack and fast forwarding paths are saturated by the attacker traffic.

If the DOTS server, gateway and the client are located in the same administrative domain, marking the IPv6 packets with the proposed hop-by-hop header option could be done in a straight forward way, while considering that an agreement exists inside the domain to avoid dropping or rate limiting of IPv6 extension headers as described above. The proposed mechanism becomes less practical and difficult to deploy when the DOST server is running on the Internet. In such scenario, the mechanism could be used in the intra-domain part to deliver the hop-by-hop option carrying the DOTS signal until it reaches a DOTS gateway located in the same domain as the client, then the gateway will apply mechanisms provided by the DOTS transport protocol [I-D.ietf-dots-signal-channel] to inform the server running on Internet about the attack. This deployment scenario requires that at least one DOTS gateway is deployed in the same domain than the DOTS client.

4.2.1.6. Impact on existing IP layer implementations

For this option to be applicable within an IP system, it requires modifications to existing IP layer implementation. At DOTS capable nodes (client, gateway and server), it requires a service interface used by upper-layer protocols and application programs to ask the IP layer to insert and listen to the Hop-by-Hop header option in IPv6 packets. A DOTS client invokes the service interface to insert the option, A DOTS gateway invokes the service interface for listening and inserting the option, and finally a DOTS server only invokes the service interface to listen to the DOTS signalling option.

Intermediate nodes (routers or middle boxes) IP layer needs to be extended to perform processing of the new Hop-by-Hop header option. They mainly parse the first host attribute of the option and make a selection of a leaving IPv6 packet where the option will be inserted.

Every node inserting the new proposed Hop-by-Hop option SHOULD only select IPv6 packets with enough left space to avoid fragmentation.

4.2.2. IPv6 SRH

DOTS signalling may be carried using IPv6 source routing header. Details will be provided in a later version of this document.

5. Security Considerations

Any IPv6 header option could be used by an attacker to create an attack on the routers and intermediate boxes that process packets containing the option. The proposed IPv6 option in this document MAY be abused by an attacker to create a covert channel at the IP layer where data is hidden inside the content of the option [RFC6564]. However, this attack is not specific to the proposed option and it is a known issue of IPv6 header extensions and options. The option MAY also be used by an attacker to forge or modify opportunistic DOTS signal leading to trigger additional processing on intermediate nodes and DOTS servers.

However the proposed option should be only initiated by a DOTS client and information embedded in new IPv6 messages by opportunistic DOTS capable routers. Defining proper policies to filter all messages with this option set and originated from other nodes would limit security issues since these DOTS opportunistic-capable agents SHOULD be trustworhy.

In addition, the message MAY be signed using techniques to enforce authenticity and integrity over the opportunistic DOTS signal channel. The signalling message specification includes a flag to indicate if the message is signed by the choice of the signature algorithm is let to the users. This signature has to be computed by the DOTS opportunistic-capable client and checked by the DOTS opportunistic-capable gateway or router. Hence, intermediate routers MUST NOT modify the message and its signature except the TTL, which so has not be considered during the signature computation.

Assuming a compromised router, the attacker could nevertheless replay the message or increase the TTL but thanks to the unique policy-id all intermediate-DOTS capable router will drop such messages and thus limiting their forwarding in the network.

Besides, an attacker can also listen opportunistic DOTS signals to monitor the impact of its own attack. These considerations are not specific to the proposed option and supposes that the attacker is able to compromise intermediate routers.

6. IANA Considerations

This draft defines a new IPv6 hop-by-hop option[RFC2460].This requires an IANA RFC3692-style update of:http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml and ultimately the assignment of a new hop-by-hop option according to the guidelines described in [RFC5237].

7. Acknowledgements

This work is partly funded by FLAMINGO, a Network of Excellence Seventh Framework Programme.

8. References

8.1. Normative References

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.ietf-dots-data-channel]
Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel", draft-ietf-dots-data-channel-00 (work in progress), April 2017.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-04 (work in progress), March 2017.

[I-D.ietf-dots-signal-channel]
Reddy, T., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-channel-01 (work in progress), April 2017.

[I-D.krishnan-ipv6-hopbyhop]
Krishnan, S., "The case against Hop-by-Hop options", draft-krishnan-ipv6-hopbyhop-05 (work in progress), October 2010.

8.2. Informative References

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <http://www.rfc-editor.org/info/rfc1157>.

[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>.

[RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <http://www.rfc-editor.org/info/rfc5237>.

[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, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[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>.

[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>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[kuhrer2014exit] M. Kuhrer, T. Hupperich, C. Rossow, T. Holz, "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks", USENIX Security Symposium 23rd, 2014.

[ripe-dnsmon-ddos] RIPE, "NCC DNS Monitoring Service (DNSMON)", <https://atlas.ripe.net/dnsmon/>.

[rootops-ddos] rootops., "Events of 2015-11-30", 2015, <http://root-servers.org/news/events-of-20151130.txt>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Jerome Francois
Inria
615 rue du jardin botanique
Villers-les-Nancy 54600
FR

Phone: +33 3 83 59 30 66
Email: jerome.francois@inria.fr

Abdelkader Lahmadi
University of Lorraine - LORIA
615 rue du jardin botanique
Villers-les-Nancy 54600
FR

Phone: +33 3 83 59 30 00
Email: Abdelkader.Lahmadi@loria.fr

Marco Davids
SIDN Labs
Meander 501
Arnhem 6825 MD
NL

Email: marco.davids@sidn.nl

Giovane Moura
SIDN Labs
Meander 501
Arnhem 6825 MD
NL

Email: marco.davids@sidn.nl

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

Table of Contents

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. incomming 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 subtetlies 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 curently 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:

- Scope
- Stability
- 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:

- Ability of an attacker to correlate network activity
- 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:

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

- TCP/IP stack address filtering
- Application-based address filtering
- 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 progremmers 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 subtetiles 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 genration 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 bond 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 administator 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 communcation 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 incomming 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 coresponding 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.garcia16@gmail.com

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

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

IPv6 maintenance Working Group (6man)
Internet-Draft
Updates: RFC4941 (if approved)
Intended status: Standards Track
Expires: September 14, 2017

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

March 13, 2017

Recommendation on Temporary IPv6 Interface Identifiers
draft-gont-6man-non-stable-iids-01

Abstract

This document clarifies the stability requirements for IPv6 addresses, and provides recommendations regarding the generation of Temporary addresses. Finally, it formally updates RFC4941 such that nodes are allowed to configure only temporary addresses.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

IPv6 StateLess Address AutoConfiguration (SLAAC) [RFC4862] has traditionally resulted in stable addresses, since the Interface Identifier (IID) has been generated by embedding a stable layer-2 numeric identifier (e.g., a MAC address). [RFC4941] implies, throughout the specification, that temporary addresses are generated and employed along with temporary addresses.

While the use of stable addresses (only) or mixed stable and temporary addresses can be desirable in a number of scenarios, there are other scenarios in which, for security and privacy reasons, a node may want to use only Temporary address (e.g., a temporary address).

This document clarifies the requirements for stability of IPv6 addresses, such that nodes are not required to configure stable addresses. It also specifies a set of requirements for the generation of Temporary addresses, and also specifies some sample algorithms that may be employed to generate temporary addresses that comply with the aforementioned requirements. Finally, it formally updates [RFC4941] such that temporary addresses can be employed without the need to configure a stable address along side.

2. Terminology

This document employs the terms defined in [RFC7721].

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

3. Problem statement

When [RFC4941] was written, its authors wanted to prevent privacy and security attacks enabled by addresses that contain "an embedded interface identifier, which remains constant over time". They observed that "Anytime a fixed identifier is used in multiple contexts, it becomes possible to correlate seemingly unrelated activity using this identifier." They were concerned with both on-path attackers who would observe the IP addresses of packets observed in transit, and attackers that would have access to the logs of servers.

Since the publication of [RFC4941] in September 2007, our understanding of threats and mitigations has evolved. The IETF is now officially concerned with Pervasive Monitoring [RFC7258], as well as the wide spread collection of information for advertising and other purposes, for example through the Real Time Bidding protocol used for advertising auctions [RTB25].

3.1. Privacy requirements

The widespread deployment of encryption advocated in [RFC7624] is a response to Pervasive Monitoring. Encryption of communication reduces the amount of information that can be collected by monitoring data links, but does not prevent monitoring of IPv6 addresses embedded in clear text packet headers. Stable IPv6 addresses enable the correlation of such data over time.

MAC Address Randomization [IETFMACRandom] is another response to pervasive monitoring. In conjunction with DHCP Anonymity [RFC7844], it ensures that devices cannot be tracked by their MAC Address or their DHCP identifiers when they connect to "hot spots". However, the privacy effects of MAC Address Randomization would be nullified if a device kept using the same IPv6 address before and after a MAC-address randomization event.

Many Web Browsers have options enabling browsing "in private". However, if the web connections during the private mode use the same IPv6 address as those in the public mode, web tracking systems similar to [RTB25] will quickly find the correlation between the public personna of the user and the supposedly private connection. Similarly, many web browsers have options to "delete history", including deleting "cookies" and other persistent data. Again, if the same IPv6 address is used before and after the deletion of

cookies, web tracking systems will easily correlate the new activity with the prior data collection.

Using temporary address alone may not be sufficient to prevent all forms of tracking. It is however quite clear that some usage of temporary addresses is necessary to provide user privacy. It is also clear that the usage of temporary addresses needs to be synchronized with other privacy defining event such as moving to a new network, performing MAC Address Randomization, or changing the privacy posture of a node.

3.2. Stability Requirements for IPv6 Addresses

Nodes are not required to generate addresses with any specific stability properties. That is, the generation of stable addresses is OPTIONAL. This means that a node may end up configuring only stable addresses, only Temporary, or both stable and temporary addresses.

3.3. Requirements for Temporary IPv6 Addresses

The requirements for temporary IPv6 addresses are as follows:

1. Temporary addresses MUST have a limited lifetime, which should be different for different addresses. The lifetime of an address essentially limits the extent to which network activity correlation can be performed based on such address.

2. The lifetime of an address MUST be further reduced when privacy-meaningful events (such as a node attaching to a new network) takes place.

3. The resulting Interface Identifiers MUST be different when addresses are configured for different prefixes. That is, if different autoconfiguration prefixes are used to configure addresses for the same network interface card, the resulting Interface Identifiers must be (statistically) different. This means that, given two addresses that employ different prefixes, it must be difficult for an outside entity to tell whether the addresses correspond to the same network interface or even whether they have been generated by the same host.

4. The resulting interface identifiers MUST NOT embed layer-2 identifiers (e.g. MAC addresses).

5. It must be difficult for an outside entity to predict the Interface Identifiers that will be generated by the algorithm, even with knowledge of the Interface Identifiers generated for configuring other addresses.

6. The resulting Interface Identifiers MUST be semantically opaque [RFC7136] and MUST NOT follow any specific patterns.

By definition, temporary addresses have a limited lifetime. This is in contrast with e.g. stable addresses [RFC7217], that do not have have a limited lifetime. Having a variable maximum lifetime prevents an observer from synchronizing with the temporary address regeneration; that is, from being able to expect when address will be regenerated, and thus infer that one newly observed addresses is the result of regenerating a previously observed one.

The lifetime of an address should be further reduced by privacy-meaningful events. For example, a host should not employ the same address across network attachment events. That is, a host that de-attaches from a network and subsequently re-attaches to a (possibly different) network should regenerate all of its temporary addresses. Similarly, a host that implements MAC address randomization should regenerate all of its temporary addresses. Other events, such as those discussed in Section 3.1 should also trigger the regeneration of all temporary addresses.

The IIDs of addresses configured for different autoconfiguration prefixes must be different, such that traffic for those addresses cannot be correlated.

The reuse of identifiers that have their own semantics or properties across different contexts or scopes can be detrimental for security and privacy [I-D.gont-predictable-numeric-ids] [RFC6973] [RFC4941]. For example, if two different layer-3 protocols generate their addresses by embedding a layer-2 identifier (e.g., a MAC address), then the traffic for such protocols could be correlated (irrespective of whether the aforementioned layer-2 identifier has been randomized or not). Besides, a node that generates an IPv6 address by embedding a link-layer address in the IPv6 address will, when configuring addresses for different prefixes, result in the same IID being used for such prefixes, thus allowing the corresponding traffic to be correlated.

For security and privacy reasons, the IIDs generated for temporary addresses must not be predictable. Otherwise, the node may be subject to many (if not all) of the security and privacy issues that are meant to be mitigated (please see [RFC7721].

Any semantics or patterns in an IID might be leveraged by an attacker to e.g. reduce the search space when performing address-scanning attacks, infer the identity of the node, etc.

4. Generation of Temporary IPv6 Addresses

The following subsections specify algorithms that may be employed to generate temporary addresses that comply with the requirements specified in Section 3.3.

4.1. RFC 4941

One possible algorithm for generating temporary IPv6 addresses is that specified in [RFC4941].

We note that, since the publication of [RFC4941], a number of issues have been found in common implementations of such algorithm [RAID2015].

> TODO: It remains an open question what to do with respect to RFC4941. If this draft was to obsolete RFC4941, instead of merely update it, we would need to include here the actual specification of the address generation algorithm.

4.2. Randomized IIDs

Another possible approach would be to select a random IID when performing SLAAC. A node employing this algorithm should generate IIDs as follows:

1. Obtain a random number (see [RFC4086] for randomness requirements for security)

2. The Interface Identifier is obtained by taking as many bits from the aforementioned random number (obtained in the previous step) as necessary.

   > We note that [RFC4291] requires that the Interface IDs of all unicast addresses (except those that start with the binary value 000) be 64 bits long. However, the method discussed in this document could be employed for generating Interface IDs of any arbitrary length, albeit at the expense of reduced entropy (when employing Interface IDs smaller than 64 bits).

   The resulting Interface Identifier SHOULD be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, this situation SHOULD be handled in the same way as the case of duplicate addresses.

A node that disconnects from the network and subsequently reconnects would employ a (statistically different) IID for the same prefix. Similarly, a different (random) IID should be employed for each autoconfiguration prefix. In the event of Duplicate Address Detection (DAD) [RFC4862] failures, another random number should be selected to recover from the DAD failure.

4.3. Hash-based generation of temporary address generation

The algorithm in [RFC7217] can be augmented for the generation of temporary addresses. The benefit of this would be that a node could employ a single algorithm for generating stable and temporary addresses, by employing appropriate parameters.

Nodes would employ the following algorithm for generating the temporary IID:

1. Compute a random identifier with the expression:

   RID = F(Prefix, MAC_Address, Network_ID, Time, DAD_Counter, secret_key)

   Where:

   RID:
   Random Identifier

   F():
   A pseudorandom function (PRF) that MUST NOT be computable from the outside (without knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret_key, even when given samples of the output of F() and knowledge or control of the other input parameters. F() SHOULD produce an output of at least 64 bits. F() could be implemented as a cryptographic hash of the concatenation of each of the function parameters. SHA-1 [FIPS-SHS] and SHA-256 are two possible options for F(). Note: MD5 [RFC1321] is considered unacceptable for F() [RFC6151].

   Prefix:
   The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message, or the link-local IPv6 unicast prefix [RFC4291].

   MAC_Address:
   The MAC address corresponding to the underlying network interface card. Employing the MAC address in this expression (in replacement of the Net_Iface parameter of the expression

in RFC7217) means that the re-generation of a randomized MAC address will result in a different temporary address.

Network_ID:
Some network-specific data that identifies the subnet to which this interface is attached -- for example, the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. Additionally, Simple DNA [RFC6059] describes ideas that could be leveraged to generate a Network_ID parameter. This parameter is SHOULD be employed if some form of "Network_ID" is available.

Time:
An implementation-dependent representation of time. One possible example is the representation in UNIX-like systems [OPEN-GROUP], that measure time in terms of the number of seconds elapsed since the Epoch (00:00:00 Coordinated Universal Time (UTC), 1 January 1970).

DAD_Counter:
A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts.

secret_key:
A secret key that is not known by the attacker. The secret key SHOULD be of at least 128 bits. It MUST be initialized to a pseudo-random number (see [RFC4086] for randomness requirements for security) when the operating system is installed or when the IPv6 protocol stack is "bootstrapped" for the first time.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit. The resulting Interface Identifier SHOULD be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, this situation SHOULD be handled in the same way as the case of duplicate addresses.

5. Update to existing RFCs

The following subsections clarify how each of the RFCs affected by this document are updated.

5.1. Update to RFC4941

The temporary addresses specified in [RFC4941] MAY be used in replacement of the stable addresses [RFC8064]. That is, a node MAY configure temporary addresses only, without configuring any stable addresses.

6. Future Work

This document clarifies the requirements for stability requirements for IPv6 addresses, and specifies requirements for temporary addresses. A separate document ([I-D.gont-6man-address-usage-recommendations]) discusses the tradeoffs involved when considering different stability properties of IPv6 addresses, and and the different configuration setups such as: stable addresses only, temporary addresses only, or mixed stable and temporary addresses.

7. 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.

8. Security Considerations

This document clarifies the stability requirements for IPv6 addresses, and specifies requirements for the generation of temporary addresses. Additionally, it formally updates [RFC4941] such that stable addresses are not required to be configured along with the temporary addresses.

The security and privacy properties of IPv6 addresses have been discussed in detail in [RFC7721] and [RFC7707].

9. Acknowledgements

The authors would like to thank (in alphabetical order) Brian Carpenter and Lorenzo Colitti for providing valuable feedback on earlier versions of this document.

10. References

10.1. Normative References

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[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>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <http://www.rfc-editor.org/info/rfc5453>.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <http://www.rfc-editor.org/info/rfc7136>.

[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>.

[RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <http://www.rfc-editor.org/info/rfc8064>.

10.2. Informative References

[FIPS-SHS]
NIST, "Secure Hash Standard (SHS)", FIPS Publication 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[I-D.gont-6man-address-usage-recommendations]
Gont, F., Gont, G., and M. Corbo, "IPv6 Address Usage Recommendations", draft-gont-6man-address-usage-recommendations-01 (work in progress), February 2017.

[I-D.gont-predictable-numeric-ids]
Gont, F. and I. Arce, "Security and Privacy Implications of Numeric Identifiers Employed in Network Protocols", draft-gont-predictable-numeric-ids-00 (work in progress), February 2016.

[IANA-RESERVED-IID]
IANA, "Reserved IPv6 Interface Identifiers", <http://www.iana.org/assignments/ipv6-interface-ids>.

[IETFMACRandom]
Zuniga, JC., "MAC Privacy", November 2014, <http://www.ietf.org/blog/2014/11/mac-privacy/>.

[OPEN-GROUP]
The Open Group, "The Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008, 2016 Edition", Section 4.16 Seconds Since the Epoch, 2016, <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/contents.html>.

[RAID2015]
Ullrich, J. and E. Weippl, "Privacy is Not an Option: Attacking the IPv6 Privacy Extension", International Symposium on Recent Advances in Intrusion Detection (RAID), 2015, <https://www.sba-research.org/wp-content/uploads/publications/Ullrich2015Privacy.pdf>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <http://www.rfc-editor.org/info/rfc1321>.

[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>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <http://www.rfc-editor.org/info/rfc6151>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <http://www.rfc-editor.org/info/rfc7624>.

[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>.

[RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <http://www.rfc-editor.org/info/rfc7844>.

[RTB25] Interactive Advertising Bureau (IAB), "Real Time Bidding (RTB) project, OpenRTB API Specification Version 2.5", December 2016, <http://www.iab.com/wp-content/uploads/2016/03/OpenRTB-API-Specification-Version-2-5-FINAL.pdf>.

Authors’ Addresses

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

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

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

Email: huitema@huitema.net
URI: http://privateoctopus.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.garcia16@gmail.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

Table of Contents

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 depending 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 transfered.

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:

- Clarified Section 1 "Introduction" that the purpose of PMTUD is to reduce the need for IPv6 fragmentation.
- Added text to Section 1 "Introduction" about the effects on PMTUD when ICMPv6 messages are blocked.
- 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.
- Added a short summary to the Section 1 "Introduction" of Packetization Layer Path MTU Discovery ((PLPMTUD) and a reference to RFC4821 that defines it.
- Aligned text in Section 2 "Terminology" to match current packetization layer terminology.
- 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.
- 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].
- 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.

- Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.

- Removed text in Section 5.2 "Storing PMTU information" about the RH0 routing header because it was deprecated by RFC5095.

- Removed text about obsolete security classification from Section 5.2 "Storing PMTU information".

- Changed title of Section 5.4 to "Packetization Layer actions" and changed to text in the first paragraph to to generalize this section to cover all packetization layers, not just TCP.

- Clarified text in Section 5.4 "Packetization Layer actions" to use normal packetization layer retransmission methods.

- Removed text in Section 5.4 "Packetization Layer actions" that described 4.2 BSD because it is obsolete, and removed reference to TP4.

- Updated text in Section 5.5 "Issues for other transport protocols" about NFS including adding a current reference to NFS and removing obsolete text.

- Added paragraph to Section 6 "Security Considerations" about black hole connections if PTB messages are not received, and comparison to PLPMTD.

- Updated Section 7 "Acknowledgements".

- 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:

- 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.
- Added references for EMTU_S and EMTU_R.
- Added clarification to Section 4 "Protocol Requirements" that nodes should detect decreases in PMTU as fast as possible.
- Added clarification Section 5.2 "Storing PMTU information" that nodes with multiple interface, Path MTU information should be stored for each link.
- Removed text in Section 5.2 about Retransmission because it was unneeded.
- Removed text in Section 5.3 about Retransmission because it was unneeded.
- Rewrote text in Section 5.4 "Packetization Layer actions" regarding reception to make it clearer.
- Rewrote the text at the end of Section 5.4 to remove unnecessary details and clarify not change congestion window.
- 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

Table of Contents

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:

- 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.

- 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.

- 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.

- 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.

- 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].

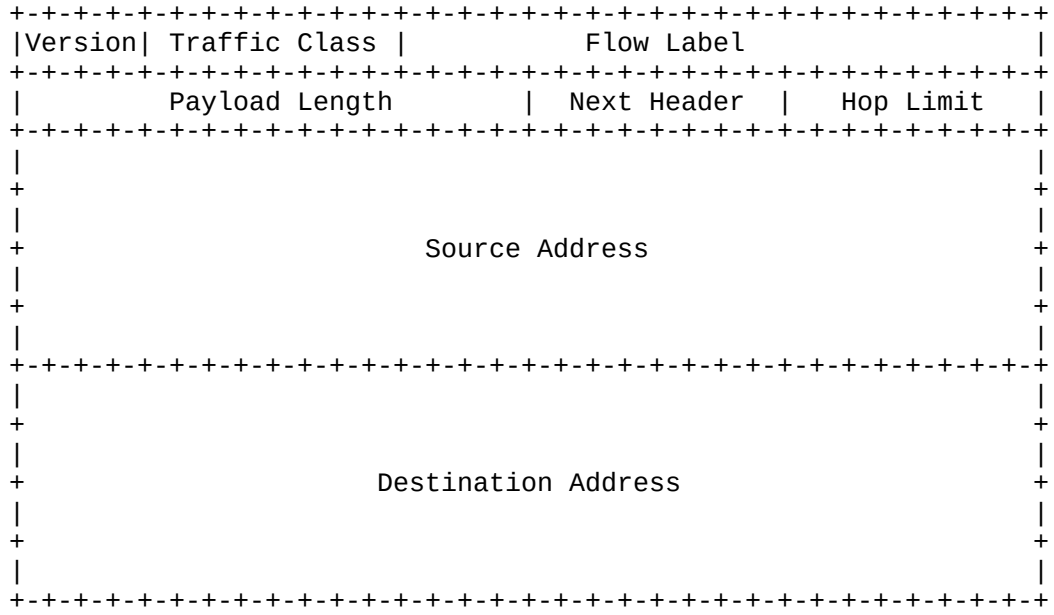
Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

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.)

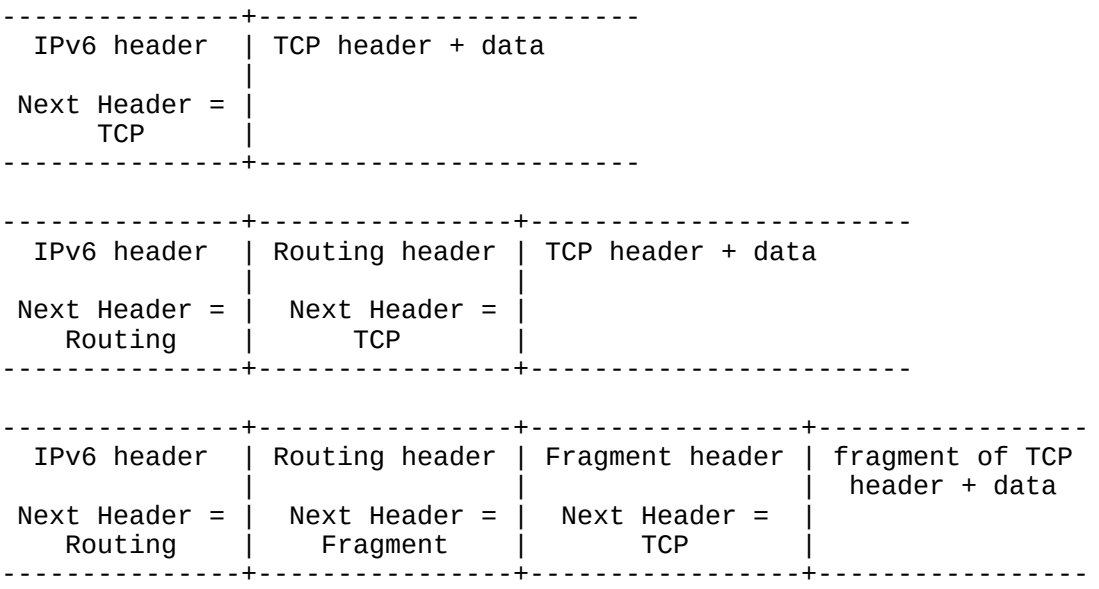
Next Header	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hop Limit	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. When forwarding, the packet is discarded if Hop Limit was 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, 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).

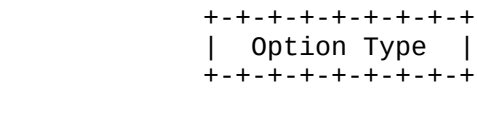
If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

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.

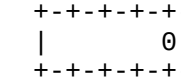
The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers.

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:

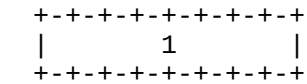
Pad1 option (alignment requirement: none)



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)



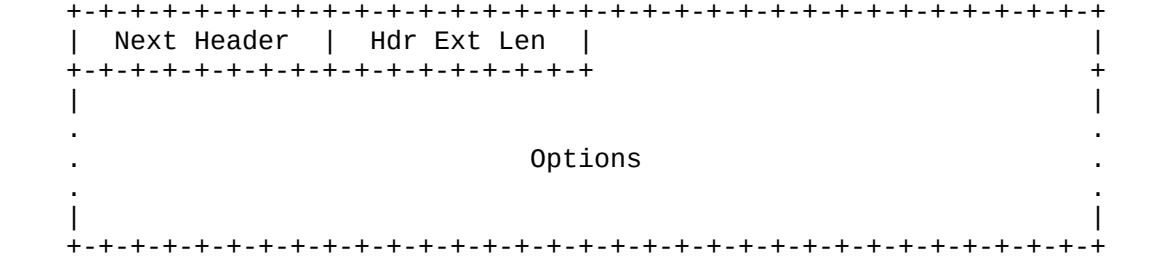
The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the

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:



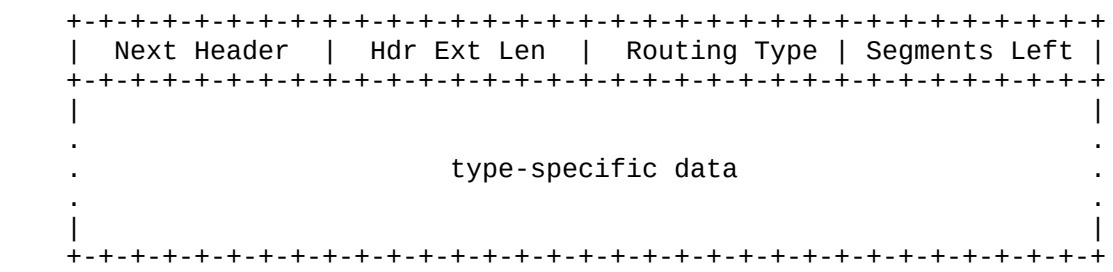
Next Header	8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Options	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.

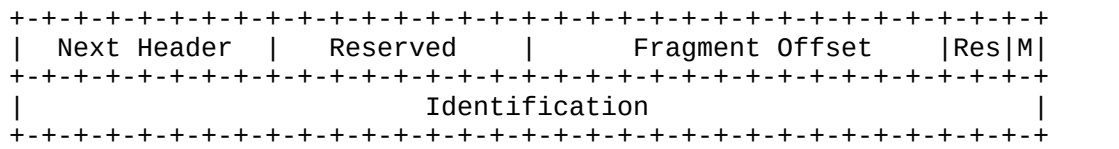
If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

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.

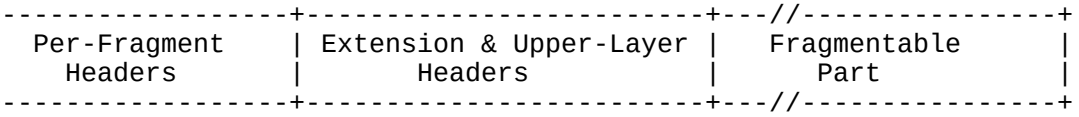
In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

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.

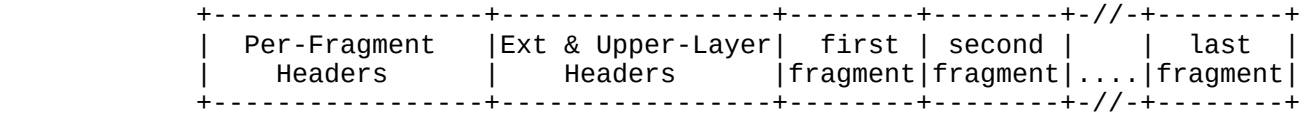
Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted ESP.

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.

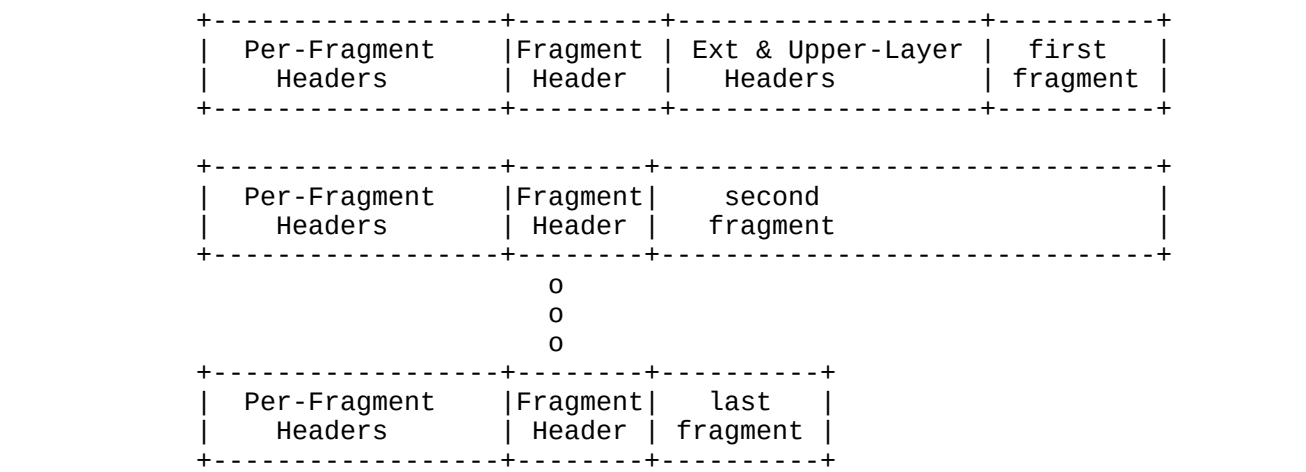
The Fragmentable Part of the original packet is divided into fragments. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s). Each complete fragment, except possibly the last ("rightmost") one, being an integer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

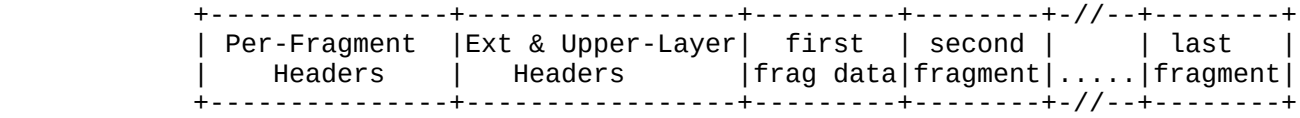
The Identification value generated for the original packet.

(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:

- 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.

- 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.

- 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.

- 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.

- If any of the fragments being reassembled overlaps with any other fragments being reassembled for the same packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded and no ICMP error messages should be sent.

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

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

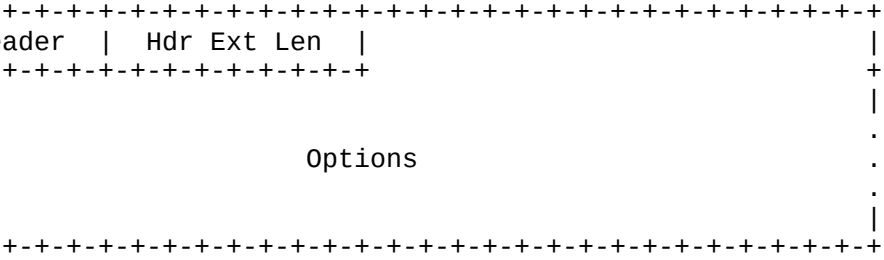
The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly.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

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

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

4.6. Destination Options Header

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



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

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

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

- 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.

- 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

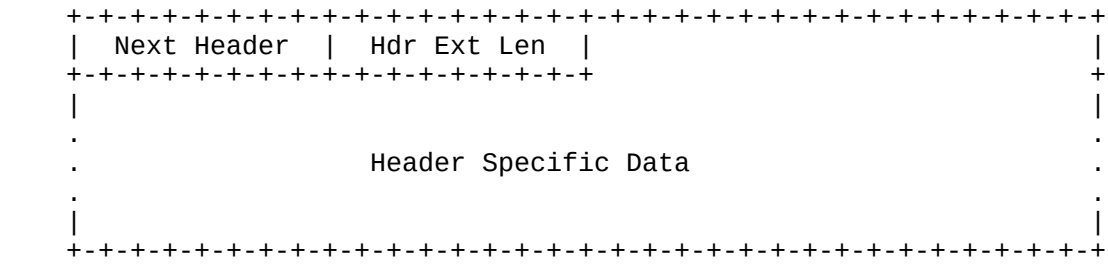
Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6 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

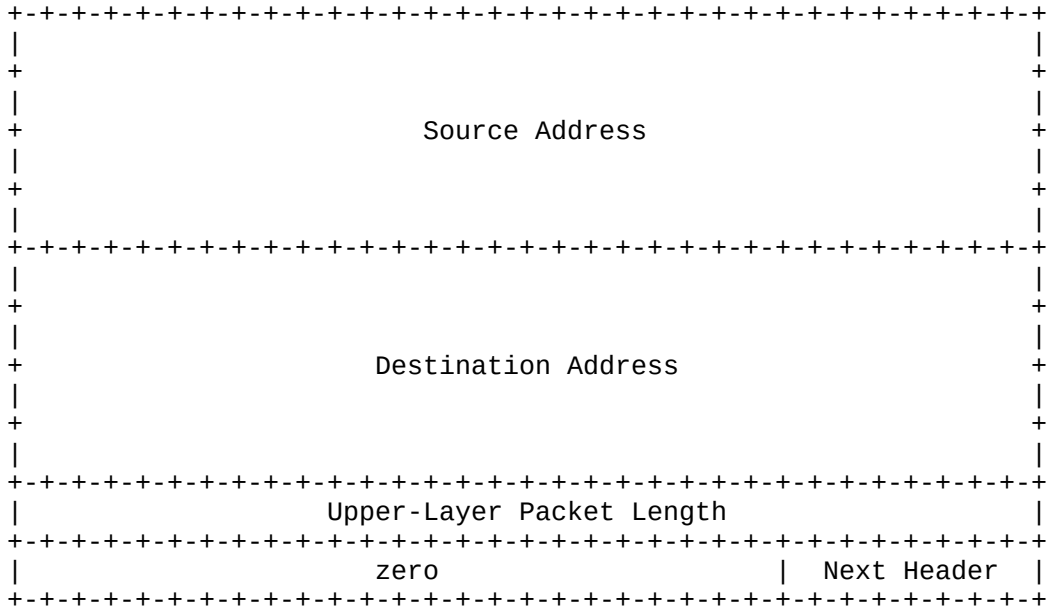
The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet or fragment might be different from the value sent by the packet's source.

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:



- 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.

- 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.

- 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:

- Response packets that do not carry Routing headers.
- 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).
- 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:

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

- Assigned Internet Protocol Numbers [IANA-PN]
- ONC RPC Network Identifiers (netids) [IANA-NI]
- Technical requirements for authoritative name servers [IANA-NS]
- Network Layer Protocol Identifiers (NLPIDs) of Interest [IANA-NL]
- Protocol Registries [IANA-PR]
- 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:

- Eavesdropping, On-path elements can observe the whole packet (including both contents and metadata) of each IPv6 datagram.
- Replay, where attacker records a sequence of packets off of the wire and plays them back to the party which originally received them.
- Packet insertion, where the attacker forges a packet with some chosen set of properties and injects it into the network.
- Packet deletion, where the attacker remove a packet from the wire.
- Packet modification, where the attacker removes a packet from the wire, modifies it, and re-injects it into the network.
- 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.
- 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 throughly, 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:

- 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.

- 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:

- 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.

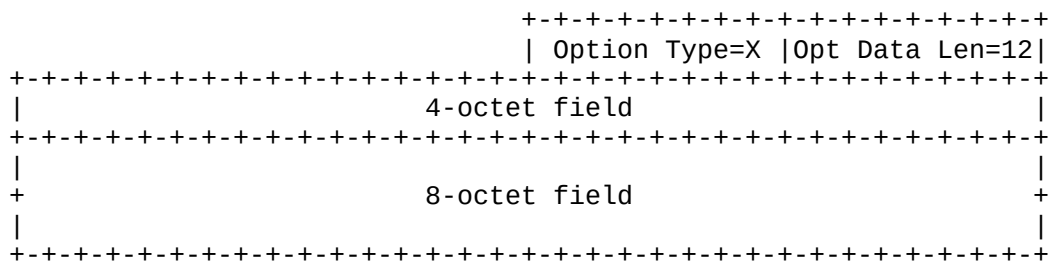
- o Another desirable feature is that the Hop-by-Hop or Destination Options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.

- 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.

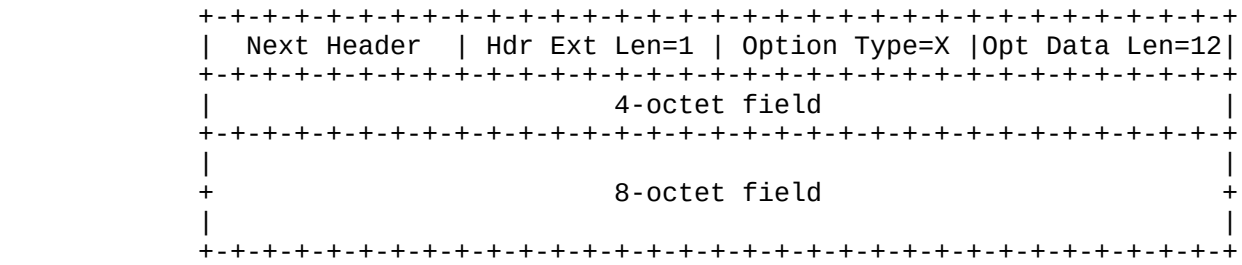
These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of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

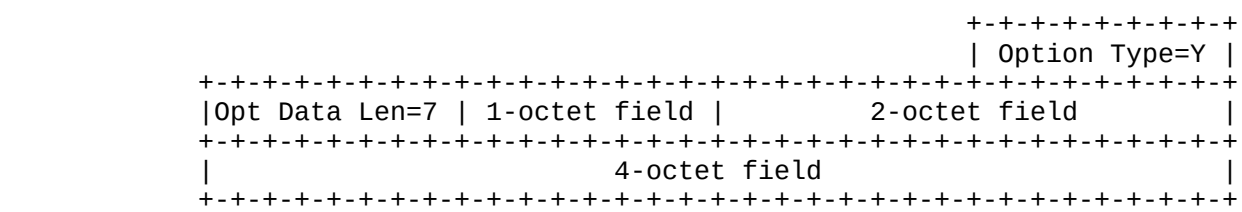


Its alignment requirement is 8n+2, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

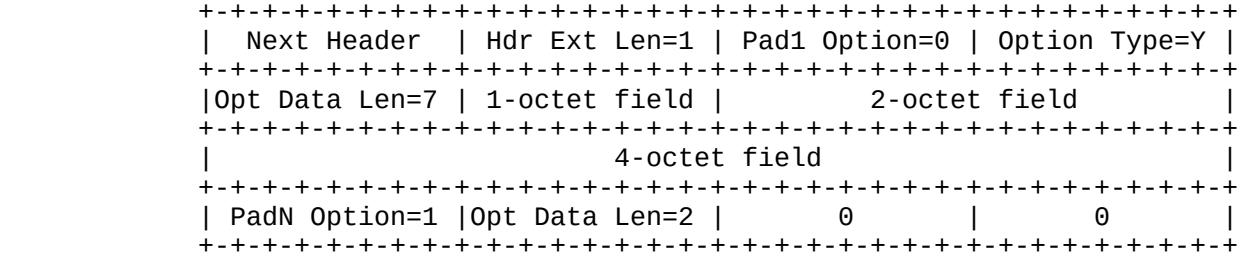


Example 2

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:

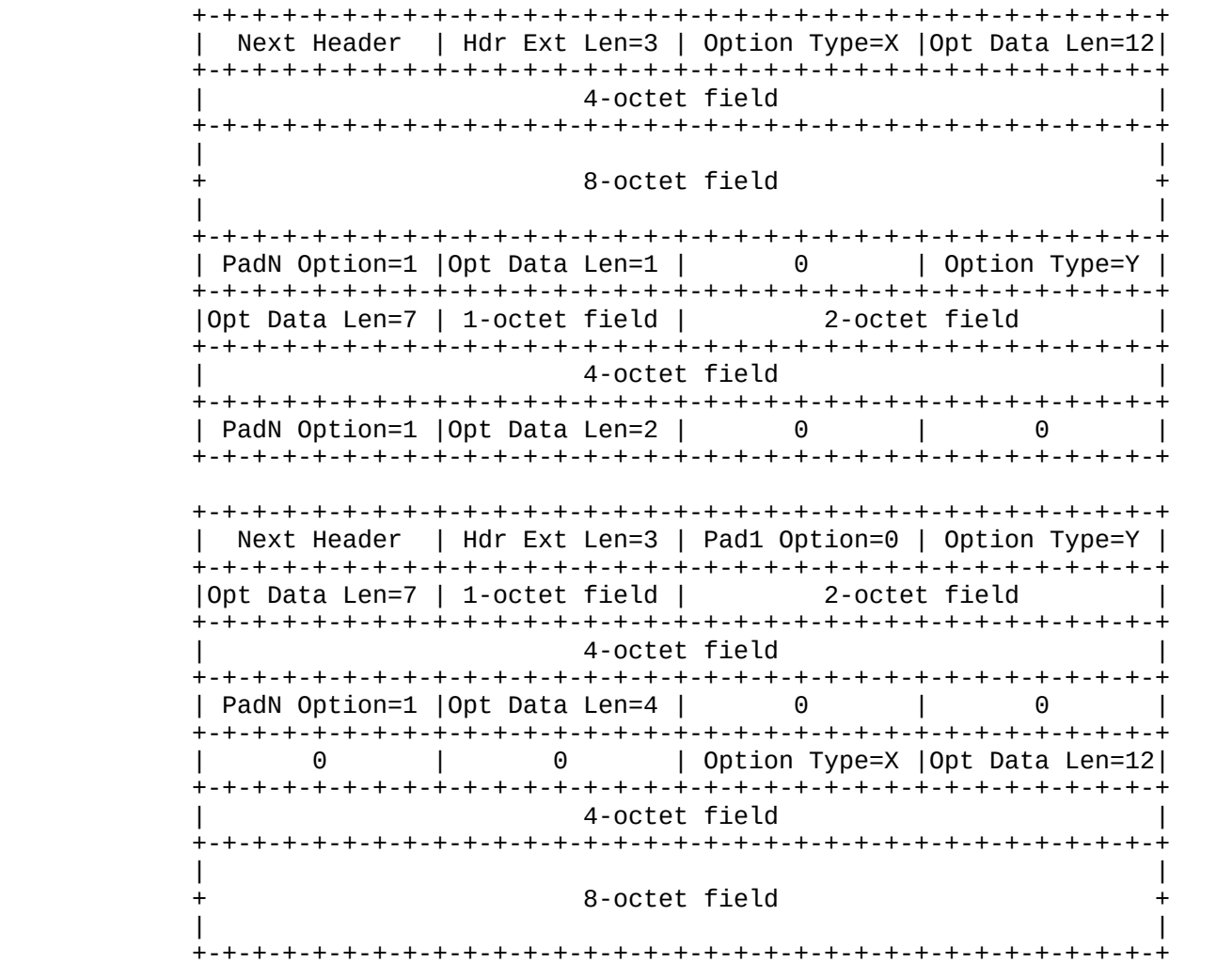


Its alignment requirement is 4n+3, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:



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.

- Removed IP Next Generation from the Abstract.
- Added text in Section 1 that the Data Transmission Order is the same as IPv4 as defined in RFC791.
- 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

Table of Contents

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 it's 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:cd30:0000:0000:0000:0000/60

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].

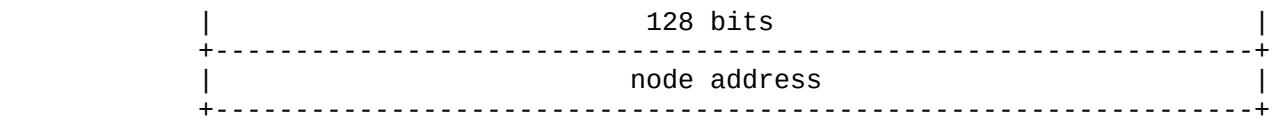
2.4. Unicast Addresses

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

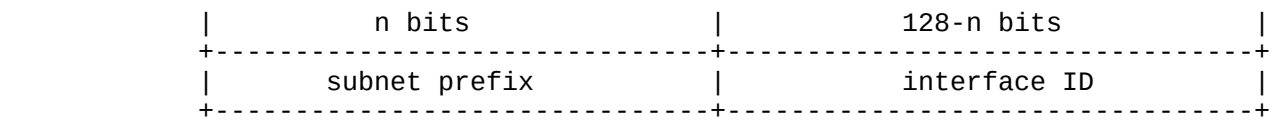
IPv6 unicast routing is based on prefixes of any valid length up to 128 [BCP198].

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 call 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.

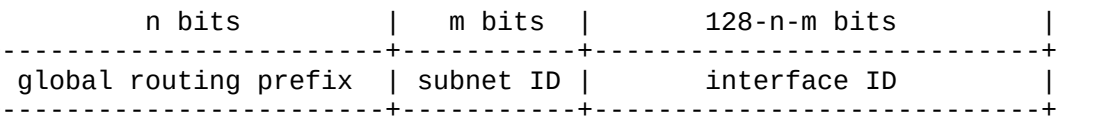
2.4.3. The Loopback Address

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.

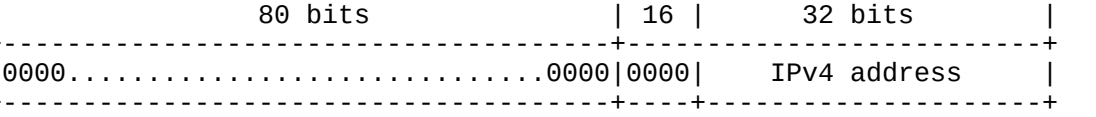
Examples of Global Unicast addresses that start with binary 000 are the IPv6 address with embedded IPv4 addresses described in Section 2.4.5. An example of global addresses starting with a binary value other than 000 (and therefore having a 64-bit interface ID field) can be found in [RFC3587].

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:

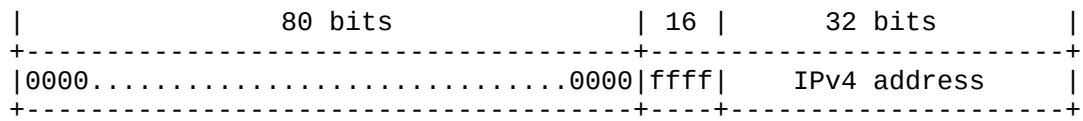


Note: The IPv4 address used in the "IPv4-Compatible IPv6 address" must be a globally-unique IPv4 unicast address.

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

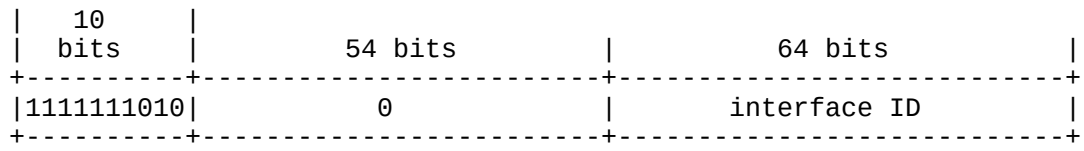
A second type of IPv6 address that holds an embedded IPv4 address is defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. The format of the "IPv4-mapped IPv6 address" is as follows:



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.

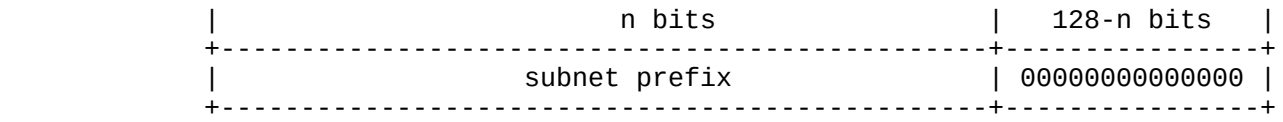
Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire Internet, which presents a severe scaling limit on 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:



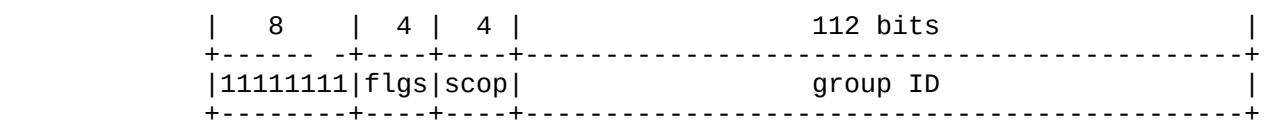
The "subnet prefix" in an anycast address is the prefix that identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero.

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.

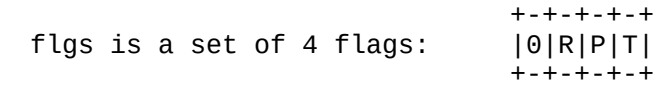
The Subnet-Router anycast address is intended to be used for applications where a node needs to communicate with any one of the set of routers.

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.



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:

- Its required Link-Local address for each interface.

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

- The Subnet-Router Anycast addresses for all interfaces for which it is configured to act as a router.
- All other Anycast addresses with which the router has been configured.
- 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:

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

- Number Resources [IANA-NR]
- Protocol Registries [IANA-PR]
- Technical requirements for authoritative name servers [IANA-NS]
- 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 relavant 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., Beebee, 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>.

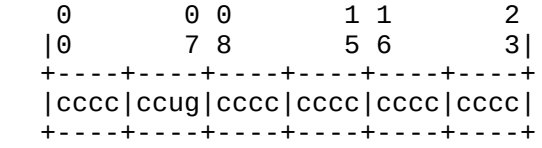
[RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <http://www.rfc-editor.org/info/rfc7428>.

[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:



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

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

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

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

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

Links or Nodes with IEEE EUI-64 Identifiers

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

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc0gccccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

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

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc1gccccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

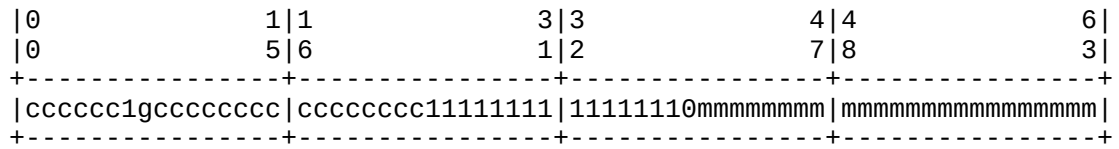
The only change is inverting the value of the universal/local bit.

Links or Nodes with IEEE 802 48-bit MACs

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

```
|0              1|1              3|3              4|
|0              5|6              1|2              7|
+----------------+----------------+----------------+
|cccccc0gccccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+
```

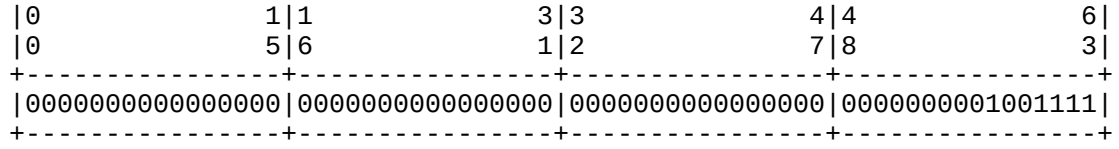
where "c" is the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate Global scope, "g" is individual/ group bit, and "m" is the bits of the manufacturer- selected extension identifier. The interface identifier would be of the form:



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 0xFF 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":

- 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.

- Added text to the last paragraph in Section 2.1 to clarify the differences on how subnets are hangled in IPv4 and IPv6, includes

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

- 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.

- 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.

- 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.

- 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.

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

- 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.

- 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.

- Added a reference to the IANA Multicast address registry in Section 2.6.1.

- Added instructions in IANA Considerations to update references in the IANA registries that currently point to RFC4291 to point to this document.

- 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.

- Add note to Section 5 section acknowledging the authors of the updating documents.

- 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.

- 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 hangled 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

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.

Table of Contents

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.

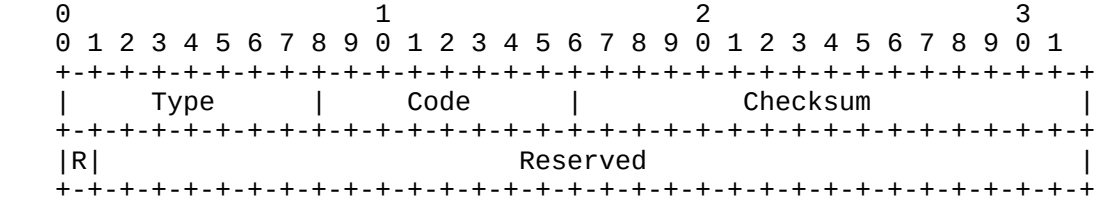
The host calculates a timeout after it has received a RTO - either per router or per link. If it is maintained per link then the host SHOULD use the minimum Refresh Time it has received from the routers on the link. The timeout is a random value uniformly distributed between 0.5 and 1.5 times the Refresh 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.



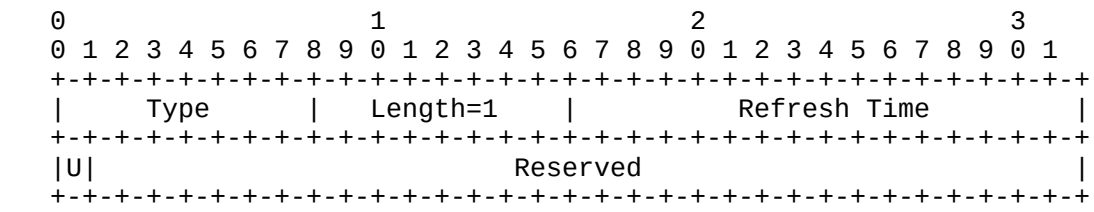
New fields:

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 resilent 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:

- Removed any suggestion that periodic RAs would not be needed. The remain required.

- Made Refresh Time zero be reserved and RTOs with this value ignored by the receiver.
- Removed notion that all-ones refresh time means infinite lifetime. It now means 65535 seconds.
- Changed default to be multicast RS refresh, with the option to specify unicast in the RTO. This enables discovering new routers on the link.
- Clarified the normative behavior for hosts that sleep on a schedule.
- Clarified the updated DNA behavior.
- 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

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

Table of Contents

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

- DHCPv6 Prefix Delegation to hosts ([3]), and
- 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).

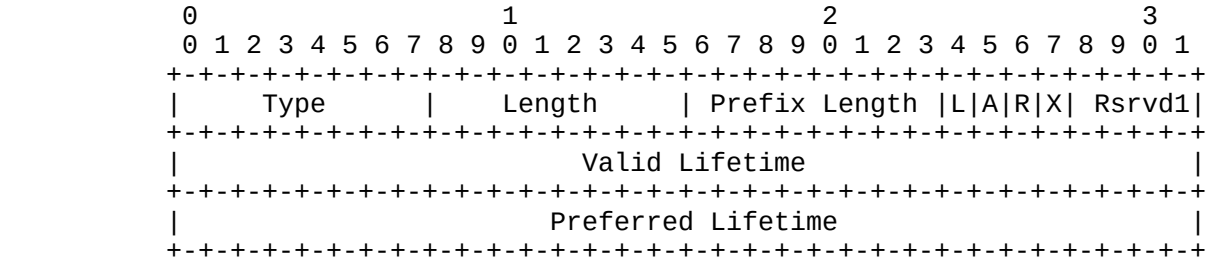
4.2.3. Host

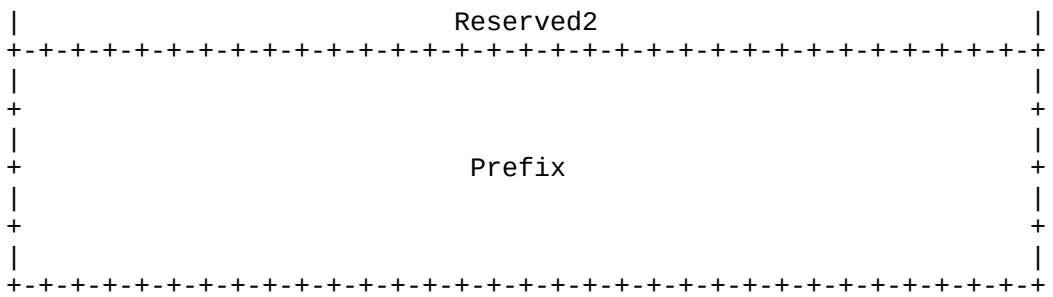
The term "host" may be used interchangeably throughout this document to mean a network node receiving and processing an RA. The receiving node may itself be a router, or may temporarily become one by routing all or a portion of an exclusive use prefix.

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





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:

- interpret the L flag as if it were 0 (L=0)
- interpret the A flag as it 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 (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

Table of Contents

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:

- 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.

- 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.

- 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.

- 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:

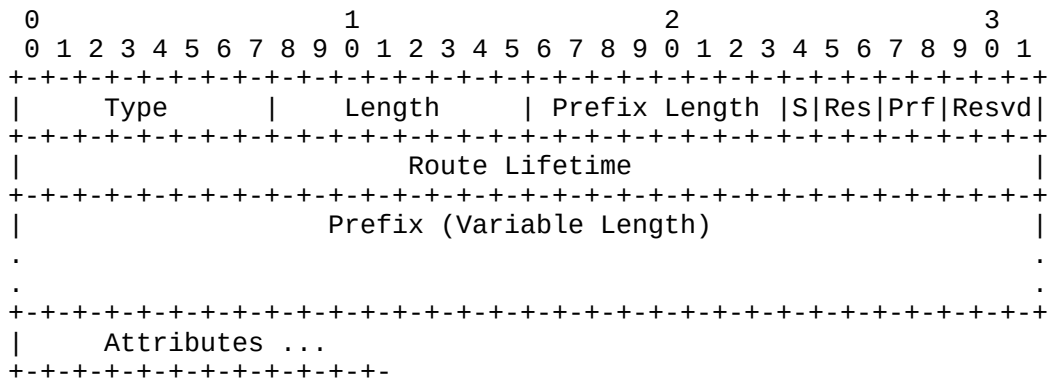


Figure 1: Updated RIO Format

This format introduces a new S flag and variable-length Attributes. The fields of the main body of the RIO are set as follows:

- Type, Prefix Length, Prf, Route Lifetime and Prefix are set exactly as specified in Section 2.3 of [RFC4191].

- For RA messages, Length is set exactly as specified in Section 2.3 of [RFC4191] and no Attributes are included. For all other IPv6 ND messages, Length MUST be initialized to exactly 1 when Prefix Length is 0, to exactly 2 when Prefix Length is between 1 and 64, and to exactly 3 when Prefix Length is greater than 64. Length is then incremented by the length of all included Attributes in units of 8-octets (see below).

- S is set to ’1’ to "Solicit" route information or to ’0’ (i.e., the default value) to "Assert" route information.

- Res and Resvd are reserved and MUST be set to ’0’.

Attributes MAY be included as ancillary route information. Each Attribute is formatted in the same manner as specified for IPv6 ND options in Section 4.6 of [RFC4861] and as shown in Figure 2:

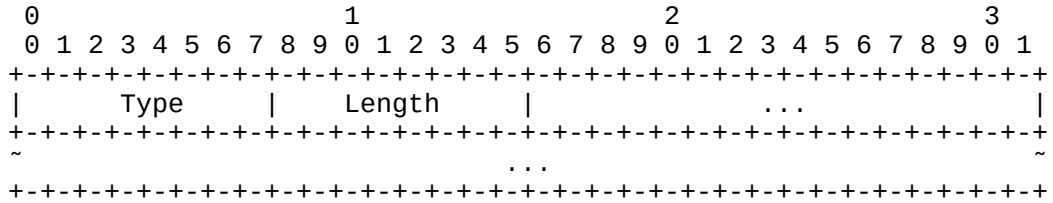


Figure 2: RIO Attribute Format

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:

- 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.

- NS and RS messages MAY include some RIOs with S set to ’1’ and others with S set to ’0’.

- 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.)

- 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:

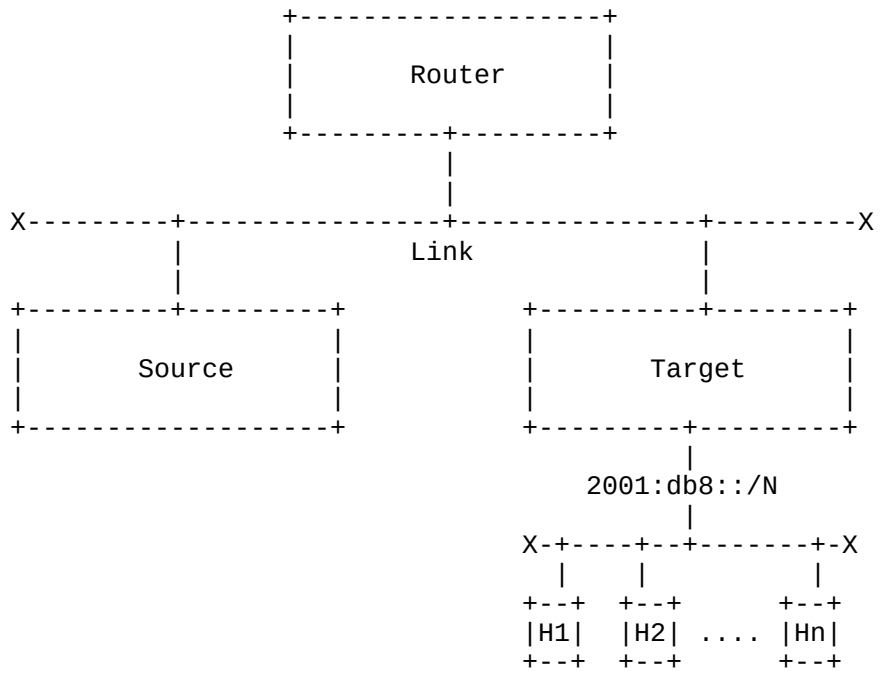


Figure 3: Classical Redirection Scenario

In addition, the Target may be a router that connects an arbitrarily-complex set of IPv6 networks (e.g., as depicted by 2001:db8::/N in the figure) with hosts H(i).

In this scenario, the Source initially has no route for 2001:db8::/N and must send initial packets destined to correspondents H(i) via a first-hop Router. Upon receiving the packets, the Router forwards the packets to the Target and may also send a Redirect message back to the Source with the Destination Address field set to the destination of the packet that triggered the Redirect, the Target Address field set to the target link-local address and with a Target Link Layer Address Option (TLLAO) that includes the target link-layer address. After receiving the message, the Source may begin sending packets destined to H(i) directly to the Target, which will then forward them to its connected networks.

This specification augments the classical Redirection scenario by allowing the Router to include entire prefixes (e.g., 2001:db8::/N) in RIOs in the Redirect message, and thereafter allowing the Source to include RIOs in an NS message and the Target to include RIOs in its NA response. The following sections present this "augmented" RIO redirection scenario.

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:

- Removed "Ver" field and version numbers.
- Included reference to ’draft-templin-v6ops-pdhost’
- Changed "MAY" to "may" in two places
- Added text on advertising interfaces

- 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