

6man	B. Carpenter
Internet-Draft	Univ. of Auckland
Updates: 2460, 2780 (if approved)	S. Jiang
Intended status: Standards Track	Huawei Technologies Co., Ltd
Expires: February 15, 2013	August 14, 2012

Transmission of IPv6 Extension Headers  
draft-carpenter-6man-ext-transmit-00

Abstract

Various IPv6 extension headers have been defined since the IPv6 standard was first published. This document updates RFC 2460 to describe how intermediate nodes should deal with such extension headers and with any that are defined in future. It also specifies how extension headers should be registered by IANA, with a corresponding minor update to RFC 2780.

Status of this Memo

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

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

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

This Internet-Draft will expire on February 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Requirement to Transmit Extension Headers . . . . .	4
3. Security Considerations . . . . .	5
4. IANA Considerations . . . . .	5
5. Acknowledgements . . . . .	5
6. Change log [RFC Editor: Please remove] . . . . .	6
7. References . . . . .	6
7.1. Normative References . . . . .	6
7.2. Informative References . . . . .	6
Authors' Addresses . . . . .	7

## 1. Introduction

An initial set of IPv6 extension headers was defined by [RFC2460], which also described how they should be handled by intermediate nodes, with the exception of the hop-by-hop options header:

"...extension headers are not 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."

This provision allowed for the addition of new extension headers, since it means that forwarding nodes should be completely transparent to them. Thus, new extension headers could be introduced progressively, used only by hosts that have been updated to create and interpret them. Several such extension headers have been defined since RFC 2460.

Unfortunately, experience has showed that the network is not transparent to these headers. The main reason for this is that some firewalls attempt to inspect the transport payload. This means that they traverse the chain of extension headers, if present, until they find the payload. If they encounter an unknown extension header type, some of these firewalls treat the packet as suspect and drop it. It is an established fact that several widely used firewalls do not recognise some or all of the extension headers defined since RFC 2460. It has also been observed that a few firewalls do not even recognise all the extension headers in RFC 2460, including the fragment header, causing fundamental problems of connectivity.

Other types of middlebox, such as load balancers or packet classifiers, might also fail in the presence of extension headers that they do not recognise.

A contributory factor to this problem is that, because extension headers are numbered out of the existing IP Protocol Number space, there is no collected list of them. For this reason, it is hard for an implementor to quickly identify the full set of valid extension headers. An implementor who consults only RFC 2460 will miss all extension headers defined subsequently.

[RFC6564] improves the situation by defining a uniform format for any future extension headers, but this in itself is insufficient. The present document clarifies that the above requirement from RFC 2460 applies to all types of node that forward IPv6 packets and to all extension headers defined now and in the future. It also requests IANA to create a subsidiary registry that clearly identifies valid extension header types, and updates RFC 2780 accordingly.

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

## 2. Requirement to Transmit Extension Headers

[NOTE IN DRAFT: These requirements may look controversial compared to RFC 2460, and are presented for discussion in the WG. In reality, hop-by-hop options are not handled by many high speed routers, or are processed only on a slow path. Also, many firewalls inspect and filter extension headers. The following text is intended to deal with those realities.]

The IPv6 Hop-by-Hop Options header SHOULD be processed by intermediate nodes as described in [RFC2460]. However, it is to be expected that some high performance routers will either ignore it, or assign packets containing it to a slow processing path. Designers planning to use a Hop-by-Hop option should be aware of this likely behaviour.

Apart from that, any node along an IPv6 packet's path, which forwards it for any reason, SHOULD do so regardless of any extension headers that are present, as described in RFC 2460. Exceptionally, if this node is designed to examine extension headers for any reason, such as firewalling, it MUST recognise and deal appropriately with all valid IPv6 extension header types. The list of currently valid extension header types is maintained by IANA (see Section 4).

RFC 2460 requires destination hosts to discard packets containing unrecognised extension headers. However, intermediate forwarding nodes MUST NOT do this by default, since that might cause them to inadvertently discard traffic using a recently defined extension header, not yet recognised by the intermediate node.

As mentioned above, firewalls that violate RFC 2460 by discarding packets containing valid extension headers are known to cause connectivity failures. Therefore, it is important that firewalls behave according to the above requirements. If a firewall chooses to discard a packet containing a valid IPv6 extension header, it MUST be the result of an explicit firewall policy, and not just the result of a failure to recognise such a header.

The IPv6 Routing Header Types 0 and 1 have been deprecated and SHOULD NOT be used. However, as specified in [RFC5095], this does not mean that the IPv6 Routing Header can be unconditionally dropped by forwarding nodes. Packets containing undeprecated Routing Headers MUST be forwarded by default.

### 3. Security Considerations

Firewall devices MUST conform to the requirements in the previous section. In particular, packets containing specific extension headers are only to be discarded as a result of explicit policy, and never by default.

### 4. IANA Considerations

IANA is requested to replace the existing empty IPv6 Next Header Types registry by an IPv6 Extension Header Types registry, clearly marked as subsidiary to the existing Assigned Internet Protocol Numbers registry. It will contain only those protocol numbers which are also IPv6 Extension Header types. The initial list will be as follows:

- o 0, Hop-by-Hop Options, [RFC2460]
- o 43, Routing, [RFC2460], [RFC5095]
- o 44, Fragment, [RFC2460]
- o 50, Encapsulating Security Payload, [RFC4303]
- o 51, Authentication, [RFC4302]
- o 58, ICMPv6, [RFC2460]
- o 59, No Next Header, [RFC2460]
- o 60, Destination Options, [RFC2460]
- o 135, MIPv6, [RFC6275]
- o 139, HIP, [RFC5201]
- o 140, shim6, [RFC5533]

Any future IPv6 Extension Header types will be added to this registry as well as to the Assigned Internet Protocol Numbers registry.

The reference to the IPv6 Next Header field in [RFC2780] applies equally to the IPv6 Extension Header field.

### 5. Acknowledgements

This document was triggered by mailing list discussions including John Leslie, Stefan Marksteiner and others. Valuable comments and contributions were made by TBD and others.

Brian Carpenter was a visitor at the Computer Laboratory, Cambridge University during part of this work.

This document was produced using the xml2rfc tool [RFC2629].

## 6. Change log [RFC Editor: Please remove]

draft-carpenter-6man-ext-transmission-00: original version,  
2012-08-14.

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, December 2007.
- [RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.
- [RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

### 7.2. Informative References

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, April 2012.

Authors' Addresses

Brian Carpenter  
Department of Computer Science  
University of Auckland  
PB 92019  
Auckland, 1142  
New Zealand

Email: [brian.e.carpenter@gmail.com](mailto:brian.e.carpenter@gmail.com)

Sheng Jiang  
Huawei Technologies Co., Ltd  
Q14, Huawei Campus  
No.156 Beiqing Road  
Hai-Dian District, Beijing 100095  
P.R. China

Email: [jiangsheng@huawei.com](mailto:jiangsheng@huawei.com)





INTAREA WG  
Internet-Draft  
Updates: 4861 (if approved)  
Intended status: Standards Track  
Expires: April 15, 2013

S. Chakrabarti  
Ericsson  
E. Nordmark  
Cisco Systems  
M. Wasserman  
Painless Security  
October 12, 2012

Efficiency aware IPv6 Neighbor Discovery Optimizations  
draft-chakrabarti-nordmark-6man-efficient-nd-00

Abstract

IPv6 Neighbor Discovery (RFC 4861) protocol has been designed for neighbor's address resolution, unreachability detection, address autoconfiguration, router advertisement and solicitation. With the progress of Internet adoption on various industries including home, wireless, m2m and Cellular(LTE) networks, there is a desire for optimizing legacy IPv6 Neighbor Discovery protocol to be more efficient in terms of number of signaling messages in the network. This document describes a method of optimization by reducing periodic multicast messages, frequent Neighbor Solicitation messages and supports interoperability with legacy IPv6 nodes and avoids Duplicate Address Detection by introducing an address Registration mechanism. Efficient IPv6 Neighbor Discovery protocol is useful for energy-efficient IPv6 networks as well as Data Center and Home Networks.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the

document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	4
2. Definition Of Terms . . . . .	6
3. Assumptions for efficiency-aware Neighbor Discovery . . . . .	7
4. The set of Requirements for efficiency and optimization . . . . .	7
5. Basic Operations . . . . .	8
6. Applicability Statement . . . . .	8
7. New Neighbor Discovery Options and Messages . . . . .	9
7.1. Address Registration Option . . . . .	9
7.2. Refresh and De-registration . . . . .	11
7.3. A New Router Advertisement Flag . . . . .	11
8. efficiency-aware Neighbor Discovery Messages . . . . .	12
9. efficiency-aware Host Behavior . . . . .	13
10. The Energy Aware Default Router (NEAR) Behavior . . . . .	14
10.1. Router Configuration Modes . . . . .	14
11. NCE Management in efficiency-aware Routers . . . . .	15
11.1. Handling ND DOS Attack . . . . .	16
12. Mixed-Mode Operations . . . . .	17
13. Bootstrapping . . . . .	18
14. Handling Sleepy Nodes . . . . .	19
15. Duplicate Address Detection . . . . .	20
16. Use Case Analysis . . . . .	20
16.1. Data Center Routers on the link . . . . .	20
16.2. Edge Routers and Home Networks . . . . .	20
16.3. M2M Networks . . . . .	21
16.4. Cellular and Wi-fi Networks . . . . .	21
17. Mobility Considerations . . . . .	21
18. Other Considerations . . . . .	21
18.1. Detecting Network Attachment(DNA) . . . . .	21
18.2. ND-Proxy . . . . .	22
18.3. DHCPv6 Interaction . . . . .	22
19. Updated Neighbor Discovery Constants . . . . .	22
20. Security Considerations . . . . .	23
21. IANA Considerations . . . . .	23
22. Changelog . . . . .	23
23. Acknowledgements . . . . .	23
24. References . . . . .	24
24.1. Normative References . . . . .	24
24.2. Informative References . . . . .	24
Authors' Addresses . . . . .	25

## 1. Introduction

IPv6 ND [ND] is based on multicast signaling messages on the local link in order to avoid broadcast messages. Following power-on and initialization of the network in IPv6 Ethernet networks, a node joins the solicited-node multicast address on the interface and then performs duplicate address detection (DAD) for the acquired link-local address by sending a solicited-node multicast message to the link. After that it sends multicast router solicitation (RS) messages to the all-router address to solicit router advertisements. Once the host receives a valid router advertisement (RA) with the "A" flag, it autoconfigures the IPv6 address with the advertised prefix in the router advertisement (RA). Besides this, the IPv6 routers usually send router advertisements periodically on the network. RAs are sent to the all-node multicast address. The minimum RA interval range can be 3sec to 600sec depending on applications. Nodes send Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages to resolve the IPv6 address of the destination on the link. These NS/NA messages are also often multicast messages and it is assumed that the node is on the same link and relies on the fact that the destination node is always powered and generally available.

The periodic RA messages in IPv6 ND [ND], and NS/NA messages require all IPv6 nodes in the link to be in listening mode even when they are in idle cycle. It requires energy for the sleepy nodes which may otherwise be sleeping during the idle period. Non-sleepy nodes also save energy if instead of continuous listening, they actually proactively synchronize their states with one or two entities in the network. With the explosion of Internet-of-things and machine to machine communication, more and more devices would be using IPv6 addresses in the near future. Today, most electricity usage in United States and in developing countries are in the home buildings and commercial buildings; the electronic Internet appliances/tablets etc. are gaining popularities in the modern home networks. These network of nodes must be conscious about saving energy in order to reduce user-cost. This will eventually reduce stress on electrical grids and carbon foot-print.

IPv6 Neighbor Discovery Optimization for 6LoWPAN [6LOWPAN-ND] addresses many of the concerns described above by optimizing the Router advertisement, minimizing periodic multicast packets in the network and introducing two new options - one for node registration and another for prefix dissemination in a network where all nodes in the network are uniquely identified by their 64-bit Interface Identifier. EUI-64 identifiers are recommended as unique Interface Identifiers, however if the network is isolated from the Internet, uniqueness of the identifiers may be obtained by other mechanisms such as a random number generator with lowest collision rate.

Although, the ND optimization [6LOWPAN-ND] applies to 6LoWPAN [LOWPAN] network, the concept is mostly applicable to a power-aware IPv6 network. Therefore, this document generalizes the address registration and multicast reduction in [6LOWPAN-ND] to all IPv6 links.

Thus optimizing the regular IPv6 Neighbor Discovery [ND] to minimize total number of related signaling messages without losing generality of Neighbor Discovery, autoconfiguration and reliable host-router communication, are desirable in any efficient IPv6 networks such as Home, Enterprise networks, Data Centers and Operator's radio networks.

The optimization will be highly effective for links and nodes which do not support multicast and as well as for a multicast network without MLD snooping switches. Moreover, in the MLD-snooping networks the MLD switches will deal with less number of multicasts.

The goal of this document is to provide an efficient Neighbor Discovery Protocol in classic IPv6 subnets and links. Research indicates that often networked- nodes require more energy than stand-alone nodes because a node's energy usage depends on network messages it receives and responds. While reducing energy consumption is essential for battery operated nodes in some machines, saving energy actually a cost factor in business in general as the explosion of more device usage is leading to usage of more servers and network infrastructure in all sectors of the society and business. Thus this document introduces the node registration concept discussed in 6LoWPAN [LOWPAN], without handling the 'multi-level subnets' scenarios that are not the typical usecases in classic IPv6 subnets.

In the process, the node registration method is also deemed to be useful for preventing Neighbor Discovery denial of service ( ND DOS) attacks.

The proposed changes can be used in two different ways. In one case all the hosts and routers on a link implement the new mechanisms, which gives the maximum benefits. In another case the link has a mixture of new hosts and/or routers and legacy [RFC4861] hosts and routers, operating in a mixed-mode providing some of the benefits.

In the following sections the document describes the basic operations of registration methods, optimization of Neighbor Discovery messages, interoperability with legacy IPv6 implementations and provides a section on use-case scenarios where it can be typically applicable.

## 2. Definition Of Terms

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].

### multi-level Subnets:

It is a wireless link determined by one IPv6 off-link prefix in a network where in order to reach a destination with same prefix a packet may have to travel through one more 'intermediate' routers which relays the packet to the next 'intermediate' router or the host in its own.

### Border Router(BR):

A border router is typically located at the junction Internet and Home Network. An IPv6 router with one interface connected to IPv6 subnet and other interface connecting to a non-classic IPv6 interface such as 6LoWPAN interface. Border router is usually the gateway to the IPv6 network or Internet.

### Efficiency-Aware Network:

An Efficiency-Aware network is composed of network elements that are sensitive to energy usage or number of signalling messages in the network. An efficiency-aware network may also contain links that do not support multicast or it does not have MLD snooping capabilities and yet the network likes to communicate most efficiently with minimum number of signaling messages. Data center networks with virtual machines, cellular IPv6 networks, any IPv6 networks with energy-sensitive nodes are examples of Efficiency-Aware networks.

### IPv6 ND-efficiency-aware Router(NEAR):

It is the default Router of the single hop IPv6 subnet. This router implements the optimizations specified in this document. This router should be able to handle both legacy IPv6 nodes and nodes that sends registration request.

### Efficiency-Aware Host(EAH):

A host in a IPv6 network is considered a IPv6 node without routing and forwarding capability. The EAH is the host which implements the host functionality for optimized Neighbor Discovery mentioned in this document.

### Legacy IPv6 Host:

A host in a IPv6 network is considered a IPv6 node without routing and forwarding capability and implements RFC 4861 host functions.

### Legacy IPv6 Router:

An IPv6 Router which implements RFC 4861 Neighbor Discovery protocols.

**EUI-64:**

It is the IEEE defined 64-bit extended unique identifier formed by concatenation of 24-bit or 36-bit company id value by IEEE Registration Authority and the extension identifier within that company-id assignment. The extension identifiers are 40-bit (for 24-bit company-id) or 28-bit (for the 36-bit company-id) respectively.

**3. Assumptions for efficiency-aware Neighbor Discovery**

- o The efficiency-aware nodes in the network carry unique interface ID in the network in order to form the auto-configured IPv6 address uniquely. An EUI-64 interface ID required for global communication.
- o All nodes are single IPv6-hop away from their default router in the subnet.
- o /64-bit IPv6 prefix is used for Stateless Auto-address configuration (SLAAC). The IPv6 Prefix may be distributed with Router Advertisement (RA) from the default router to all the nodes in that link.

**4. The set of Requirements for efficiency and optimization**

- o Node Registration: Node initiated Registration and address allocation is done in order to avoid periodic multicast Router Advertisement messages and often Neighbor Address resolution can be skipped as all packets go via the default router which now knows about all the registered nodes. Node Registration enables reduction of all-node and solicited-node multicast messages in the subnet.
- o Address allocation of registered nodes [ND] are performed using IPv6 Autoconfiguration [AUTOCONF].
- o Host initiated Registration and Refresh is done by sending a Router Solicitation and then a Neighbor Solicitation Message using Address Registration Option (described below).
- o The node registration may replace the requirement of doing Duplicate Address Detection.
- o Sleepy hosts are supported by this Neighbor Discovery procedures as they are not woken up periodically by Router Advertisement multicast messages or Neighbor Solicitation multicast messages. Sleepy nodes may wake up in its own schedule and send unicast registration refresh messages when needed.
- o Since this document requires formation of an IPv6 address with a unique 64-bit Interface ID(EUI-64) is required for global IPv6 addresses, if the network is an isolated one and uses ULA [ULA] as its IPv6 address then the deployment should make sure that each

- MAC address in that network has unique address and can provide a unique 64-bit ID for each node in the network.
- o /64-bit Prefix is required to form the IPv6 address.
- o MTU requirement is same as IPv6 network.

## 5. Basic Operations

In the efficient-nd IPv6 Network, the NEAR routers are the default routers for the efficiency-aware hosts (EAH). During the startup or joining the network the host does not wait for the Router Advertisements as the NEAR routers do not perform periodic multicast RA as per RFC 4861. Instead, the EAH sends a multicast RS to find out a NEAR router in the network. The RS message is the same as in RFC 4861. The advertising routers in the link responds to the RS message with RA with Prefix Information Option and any other options configured in the network. If EAH hosts will look for a RA from a NEAR (E-flag) and choose a NEAR as its default router and consequently sends a unicast Neighbor Solicitation Message with ARO option in order to register itself with the default router. The EAH does not do Duplicate Address Detection or NS Resolution of addresses. NEAR maintains a binding of registered nodes and registration life-time information along with the neighbor Cache information. The NEAR is responsible for forwarding all the messages from its EAH including on-link messages from one EAH to another. For details of protocol operations please see the sections below.

When a IPv6 network consists of both legacy hosts and EAH, and if the NEAR is configured for 'mixed mode' operation, it should be able to handle ARO requests and send periodic RA. Thus it should be able to serve both efficiency-aware hosts and legacy hosts. Similarly, a legacy host compatible EAH falls back to RFC 4861 host behavior if a NEAR is not present in the link. See the section on 'Mixed Mode Operations' for details below.

## 6. Applicability Statement

This document aims to guide the implementors to choose an appropriate IPv6 neighbor discovery and Address configuration procedures suitable for any efficient IPv6 network. These optimization is equally useful for the energy-sensitive, non-multicast links and for classical IPv6 networks i.e home networks, Data-Center IPv6 overlay networks where saving Neighbor Discovery messages will reduce cost and increase bandwidth availability. See use cases towards the end of the document.

Note that the specification allows 'Mixed-mode' operation in the



efficiency-aware nodes for backward compatibility and transitioning to a complete efficiency-aware network of hosts and routers. Though the efficiency-aware only nodes will minimize the ND signalling and DOS attacks in the LAN.

Applicability of this solution is limited to the legacy IPv6 nodes and subnets and it optimizes the generic IPv6 signalling activities at network layer. However, further optimization at the application layers are possible for increased efficiency based on particular usecases and applications.

## 7. New Neighbor Discovery Options and Messages

This section will discuss the registration and de-registration procedure of the hosts in the network.

### 7.1. Address Registration Option

The Address Registration Option (ARO) is useful for avoiding Duplicate Address Detection messages since it requires a unique ID for registration. The address registration is used for maintaining reachability of the node or host by the router. This option is exactly the same as in [6LOWPAN-ND] which is reproduced here for the benefits of the readers.

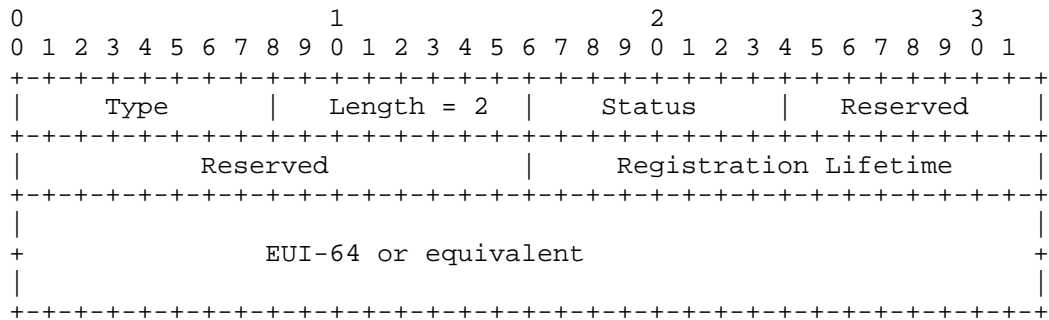
The routers keep track of host IP addresses that are directly reachable and their corresponding link-layer addresses. This is useful for lossy and lowpower networks and as well as wired networks. An Address Registration Option (ARO) can be included in unicast Neighbor Solicitation (NS) messages sent by hosts. Thus it can be included in the unicast NS messages that a host sends as part of Neighbor Unreachability Detection to determine that it can still reach a default router. The ARO is used by the receiving router to reliably maintain its Neighbor Cache. The same option is included in corresponding Neighbor Advertisement (NA) messages with a Status field indicating the success or failure of the registration. This option is always host initiated.

The ARO is required for reliability and power saving. The lifetime field provides flexibility to the host to register an address which should be usable (the reachability information may be propagated to the routing protocols) during its intended sleep schedule of nodes that switches to frequent sleep mode.

The sender of the NS also includes the EUI-64 of the interface it is registering an address from. This is used as a unique ID for the detection of duplicate addresses. It is used to tell the difference

between the same node re-registering its address and a different node (with a different EUI-64) registering an address that is already in use by someone else. The EUI-64 is also used to deliver an NA carrying an error Status code to the EUI-64 based link-local IPv6 address of the host.

When the ARO is used by hosts an SLLA option MUST be included and the address that is to be registered MUST be the IPv6 source address of the Neighbor Solicitation message.



#### Fields:

Type: TBD1 ( See [6LOWPAN-ND] )

Length: 8-bit unsigned integer. The length of the option in units of 8 bytes. Always 2.

Status: 8-bit unsigned integer. Indicates the status of a registration in the NA response. MUST be set to 0 in NS messages. See below.

Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Registration Lifetime: 16-bit unsigned integer. The amount of time in a unit of 10 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option.

EUI-64: 64 bits. This field is used to uniquely identify the interface of the registered address by including the EUI-64 identifier assigned to it unmodified.

The Status values used in Neighbor Advertisements are:

Status	Description
0	Success
1	Duplicate Address
2	Neighbor Cache Full
3-255	Allocated using Standards Action [RFC2434]

Table 1

### 7.2. Refresh and De-registration

A host SHOULD send a Registration message in order to renew its registration before its registration lifetime expires in order to continue its connectivity with the network. If anytime, the node decides that it does not need the default router's service anymore, it MUST send a de-registration message - i.e, a registration message with lifetime being set to zero. A mobile host SHOULD first de-register with the default router before it moves away from the subnet.

### 7.3. A New Router Advertisement Flag

A new Router Advertisement flag [RF] is needed in order to distinguish a router advertisement from a efficiency-aware default router or a legacy IPv6 router. This flag is ignored by the legacy IPv6 hosts. EAH hosts use this flag in order to discover a NEAR router if it receives multiple RA from both legacy and NEAR routers.

```

0 1 2 3 4 5 6 7
+---+---+---+---+
|M|O|H|Prf|P|E|R|
+---+---+---+---+

```

The 'E' bit above MUST be 1 when a IPv6 router implements and configures the efficiency-aware Router behavior for Neighbor Discovery as per this document. All other cases E bit is 0.

The legacy IPv6 hosts will ignore the E bit in RA advertisement. All EAH MUST look for E bit in RA in order to determine the efficiency-aware support in the default router in the link.

This document assumes that an implementation will have configuration knobs to determine whether it is running in classical IPv6 ND [ND] or

Optimized Energy Aware ND (this document) mode or both(Mixed mode).

## 8. efficiency-aware Neighbor Discovery Messages

Router Advertisement(RA): Periodic RAs SHOULD be avoided. Only solicited RAs are RECOMMENDED. An RA MUST contain the Source Link-layer Address option containing Router's link-layer address (this is optional in [ND]). An RA MUST carry Prefix information option with L bit being unset, so that hosts do not multicast any NS messages as part of address resolution. A new flag (E-flag) is introduced in the RA in order to characterize the efficiency-aware mode support. Unlike RFC4861 which suggests multicast Router Advertisements, this specification optimizes the exchange by always unicasting RAs in response to RS. This is possible since the RS always includes a SLLA option, which is used by the router to unicast the RA.

Router Solicitation(RS): Upon system startup, the node sends a multicast or link broadcast (when multicast is not supported at the link-layer) RS to find out the available routers in the link. An RS may be sent at other times as described in section 6.3.7 of RFC 4861. A Router Solicitation MUST carry Source Link-layer Address option. Since no periodic RAs are allowed in the efficiency-aware IPv6 network, the host may send periodic unicast RS to the routers. The time-periods for the RS varies on the deployment scenarios and the Default Router Lifetime advertised in the RAs.

Default Router Selection: Same as in section 6.3.6 of RFC 4861[ND].

Neighbor Solicitation (NS): Neighbor solicitation is used between the hosts and the default-router as part of NUD and registering the host's address(es). An NS message MUST use the Address Registration option in order to accomplish the registration.

Neighbor Advertisement (NA): As defined in [ND] and ARO option.

Redirect Messages: A router SHOULD NOT send a Redirect message to a host since the link has non-transitive reachability. The host behavior is same as described in section 8.3 of RFC 4861[ND], i.e. a host MUST NOT send or accept redirect

messages when in efficiency-aware mode.  
Same as in RFC 4861[ND]  
MTU option: As per the RFC 4861.  
Address Resolution: No NS/NA are sent as the prefixes are treated as off-link. Thus no address resolution is performed at the hosts. The routers keep track of Neighbor Solicitations with Address Registration options(ARO) and create an extended neighbor cache of reachable addresses. The router also knows the nexthop link-local address and corresponding link-layer address when it wants to route a packet.  
Neighbor Unreachability Detection(NUD): NUD is performed in "forward-progress" fashion as described in section 7.3.1 of RFC 4861[ND]. However, if Address Registration Option is used, the NUD SHOULD be combined with the Re-registration of the node. This way no extra message for NUD is required.

## 9. efficiency-aware Host Behavior

A host sends Router Solicitation at the system startup and also when it suspects that one of its default routers have become unreachable(after NUD fails). The EAH MUST process the E-bit in RA as described in this document. The EAH MUST use ARO option to register with the neighboring NEAR router.

A host SHOULD be able to autoconfigure its IPv6 addresses using the IPv6 prefix obtained from Router Advertisement. The host SHOULD form its link-local address from the EUI-64 as specified by IEEE Registration Authority and RFC 2373. If this draft feature is implemented and configured, the host MUST NOT re-direct Neighbor Discovery messages. The host does not require to join solicited-node multicast address but it MUST join the all-node multicast address.

A host always sends packets to (one of) its default router(s). This is accomplished by the routers never setting the 'L' flag in the Prefix options.

The host is unable to forward routes or participate in a routing protocol. A legacy IPv6 Host compliant EAH SHOULD be able to fall back to RFC 4861 host behavior if there is no efficiency-aware router (NEAR) in the link.

The efficiency-aware host MUST NOT send or accept re-direct messages.

It does not join solicited node multicast address.

#### 10. The Energy Aware Default Router (NEAR) Behavior

The main purpose of the default router in the context of this document is to receive and process the registration request, forward packets from one neighbor to the other, informs the routing protocol about the un-availability of the registered nodes if the routing protocol requires this information for the purpose of mobility or fast convergence. A default router (NEAR) behavior may be observed in one or more interfaces of a Border Router(BR).

A Border Router normally may have multiple interfaces and connects the nodes in a link like a regular IPv6 subnet(s) or acts as a gateway between separate networks such as Internet and home networks. The Border Router is responsible for distributing one or more /64 prefixes to the nodes to identify a packet belonging to the particular network. One or more of the interfaces of the Border Router may be connected with the efficiency-aware hosts or a efficiency-aware router(NEAR).

The efficiency-aware default router MUST not send periodic RA unless it is configured to support both legacy IPv6 and efficiency-aware hosts. If the Router is configured for efficiency-aware hosts support, it MUST send Router Advertisements with E-bit flag ON and MUST NOT set 'L' bit in the advertisements.

The router SHOULD NOT garbage collect Registered Neighbor Cache entries since they need to retain them until the Registration Lifetime expires. If a NEAR receives a NS message from the same host one with ARO and another without ARO then the NS message with ARO gets the precedence and the NS without ARO is ignored. This behavior protects the router from Denial Of Service attacks. Similarly, if Neighbor Unreachability Detection on the router determines that the host is UNREACHABLE (based on the logic in [ND]), the Neighbor Cache entry SHOULD NOT be deleted but be retained until the Registration Lifetime expires. If an ARO arrives for an NCE that is in UNCREACHABLE state, that NCE should be marked as STALE.

A default router keeps a cache for all the nodes' IP addresses, created from the Address Registration processing.

##### 10.1. Router Configuration Modes

An efficiency-aware Router(NEAR) MUST be able to configure in efficiency-aware-only mode where it will expect all hosts register with the router following RS; thus will not support legacy hosts.

However, it will create legacy NCE for NS messages for other routers in the network. This mode is able to prevent ND flooding on the link.

An efficiency-aware Router(NEAR) SHOULD be able to have configuration knob to configure itself in Mixed-Mode where it will support both efficiency-aware hosts and legacy hosts. However even in mixed-mode the router should check for duplicate entries in the NCE before creating a new ones and it should rate-limit creating new NCE based on requests from the same host MAC address.

The RECOMMENDED default mode of operation for the efficiency-aware router is Mixed-mode.

## 11. NCE Management in efficiency-aware Routers

The use of explicit registrations with lifetimes plus the desire to not multicast Neighbor Solicitation messages for hosts imply that we manage the Neighbor Cache entries slightly differently than in [ND]. This results in two different types of NCEs and the types specify how those entries can be removed:

Legacy:	Entries that are subject to the normal rules in [ND] that allow for garbage collection when low on memory. Legacy entries are created only when there is no duplicate NCE. In mixed-mode and efficiency-aware mode the legacy entries are converted to the registered entries upon successful processing of ARO. Legacy type can be considered as union of garbage-collectible and Tentative Type NCEs described in [6LOWPAN-ND].
Registered:	Entries that have an explicit registered lifetime and are kept until this lifetime expires or they are explicitly unregistered.

Note that the type of the NCE is orthogonal to the states specified in [ND].

When a host interacts with a router by sending Router Solicitations that does not match with the existing NCE entry of any type, a Legacy NCE is first created. Once a node successfully registers with a Router the result is a Registered NCE. As Routers send RAs to legacy hosts, or receive multicast NS messages from other Routers the result is Legacy NCEs. There can only be one kind of NCE for an IP address at a time.

A Router Solicitation might be received from a host that has not yet registered its address with the router or from a legacy[ND] host in the Mixed-mode of operation.

In the 'Efficiency-aware' only mode the router MUST NOT modify an existing Neighbor Cache entry based on the SLLA option from the Router Solicitation. Thus, a router SHOULD create a tentative Legacy Neighbor Cache entry based on SLLA option when there is no match with the existing NCE. Such a legacy Neighbor Cache entry SHOULD be timed out in TENTATIVE\_LEGACY\_NCE\_LIFETIME seconds unless a registration converts it into a Registered NCE.

However, in 'Mixed-mode' operation, the router does not require to keep track of TENTATIVE\_LEGACY\_NCE\_LIFETIME as it does not know if the RS request is from a legacy host or the efficiency-aware hosts. However, it creates the legacy type of NCE and updates it to a registered NCE if the ARO NS request arrives corresponding to the legacy NCE. Successful processing of ARO will complete the NCE creation phase.

If ARO did not result in a duplicate address being detected, and the registration life-time is non-zero, the router creates and updates the registered NCE for the IPv6 address. If the Neighbor Cache is full and new entries need to be created, then the router SHOULD respond with a NA with status field set to 2. For successful creation of NCE, the router SHOULD include a copy of ARO and send NA to the requestor with the status field 0. A TLLA(Target Link Layer) Option is not required with this NA.

Typically for efficiency-aware routers (NEAR), the registration life-time and EUI-64 are recorded in the Neighbor Cache Entry along with the existing information described in [ND]. The registered NCE are meant to be ready and reachable for communication and no address resolution is required in the link. The efficiency-aware hosts will renew their registration to keep maintain the state of reachability of the NCE at the router. However the router may do NUD to the idle or unreachable hosts as per [ND].

#### 11.1. Handling ND DOS Attack

IETF community has discussed possible issues with /64 DOS attacks on the ND networks when an attacker host can send thousands of packets to the router with a on-link destination address or sending RS messages to initiate a Neighbor Solicitation from the neighboring router which will create a number of INCOMPLETE NCE entries for non-existent nodes in the network resulting in table overflow and denial of service of the existing communications.



The efficiency-aware behavior documented in this specification avoids the ND DOS attacks by:

- o Having the hosts register with the default router
- o Having the hosts send their packets via the default router
- o Not resolving addresses for the Routing Solicitor by mandating SLLA option along with RS
- o Checking for duplicates in NCE before the registration
- o Checking against the MAC-address and EUI-64 id is possible now for NCE matches
- o On-link IPv6-destinations on a particular link must be registered else these packets are not resolved and extra NCEs are not created

It is recommended that Mixed-mode operation and legacy hosts SHOULD NOT be used in the IPv6 link in order to avoid the ND DOS attacks. For the general case of Mixed-mode the router does not create INCOMPLETE NCEs for the registered hosts, but it follows the [ND] steps of NCE states for legacy hosts.

## 12. Mixed-Mode Operations

Mixed-Mode operation discusses the protocol behavior where the IPv6 subnet is composed with legacy IPv6 Neighbor Discovery compliant nodes and efficiency-aware IPv6 nodes implementing this specification.

The mixed-mode model SHOULD support the following configurations in the IPv6 link:

- o The legacy IPv6 hosts and efficiency-aware-hosts in the network and a NEAR router
- o legacy IPv6 default-router and efficiency-aware hosts(EAH) in the link
- o one router is in mixed mode and the link contains both legacy IPv6 hosts and EAH
- o A link contains both efficiency-aware IPv6 router and hosts and legacy IPv6 routers and hosts and each host should be able to communicate with each other.

In mixed-mode operation, a NEAR MUST be configured for mixed-mode in order to support the legacy IPv6 hosts in the network. In mixed-mode, the NEAR MUST act as proxy for Neighbor Solicitation for DAD and Address Resolution on behalf of its registered hosts on that link. It should follow the NCE management for the EAH as described in this document and follow RFC 4861 NCE management for the legacy IPv6 hosts. Both in mixed-mode and efficiency-aware mode, the NEAR sets E-bit flag in the RA and does not set 'L' on-link bit.

If a NEAR receives NS message from the same host one with ARO and another without ARO then the NS message with ARO gets the precedence.

An efficiency-aware Host implementation SHOULD support falling back to legacy IPv6 node behavior when no efficiency-aware routers are available in the network during the startup. If the EAH was operational in efficiency-aware mode and it determines that the NEAR is no longer available, then it should send a RS and find an alternate default router in the link. If no efficiency-aware router is indicated from the RA, then the EAH SHOULD fall back into RFC 4861 behavior. On the otherhand, in the efficiency-aware mode EAH SHOULD ignore multicast Router Advertisements(RA) sent by the legacy and Mixed-mode routers in the link.

The routers that are running on efficiency-aware mode or legacy mode SHOULD NOT dynamically switch the mode without flushing the Neighbor Cache Entries.

### 13. Bootstrapping

If the network is a efficiency-aware IPv6 subnet, and the efficiency-aware Neighbor Discovery mechanism is used by the hosts and routers as described in this document. At the start, the node uses its link-local address to send Router Solicitation and then it sends the Node Registration message as described in this document in order to form the address. The Duplicate address detection process should be skipped if the network is guaranteed to have unique interface identifiers which is used to form the IPv6 address.

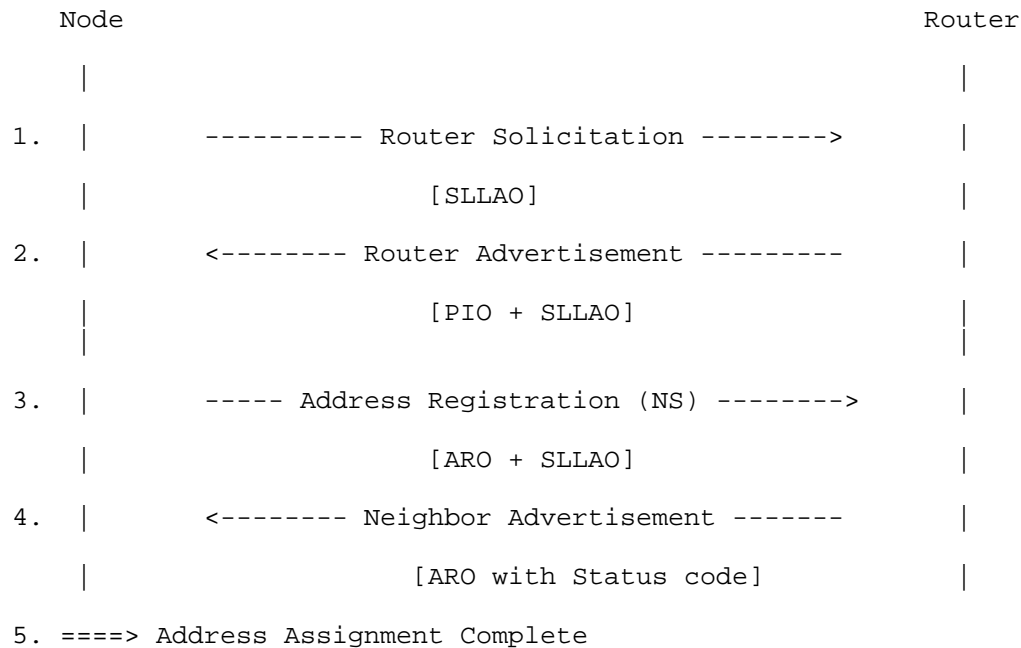


Figure 1: Neighbor Discovery Address Registration and bootstrapping

In the mixed mode operation, it is expected that logically there will be at least one legacy IPv6 router and another NEAR router present in the link. The legacy IPv6 router will follow RFC 4861 behavior and NEAR router will follow the efficiency-aware behavior for registration, NCE maintenance, forwarding packets from a EAH and it will also act as a ND proxy for the legacy IPv6 hosts querying to resolve a EAH node.

A legacy IPv6 host and EAH are not expected to see a difference in their bootstrapping if both legacy and efficiency-aware functionalities of routers are available in the network. It is RECOMMENDED that the EAH implementation SHOULD be able to behave like a legacy IPv6 host if it discovers that no efficiency-aware routing support is present in the link.

#### 14. Handling Sleepy Nodes

The solution allows the sleepy nodes to complete its sleep schedule without waking up due to periodic Router Advertisement messages or due to Multicast Neighbor Solicitation for address resolution. The node registration lifetime SHOULD be synchronized with its sleep

interval period in order to avoid waking up in the middle of sleep for registration refresh. Depending on the application, the registration lifetime SHOULD be equal to or integral multiple of a node's sleep interval period.

#### 15. Duplicate Address Detection

In efficiency-aware mode, there is no need for Duplicate Address Detection(DAD) assuming that the deployment ensures unique 64bit identification availability for each registered host. In the event of collision of EUI64 field of ARO by two registration requests, the later request is denied if the first one is a valid request. The denied EAH node SHOULD pick another alternative IPv6 address to register to prevent further registration denial. The method of assignment of alternate IPv6 address is out of scope of this document.

#### 16. Use Case Analysis

This section provides applicability scenarios where the efficiency-aware Neighbor Discovery will be most beneficial.

##### 16.1. Data Center Routers on the link

efficiency-aware Routers and hosts are useful in IPv6 networks in the Data Center as they produce less signaling and also provides ways to minimize the ND flood of messages. Moreover, this mechanism will work with data-center nodes which are deliberately in sleep mode for saving energy.

This solution will work well in Data Center Virtual network and VM scenarios where number of VLANs are very high and ND signalings are undesirably high due the multicast messaging and periodic Router Advertisements and Neighbor Unreachability detections.

##### 16.2. Edge Routers and Home Networks

An Edge Router in the network will also benefit implementing the efficiency-aware Neighbor Discovery behavior in order to save the signaling and keeping track of the registered nodes in its domain. A BNG sits at the operator's edge network and often the BNG has to handle a large number of home CPEs. If a BNG runs Neighbor Discovery protocol and acts as the default router for the CPE at home, this solution will be helpful for reducing the control messages and improving network performances.

The same solution can be run on CPE or Home Residential Gateways to assign IPv6 addresses to the wired and wireless home devices without the problem of ND flooding issues and consuming less power. It provides mechanism for the sleepy nodes to adjust their registration lifetime according to their sleep schedules.

### 16.3. M2M Networks

Any Machine-to-machine(M2M) networks such as IPv6 surveillance networks, wireless monitoring networks and other m2m networks desire for efficiency-aware control protocols and dynamic address allocation. The in-built address allocation and autoconfiguration mechanism in IPv6 along with the default router capability will be useful for the simple small-scale networks without having the burden of DHCPv6 service and Routing Protocols.

### 16.4. Cellular and Wi-fi Networks

The cellular and Wi-fi IPv6 devices can act as efficiency-aware hosts and register with their nearest default router. This will reduce number of signalings and the registration method(ARO) will provide operators a mechanism for tracking the nodes in the default router.

## 17. Mobility Considerations

If the hosts move from one subnet to another, they MUST first de-register and then register themselves in the new subnet or network. Otherwise, the regular IPv6 Mobility [IPV6M]behavior applies.

## 18. Other Considerations

### 18.1. Detecting Network Attachment(DNA)

IPv6 DNA[DNA] uses unicast ND probes and link-layer indications to detect movement of its network attachments. Upon detecting link-layer indication, it sends a all-routers Solicitation message. However DNA [DNA] optimizes the IPv6 address operability while a node is moving and its network attachments are changing with respect to the neighboring routers. This document does not expect Router Advertisements from the neighboring routers, thus this solution will rely on the ND probes for movement detection and as well as link-layer indication. When the node implements this document along with DNA, it MUST send ARO option with its Neighbor Solicitation unicast message if the candidate router advertisement indicates that the router is a NEAR router. If the candiate router is a legacy router then and it is the only choice then the EAH host adapt to the legacy

behavior. However if EAH node implements DNA host capability as well, then it SHOULD give preference to the NEAR routers in its candidate list of routers.

## 18.2. ND-Proxy

ND proxy support in mixed-mode operation: The ND Proxy will continue to support the legacy IPv6 Neighbor Solicitation requests in the mixed mode. The NEAR router SHOULD act as the ND proxy on behalf of EAH hosts for the legacy nodes' NS requests for EAH.

## 18.3. DHCPv6 Interaction

Co-existence with DHCP: For classical IPv6, if DHCPv6 or central address allocation mechanism is used, then Neighbor Discovery autoconfiguration is not used for global address allocation. However, link-local duplicate address detection, Neighbor solicitation, Neighbor unreachability detection are still used. Upon assignment of the IPv6-address from DHCPv6, a EAH node SHOULD then register the IP-address with the default router for avoiding Duplicate address detection and Address Resolution. For Legacy DHCPv6 nodes there is no change in behavior. NOTE: DHCPv6 Server MUST be notified by NEAR for its efficiency-aware service interfaces. DHCPv6 server then SHOULD inform the DHCP requestor node about the default-router capability during the address assignment period.

Although the solution described in this document prevents unnecessary multicast messages in the IPv6 ND procedure, it does not affect normal IPv6 multicast packets and ability of nodes to join and leave the multicast group or forwarding multicast traffic or responding to multicast queries.

## 19. Updated Neighbor Discovery Constants

This section discusses the updated default values of ND constants based on [ND] section 10. New and changed constants are listed only for efficiency-aware-nd implementation.

### Router Constants:

MAX_RTR_ADVERTISEMENTS(NEW)	3 transmissions
MIN_DELAY_BETWEEN_RAS(CHANGED)	1 second
TENTATIVE_LEGACY_NCE_LIFETIME(NEW)	30 seconds

### Host Constants:

MAX\_RTR\_SOLICITATION\_INTERVAL(NEW)      60 seconds

## 20. Security Considerations

These optimizations are not known to introduce any new threats against Neighbor Discovery beyond what is already documented for IPv6 [RFC 3756].

Section 11.2 of [ND] applies to this document as well.

This mechanism minimizes the possibility of ND /64 DOS attacks in efficiency-aware mode. See Section 11.1.

## 21. IANA Considerations

A new flag (E-bit) in RA has been introduced. IANA assignment of the E-bit flag is required upon approval of this document.

## 22. Changelog

Changes from draft-chakrabarti-nordmark-energy-aware-nd-02:

- o Replaced energy-aware with efficiency-aware which covers both energy efficiency and other operational efficiency
- o Added consideration for DNA, ND proxy and clarified that this is useful for networks with MLD-snooping switches as well
- o Added use-cases, Support for Mixed-mode operations and a diagram for bootstrapping scenario.
- o Clarified its applicability when DHCP is used for address allocation.

## 23. Acknowledgements

The primary idea of this document are from 6LoWPAN Neighbor Discovery document [6LoWPAN-ND] and the discussions from the 6lowpan working group members, chairs Carsten Bormann and Geoff Mulligan and through our discussions with Zach Shelby, editor of the [6LoWPAN-ND].

The inspiration of such a IPv6 generic document came from Margaret Wasserman who saw a need for such a document at the IOT workshop at Prague IETF.

The authors acknowledge the ND denial of service issues and key causes mentioned in the draft-halpern-6man-nddos-mitigation document by Joel Halpern. Thanks to Joel Halpern for pinpointing the problems

that are now addressed in the NCE management section.

The authors like to thank Dave Thaler, Jari Arkko, Ylva Jading, Niklas J. Johnsson for their useful comments on this work.

## 24. References

### 24.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [6LOWPAN-ND] Shelby, Z., Chakrabarti, S., and E. Nordmark, "ND Optimizations for 6LoWPAN", draft-ietf-6lowpan-nd-17.txt (work in progress), June 2011.
- [ND] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6", RFC 4861, September 2007.
- [LOWPAN] Montenegro, G. and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 networks", RFC 4944, September 2007.
- [LOWPANG] Kushalnagar, N. and G. Montenegro, "6LoWPAN: Overview, Assumptions, Problem Statement and Goals", RFC 4919, August 2007.

### 24.2. Informative References

- [IPV6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6), Specification", RFC 2460, December 1998.
- [DNA] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachments in IPv6", RFC 6059, November 2010.
- [AUTOCONF] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Autoconfiguration", RFC 4862, September 2007.
- [SEND] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "Secure



Neighbor Discovery", RFC 3971, March 2005.

[AUTOADHOC]

Baccelli, E. and M. Townsley, "IP Addressing Model in Adhoc Networks", RFC 5889, September 2010.

[NDDOS-halpern]

Halpern, J., "IP Addressing Model in Adhoc Networks", draft-halpern-6man-nddos-mitigation-00.txt (work in progress), October 2011.

[IEEE]

IEEE Computer Society, "IEEE Std. 802.15.4-2003", , October 2003.

[PD]

Miwakawya, S., "Requirements for Prefix Delegation", RFC 3769, June 2004.

[RF]

Haberman, B. and B. Hinden, "IPv6 Router Advertisement Flags option", RFC 5175, March 2008.

[ULA]

"Unique Local IPv6 Addresses", RFC 4193.

[IPV6M]

Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

Authors' Addresses

Samita Chakrabarti  
Ericsson  
San Jose, CA  
USA

Email: samita.chakrabarti@ericsson.com

Erik Nordmark  
Cisco Systems  
San Jose, CA  
USA

Email: nordmark@cisco.com

Margaret Wasserman  
Painless Security

Email: [mrw@lilacglade.org](mailto:mrw@lilacglade.org)



6man Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 25, 2013

A. Matsumoto  
T. Fujisaki  
NTT  
T. Chown  
University of Southampton  
September 21, 2012

Distributing Address Selection Policy using DHCPv6  
draft-ietf-6man-addr-select-opt-06.txt

Abstract

RFC 6724 defines default address selection mechanisms for IPv6 that allow nodes to select appropriate address when faced with multiple source and/or destination addresses to choose between. The RFC 6724 allowed for the future definition of methods to administratively configure the address selection policy information. This document defines a new DHCPv6 option for such configuration, allowing a site administrator to distribute address selection policy overriding the default address selection parameters and policy table, and thus control the address selection behavior of nodes in their site.

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 March 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

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

## 1. Introduction

RFC 3484 [RFC3484] describes default algorithms for selecting an address when a node has multiple destination and/or source addresses to choose from by using an address selection policy. In Section 2 of RFC 6724, it is suggested that the default policy table may be administratively configured to suit the specific needs of a site. This specification defines a new DHCPv6 option for such configuration.

Some problems have been identified with the default RFC 3484 address selection policy [RFC5220]. It is unlikely that any default policy will suit all scenarios, and thus mechanisms to control the source address selection policy will be necessary. Requirements for those mechanisms are described in [RFC5221], while solutions are discussed in [I-D.ietf-6man-addr-select-sol] and [I-D.ietf-6man-addr-select-considerations]. Those documents have helped shape the improvements in the default address selection algorithm [RFC6724] as well as the DHCPv6 option defined in this specification.

### 1.1. Conventions Used in This Document

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].

## 1.2. Terminology

This document uses the terminology defined in [RFC2460] and the DHCPv6 specification defined in [RFC3315]

## 2. Address Selection options

The Address Selection option provides the address selection policy table, and some other configuration parameters.

A address selection option contains zero or more policy table options. Multiple policy table options in a Policy Table option constitute a single policy table.

The format of the Address Selection option is given below.

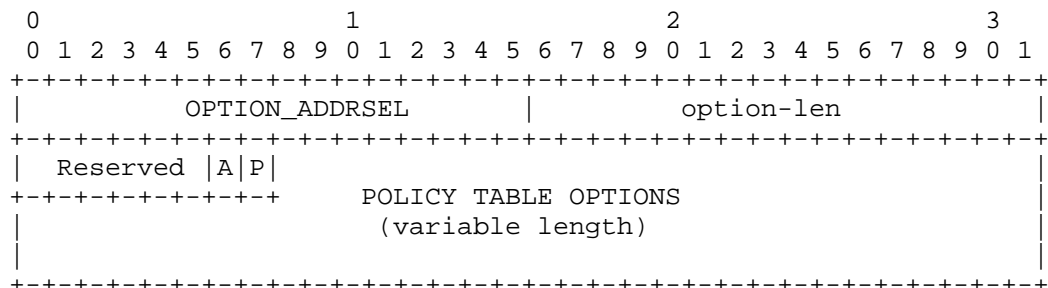


Figure 1: Address Selection option format

option-code: OPTION\_ADDRSEL (TBD).

option-len: The total length of the Reserved field, A, P flags, and POLICY TABLE OPTONS in octets.

Reserved: Reserved field. Server MUST set this value to zero and client MUST ignore its content.

A: Automatic Row Addition flag. This flag toggles the Automatic Row Addition flag at client hosts, which is described in the section 2.1 in RFC 6724 [RFC6724]. If this flag is set to 1, it does not change client host behavior, that is, a client MAY automatically add additional site-specific rows to the policy table. If set to 0, the Automatic Row Addition flag is disabled, and a client MAY NOT automatically add rows to the policy table.

P: Privacy Preference flag. This flag toggles the Privacy Preference flag at client hosts, which is described in the section 5 in RFC 6724 [RFC6724]. If this flag is set to 1, it does not change client host behavior, that is, a client SHOULD prefer temporary addresses. If set to 0, the Privacy Preference flag is disabled, and a client SHOULD prefer public addresses.

POLICY TABLE OPTIONS: Zero or more Address Selection Policy Table options described below.

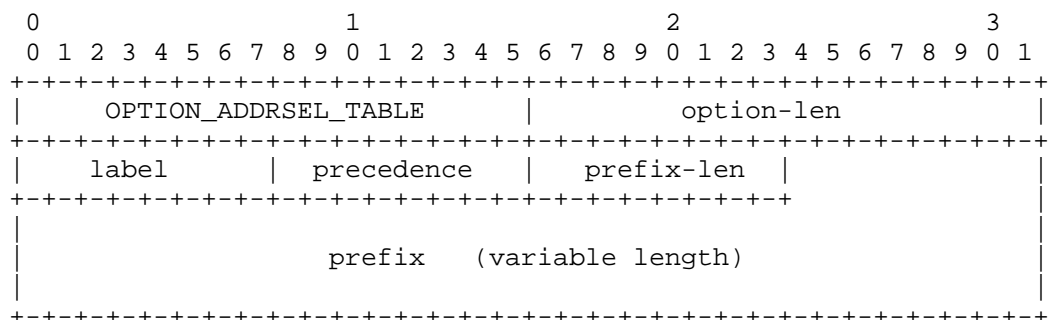


Figure 2: Address Selection Policy Table option format

option-code: OPTION\_ADDRSEL\_TABLE (TBD).

option-len: The total length of the label field, precedence field, prefix-len field, and prefix field.

label: An 8-bit unsigned integer; this value is for correlation of source address prefixes and destination address prefixes.

precedence: An 8-bit unsigned integer; this value is used for sorting destination addresses.

prefix-len: An 8-bit unsigned integer; the number of leading bits in the prefix that are valid. The value ranges from 0 to 128.

prefix: A variable-length field containing an IP address or the prefix of an IP address. An IPv4-mapped address [RFC4291] must be used to represent an IPv4 address as a prefix value. The prefix should be zero padded up to the next octet boundary. So the length of this field should be between 0 and 16 bytes.

### 3. Appearance of the Address Selection options

The Address Selection options MUST NOT appear in any messages other than the following ones: Solicit, Advertise, Request, Renew, Rebind, Reconfigure, Information-Request, and Reply.

### 4. Processing the Policy Table option

This section describes how to process received Policy Table option at the DHCPv6 client.

This option's concept is to serve as a hint for a node about how to behave in the network. So, basically, it should be up to the node's administrator how to deal with the received policy information in the way described below.

#### 4.1. Handling of the local policy table

RFC 6724 defines the default policy table. Also, a user is usually able to configure the policy table to satisfy his requirement.

The client implementation SHOULD provide the following choices to the user:

- a) It receives distributed policy table, and replaces the existing policy tables with that.
- b) It preserves the default policy table, or manually configured policy.

#### 4.2. Handling of the stale policy table

When the information from the DHCP server goes stale, the policy received from the DHCP server should be removed and the default policy should be restored.

The received information can be considered stale in several cases, such as, when the interface goes down, the DHCP server does not respond for a certain amount of time, and the Information Refresh Time is expired.

#### 4.3. Multi-interface situation

The policy table, and other parameters specified in this document are node-global information by its nature. One of the reason is that the outbound interface is usually chosen after destination address selection. So, a host cannot make use of multiple address selection policy even if they are stored per interface.



Even if the received policy from one source is merged with one from another source, the effect of both policy are more or less changed. The policy table is defined as a whole, so the slightest addition/deletion from the policy table brings a change in semantics of the policy.

It also should be noted that absence of the distributed policy from a certain network interface should not be treated as absence of policy itself, because it may mean preference for the default address selection policy.

Under the above assumptions, how to handle received policy is specified below.

A node MAY use Address Selection options by default in any of the following two cases:

- 1: The host is single-homed, where the host belongs to one administrative network domain exclusively usually through one active network interface.
- 2: The host implements some advanced heuristics to deal with multiple received policy, which is outside the scope of this document.

The above restrictions do not preclude implementations from providing configuration options to enable this option on a certain network interface.

Nor, they do not preclude implementations from storing distributed address selection policies per interface. They can be used effectively on such implementations that adopt per-application interface selection.

## 5. Implementation Considerations

- o The value 'label' is passed as an unsigned integer, but there is no special meaning for the value, that is whether it is a large or small number. It is used to select a preferred source address prefix corresponding to a destination address prefix by matching the same label value within the DHCP message. DHCPv6 clients need to convert this label to a representation specified by each implementation (e.g., string).
- o Currently, the label and precedence values are defined as 8-bit unsigned integers. In almost all cases, this value will be enough.

- o The maximum number of address selection rules that may be conveyed in one DHCPv6 message depends on the prefix length of each rule and the maximum DHCPv6 message size defined in RFC 3315. It is possible to carry over 3,000 rules in one DHCPv6 message (maximum UDP message size). However, it should not be expected that DHCP clients, servers and relay agents can handle UDP fragmentation. So, the number of the options and the total size of the options should be taken care of.
- o Since the number of selection rules could be large, an administrator configuring the policy to be distributed should consider the resulting DHCPv6 message size.

## 6. Security Considerations

A rogue DHCPv6 server could issue bogus address selection policies to a client. This might lead to incorrect address selection by the client, and the affected packets might be blocked at an outgoing ISP because of ingress filtering. Alternatively, an IPv6 transition mechanism might be preferred over native IPv6, even if it is available. To guard against such attacks, a legitimate DHCPv6 server should be communicated through a secure, trusted channel, such as a channel protected by IPsec, SEND and DHCP authentication, as described in section 21 of RFC 3315,

Another threat is about privacy concern. As in the security consideration section of RFC 6724, at least a part of, the address selection policy stored in a host can be leaked by a packet from a remote host. This issue will not be degraded regardless of the introduction of this option, or regardless of whether the host is multihomed or not.

## 7. IANA Considerations

IANA is requested to assign option codes to `OPTION_ADDRSEL`, `OPTION_ADDRSEL_TABLE`, and `OPTION_ADDRSEL_ZONE` from the option-code space as defined in section "DHCPv6 Options" of RFC 3315.

## 8. References

### 8.1. Normative References

[I-D.ietf-6man-stable-privacy-addresses]  
Gont, F., "A method for Generating Stable Privacy-Enhanced

Addresses with IPv6 Stateless Address Autoconfiguration (SLAAC)", draft-ietf-6man-stable-privacy-addresses-00 (work in progress), May 2012.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

## 8.2. Informative References

- [I-D.ietf-6man-addr-select-considerations] Chown, T. and A. Matsumoto, "Considerations for IPv6 Address Selection Policy Changes", draft-ietf-6man-addr-select-considerations-04 (work in progress), October 2011.
- [I-D.ietf-6man-addr-select-sol] Matsumoto, A., Fujisaki, T., and R. Hiromi, "Solution approaches for address-selection problems", draft-ietf-6man-addr-select-sol-03 (work in progress), March 2010.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC5220] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Problem Statement for Default Address Selection in Multi-

Prefix Environments: Operational Issues of RFC 3484  
Default Rules", RFC 5220, July 2008.

[RFC5221] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama,  
"Requirements for Address Selection Mechanisms", RFC 5221,  
July 2008.

## Appendix A. Acknowledgements

Authors would like to thank to Dave Thaler, Pekka Savola, Remi Denis-Courmont, Francois-Xavier Le Bail, Ole Troan, Bob Hinden, Dmitry Anipko, and the members of 6man's address selection design team for their invaluable contributions to this document.

## Appendix B. Past Discussion

- o The 'zone index' value is used to specify a particular zone for scoped addresses. This can be used effectively to control address selection in the site scope (e.g., to tell a node to use a specified source address corresponding to a site-scoped multicast address). However, in some cases such as a link-local scope address, the value specifying one zone is only meaningful locally within that node. There might be some cases where the administrator knows which clients are on the network and wants specific interfaces to be used though. However, in general case, it is really rare case, and the field was removed.

## Authors' Addresses

Arifumi Matsumoto  
NTT NT Lab  
3-9-11 Midori-Cho  
Musashino-shi, Tokyo 180-8585  
Japan

Phone: +81 422 59 3334  
Email: arifumi@nttv6.net

Tomohiro Fujisaki  
NTT NT Lab  
3-9-11 Midori-Cho  
Musashino-shi, Tokyo 180-8585  
Japan

Phone: +81 422 59 7351  
Email: fujisaki@nttv6.net

Tim Chown  
University of Southampton  
Southampton, Hampshire SO17 1BJ  
United Kingdom

Email: tjc@ecs.soton.ac.uk



6man Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 12, 2014

A. Matsumoto  
T. Fujisaki  
NTT  
T. Chown  
University of Southampton  
October 09, 2013

Distributing Address Selection Policy using DHCPv6  
draft-ietf-6man-addr-select-opt-13.txt

Abstract

RFC 6724 defines default address selection mechanisms for IPv6 that allow nodes to select an appropriate address when faced with multiple source and/or destination addresses to choose between. RFC 6724 allows for the future definition of methods to administratively configure the address selection policy information. This document defines a new DHCPv6 option for such configuration, allowing a site administrator to distribute address selection policy overriding the default address selection parameters and policy table, and thus to control the address selection behavior of nodes in their site.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 12, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

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

## 1. Introduction

[RFC6724] describes default algorithms for selecting an address when a node has multiple destination and/or source addresses to choose from by using an address selection policy. This specification defines a new DHCPv6 option for configuring the default policy table.

Some problems were identified with the default address selection policy as originally defined in [RFC3484]. As a result, RFC 3484 was updated and obsoleted by [RFC6724]. While this update corrected a number of issues identified from operational experience, it is unlikely that any default policy will suit all scenarios, and thus mechanisms to control the source address selection policy will be necessary. Requirements for those mechanisms are described in [RFC5221], while solutions are discussed in [I-D.ietf-6man-addr-select-considerations]. Those documents have helped shape the improvements in the default address selection algorithm in [RFC6724] as well as the requirements for the DHCPv6 option defined in this specification.

This option's concept is to serve as a hint for a node about how to behave in the network. Ultimately, while the node's administrator can control how to deal with the received policy information, the implementation SHOULD follow the method described below uniformly, to ease troubleshooting and to reduce operational costs.

### 1.1. Conventions Used in This Document



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].

## 1.2. Terminology

This document uses the terminology defined in [RFC2460] and the DHCPv6 specification defined in [RFC3315]

## 2. Address Selection options

The Address Selection option provides the address selection policy table, and some other configuration parameters.

An Address Selection option contains zero or more policy table options. Multiple policy table options in an Address Selection option constitute a single policy table. When an Address Selection option does not contain a policy table option, it may be used to just convey the A and P flags.

The format of the Address Selection option is given below.

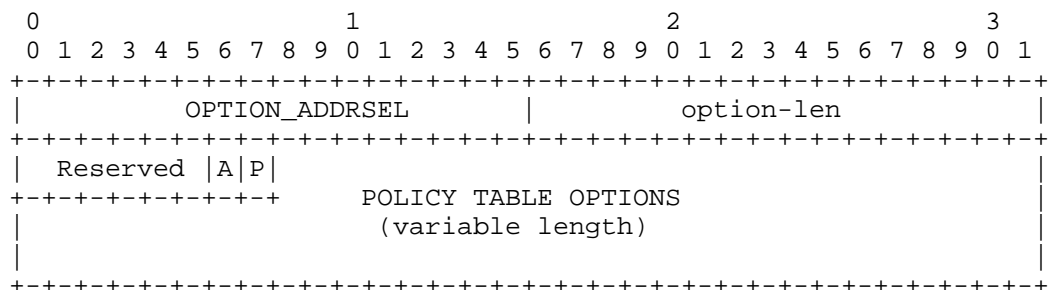


Figure 1: Address Selection option format

option-code: OPTION\_ADDRSEL (TBD).

option-len: The total length of the Reserved field, A, P flags, and POLICY TABLE OPTIONS in octets.

Reserved: Reserved field. The server MUST set this value to zero and the client MUST ignore its content.

- A: Automatic Row Addition flag. This flag toggles the Automatic Row Addition flag at client hosts, which is described in section 2.1 of [RFC6724]. If this flag is set to 1, it does not change client host behavior, that is, a client MAY automatically add additional site-specific rows to the policy table. If set to 0, the Automatic Row Addition flag is disabled, and a client SHOULD NOT automatically add rows to the policy table. If the option contains a POLICY TABLE option, this flag is meaningless, and automatic row addition SHOULD NOT be performed against the distributed policy table. This flag SHOULD be set to 0 only when the Automatic Row Addition at client hosts is harmful for site-specific reasons.
- P: Privacy Preference flag. This flag toggles the Privacy Preference flag on client hosts, which is described in section 5 of [RFC6724]. If this flag is set to 1, it does not change client host behavior, that is, a client will prefer temporary addresses [RFC4941]. If set to 0, the Privacy Preference flag is disabled, and a client will prefer public addresses. This flag SHOULD be set to 0 only when the temporary addresses should not be preferred for site-specific reasons.

POLICY TABLE OPTIONS: Zero or more Address Selection Policy Table options, as described below. This option corresponds to a row in the policy table defined in section 2.1 of [RFC6724].

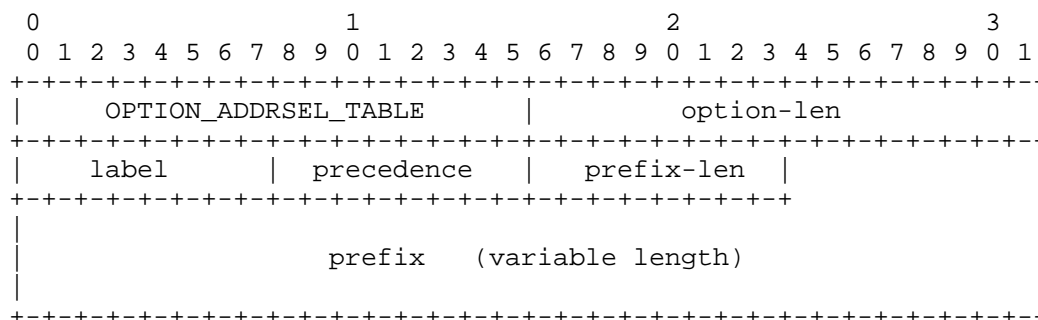


Figure 2: Address Selection Policy Table option format

option-code: OPTION\_ADDRSEL\_TABLE (TBD).

option-len: The total length of the label field, precedence field, prefix-len field, and prefix field.

label: An 8-bit unsigned integer; this value is for correlation of source address prefixes and destination address prefixes. This field is used to deliver a label value in the [RFC6724] policy table.

precedence: An 8-bit unsigned integer; this value is used for sorting destination addresses. This field is used to deliver a precedence value in [RFC6724] policy table.

prefix-len: An 8-bit unsigned integer; the number of leading bits in the prefix that are valid. The value ranges from 0 to 128. If an option with a prefix length greater than 128 is included, the whole Address Selection option MUST be ignored.

prefix: A variable-length field containing an IP address or the prefix of an IP address. An IPv4-mapped address [RFC4291] must be used to represent an IPv4 address as a prefix value. This field is padded with zeros up to the nearest octet boundary when prefix-len is not divisible by 8. This can be expressed using the following equation:  $(\text{prefix-len} + 7) / 8$ . So the length of this field should be between 0 and 16 bytes. For example, the prefix 2001:db8::/60 would be encoded with an prefix-len of 60, the prefix would be 8 octets and would contain octets 20 01 0d b8 00 00 00 00.

### 3. Processing the Address Selection option

This section describes how to process a received Address Selection option at the DHCPv6 client.

This option's concept is to serve as a hint for a node about how to behave in the network. Ultimately, while the node's administrator can control how to deal with the received policy information, the implementation SHOULD follow the method described below uniformly, to ease troubleshooting and to reduce operational costs.

#### 3.1. Handling local configurations

[RFC6724] defines two flags (A, P) and the default policy table. Also, users are usually able to configure the flags and the policy table to satisfy their own requirements.

The client implementation SHOULD provide the following choices to the user.

- (a) replace the existing flags and active policy table with the DHCPv6 distributed flags and policy table.
- (b) preserve the existing flags and active policy table, whether this be the default policy table, or user configured policy.

Choice (a) SHOULD be the default, i.e. that the policy table is not explicitly configured by the user.

### 3.2. Handling stale distributed flags and policy table

When the information from the DHCP server goes stale, the flags and the policy table received from the DHCP server SHOULD be deprecated. The local configuration SHOULD be restored when the DHCP-supplied configuration has been deprecated. In order to implement this, a host can retain the local configuration even after the flags and the policy table is updated by the distributed flags and policy table.

The received information can be considered stale in several cases, e.g., when the interface goes down, the DHCP server does not respond for a certain amount of time, or the Information Refresh Time has expired.

### 3.3. Handling multiple interfaces

The policy table, and other parameters specified in this document, are node-global information by their nature. One reason being that the outbound interface is usually chosen after destination address selection. So a host cannot make use of multiple address selection policies even if they are stored per interface.

The policy table is defined as a whole, so the slightest addition/deletion from the policy table brings a change in the semantics of the policy.

It also should be noted that the absence of a DHCP-distributed policy from a certain network interface should not infer that the network administrator does not care about address selection policy at all, because it may mean there is a preference to use the default address selection policy. So, it should be safe to assume that the default address selection policy should be used where no overriding policy is provided.

Under the above assumptions, we can specify how to handle received policy as follows.

In the absence of distributed policy for a certain network interface, the default address selection policy SHOULD be used. A node should use Address Selection options by default in any of the following two cases:

- 1: A single-homed host SHOULD use default address selection options, where the host belongs exclusively to one administrative network domain, usually through one active network interface.
- 2: Hosts that use advanced heuristics to deal with multiple received policies that are defined outside the scope of this document SHOULD use Address Selection options.

Implementations MAY provide configuration options to enable this protocol on a per interface basis.

Implementations MAY store distributed address selection policies per interface. They can be used effectively on implementations that adopt per-application interface selection.

#### 4. Implementation Considerations

- o The value 'label' is passed as an unsigned integer, but there is no special meaning for the value, that is whether it is a large or small number. It is used to select a preferred source address prefix corresponding to a destination address prefix by matching the same label value within the DHCP message. DHCPv6 clients SHOULD convert this label to a representation appropriate for the local implementation (e.g., string).
- o The maximum number of address selection rules that may be conveyed in one DHCPv6 message depends on the prefix length of each rule and the maximum DHCPv6 message size defined in [RFC3315]. It is possible to carry over 3,000 rules in one DHCPv6 message (maximum UDP message size). However, it should not be expected that DHCP clients, servers and relay agents can handle UDP fragmentation.

Network administrators SHOULD consider local limitations to the maximum DHCPv6 message size that can be reliably transported via their specific local infrastructure to end nodes; and therefore they SHOULD consider the number of options, the total size of the options, and the resulting DHCPv6 message size, when defining their policy table.

## 5. Security Considerations

A rogue DHCPv6 server could issue bogus address selection policies to a client. This might lead to incorrect address selection by the client, and the affected packets might be blocked at an outgoing ISP because of ingress filtering, incur additional network charges, or be misdirected to an attacker's machine. Alternatively, an IPv6 transition mechanism might be preferred over native IPv6, even if it is available. To guard against such attacks, a legitimate DHCPv6 server should communicate through a secure, trusted channel, such as a channel protected by IPsec, SEND and DHCP authentication, as described in section 21 of [RFC3315]. A commonly used alternative mitigation is to employ DHCP snooping at Layer 2.

Another threat surrounds the potential privacy concern as described in the security considerations section of [RFC6724], whereby an attacker can send packets with different source addresses to a destination to solicit different source addresses in the responses from that destination. This issue will not be modified by the introduction of this option, regardless of whether the host is multihomed or not.

## 6. IANA Considerations

IANA is requested to assign option codes to `OPTION_ADDRSEL` and `OPTION_ADDRSEL_TABLE` from the "DHCP Option Codes" registry (<http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xml>).

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

## 7.2. Informative References

- [I-D.ietf-6man-addr-select-considerations]  
Chown, T. and A. Matsumoto, "Considerations for IPv6 Address Selection Policy Changes", draft-ietf-6man-addr-select-considerations-05 (work in progress), April 2013.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC5220] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Problem Statement for Default Address Selection in Multi-Prefix Environments: Operational Issues of RFC 3484 Default Rules", RFC 5220, July 2008.
- [RFC5221] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Requirements for Address Selection Mechanisms", RFC 5221, July 2008.

## Appendix A. Acknowledgements

Authors would like to thank to Dave Thaler, Pekka Savola, Remi Denis-Courmont, Francois-Xavier Le Bail, Ole Troan, Bob Hinden, Dmitry Anipko, Ray Hunter, Rui Paulo, Brian E Carpenter, Tom Petch, and the members of 6man's address selection design team for their invaluable contributions to this document.

## Appendix B. Examples

[RFC5220] gives several cases where address selection problems happen. This section contains some examples for solving those cases by using the DHCP option defined in this text to update the hosts' policy table in a network accordingly. There is also some discussion of example policy tables in sections 10.3 to 10.7 of RFC 6724.

### B.1. Ingress Filtering Problem

In the case described in section 2.1.2 of [RFC5220], the following policy table should be distributed, when Router performs static routing and directs the default route to ISP1 as per Figure 2. By putting the same label value to all IPv6 addresses (::/0) and the local subnet (2001:db8:1000:1::/64), a host picks a source address in this subnet to send a packet via the default route.

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
2001:db8:1000:1::/64	45	1
2001:db8:8000:1::/64	45	14
::ffff:0:0/96	35	4
2002::/16	30	2
2001::/32	5	5
fc00::/7	3	13
::/96	1	3
fec0::/10	1	11
3ffe::/16	1	12

### B.2. Half-Closed Network Problem

In the case described in section 2.1.3 of [RFC5220], the following policy table should be distributed. By splitting the closed network prefix (2001:db8:8000::/36) from all IPv6 addresses (::/0) and giving different labels, the closed network prefix will only be used when packets are destined for the closed network.

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
2001:db8:8000::/36	45	14
::ffff:0:0/96	35	4
2002::/16	30	2
2001::/32	5	5
fc00::/7	3	13
::/96	1	3
fec0::/10	1	11
3ffe::/16	1	12

### B.3. IPv4 or IPv6 Prioritization



In the case described in section 2.2.1 of [RFC5220], the following policy table should be distributed to prioritize IPv6. This case is also described in [RFC6724]

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1
::ffff:0:0/96	100	4
2002::/16	30	2
2001::/32	5	5
fc00::/7	3	13
::/96	1	3
fec0::/10	1	11
3ffe::/16	1	12

#### B.4. ULA or Global Prioritization

In the case described in section 2.2.3 of [RFC5220], the following policy table should be distributed, or Automatic Row Addition flag should be set to 1. By splitting the ULA in this site (fc12:3456:789a::/48) from all IPv6 addresses (::/0) and giving it higher precedence, the ULA will be used to connect to servers in the same site.

Prefix	Precedence	Label
::1/128	50	0
fc12:3456:789a::/48	45	14
::/0	40	1
::ffff:0:0/96	35	4
2002::/16	30	2
2001::/32	5	5
fc00::/7	3	13
::/96	1	3
fec0::/10	1	11
3ffe::/16	1	12

#### Authors' Addresses

Arifumi Matsumoto  
NTT NT Lab  
3-9-11 Midori-Cho  
Musashino-shi, Tokyo 180-8585  
Japan

Phone: +81 422 59 3334  
Email: arifumi@nttv6.net

Tomohiro Fujisaki  
NTT NT Lab  
3-9-11 Midori-Cho  
Musashino-shi, Tokyo 180-8585  
Japan

Phone: +81 422 59 7351  
Email: fujisaki@nttv6.net

Tim Chown  
University of Southampton  
Southampton, Hampshire SO17 1BJ  
United Kingdom

Email: tjc@ecs.soton.ac.uk

Network Working Group  
Internet-Draft  
Updates: 2460 (if approved)  
Intended status: Standards Track  
Expires: March 9, 2013

M. Eubanks  
AmericaFree.TV LLC  
P. Chimento  
Johns Hopkins University Applied  
Physics Laboratory  
M. Westerlund  
Ericsson  
September 5, 2012

UDP Checksums for Tunneled Packets  
draft-ietf-6man-udpchecksums-04

Abstract

This document provides an update of the Internet Protocol version 6 (IPv6) specification (RFC2460) to improve the performance of IPv6 in the use case when a tunnel protocol uses UDP with IPv6 to tunnel packets. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for suitable tunneling protocol where header information is protected on the "inner" packet being carried. This relaxation removes the overhead associated with the computation of UDP checksums on IPv6 packets used to carry tunnel protocols and thereby improves the efficiency of the traversal of firewalls and other network middleboxes by such protocols. We describe how the IPv6 UDP checksum requirement can be relaxed in the situation where the encapsulated packet itself contains a checksum, the limitations and risks of this approach, and defines restrictions on the use of this relaxation to mitigate these risks.

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 March 9, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Some Terminology . . . . .	4
2.1. Requirements Language . . . . .	4
3. Problem Statement . . . . .	4
4. Discussion . . . . .	4
5. The Zero-Checksum Update . . . . .	6
6. Additional Observations . . . . .	9
7. IANA Considerations . . . . .	10
8. Security Considerations . . . . .	10
9. Acknowledgements . . . . .	10
10. References . . . . .	10
10.1. Normative References . . . . .	10
10.2. Informative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

This work constitutes an update of the Internet Protocol Version 6 (IPv6) Specification [RFC2460], in the use case when a tunnel protocol uses UDP with IPv6 to tunnel packets. With the rapid growth of the Internet, tunneling protocols have become increasingly important to enable the deployment of new protocols. Tunneled protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, switches and end hosts on the global Internet for a new protocol is now measured in decades. At the same time, the increasing use of firewalls and other security related middleboxes means that truly new tunnel protocols, with new protocol numbers, are also unlikely to be deployable in a reasonable time frame, which has resulted in an increasing interest in and use of UDP-based tunneling protocols. In such protocols, there is an encapsulated "inner" packet, and the "outer" packet carrying the tunneled inner packet is a UDP packet, which can pass through firewalls and other middleboxes filtering that is a fact of life on the current Internet.

Tunnel endpoints may be routers or middleboxes aggregating traffic from a large number of tunnel users, therefore the computation of an additional checksum on the outer UDP packet, may be seen as an unwarranted burden on nodes that implement a tunneling protocol, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum on the IP packet itself, and the checksum on the outer UDP packet can be set to zero. However in IPv6 there is not a checksum on the IP packet and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a zero checksum. So, while sending a UDP packet with a zero checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. To improve support for IPv6 UDP tunnels, this document updates RFC 2460 to allow tunnel endpoints to use a zero UDP checksum under constrained situations (IPv6 tunnel transports that carry checksum-protected packets), following the considerations in [I-D.ietf-6man-udpzero].

Unicast UDP Usage Guidelines for Application Designers [RFC5405] should be consulted when reading this specification. It discusses both UDP tunnels (Section 3.1.3) and the usage of Checksums (Section 3.4).

While the origin of this specification is the problem raised by the draft titled "Automatic IP Multicast Without Explicit Tunnels", also known as "AMT," [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability. Since the first version of this document, the need for an efficient UDP tunneling mechanism has increased. Other IETF Working Groups, notably LISP [I-D.ietf-lisp] and Softwires

[RFC5619] have expressed a need to update the UDP checksum processing in RFC 2460. We therefore expect this update to be applicable in future to other tunneling protocols specified by these and other IETF Working Groups.

## 2. Some Terminology

For the remainder of this document, we discuss only IPv6, since this problem does not exist for IPv4. Therefore all reference to 'IP' should be understood as a reference to IPv6.

The document uses the terms "tunneling" and "tunneled" as adjectives when describing packets. When we refer to 'tunneling packets' we refer to the outer packet header that provides the tunneling function. When we refer to 'tunneled packets' we refer to the inner packet, i.e. the packet being carried in the tunnel.

### 2.1. 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].

## 3. Problem Statement

This document provides an update for the case where a tunnel protocol transports tunneled packets that already have a UDP header with a checksum, there is both a benefit and a cost to compute and check the UDP checksum of the outer (encapsulating) UDP transport header. In certain cases, where reducing the forwarding cost is important, such as for systems that perform the check in software, the cost may outweigh the benefit; this document describes a means to avoid that cost, in the case where there is an inner header with a checksum.

## 4. Discussion

IPv6 UDP Checksum Considerations [I-D.ietf-6man-udpzero] describes the issues related to allowing UDP over IPv6 to have a valid checksum of zero and is not repeated here.

Section 5.1 of [I-D.ietf-6man-udpzero], identifies 9 requirements that introduce constraints on the usage of a zero checksum for UDP over IPv6. This document is intended to satisfy these requirements.

[I-D.ietf-6man-udpzero] and mailing list discussions have noted there

is still the possibility of deep-inspection firewall devices or other middleboxes checking the UDP checksum field of the outer packet and thereby discarding the tunneling packets. This would be an issue also for any legacy IPv6 system that has not implemented this update to the IPv6 specification. In this case, the system (according to RFC 2460) will discard the zero-checksum UDP packets, and should log this as an error.

The below discuss how path errors can be detected and handled in an UDP tunneling protocol when the checksum protection is disabled. Note that other (non-tunneling) protocols may have different approaches, but these are not the topic of this update. We propose the following approach to handle this problem:

- o Context (i.e. tunneling state) should be established via application Protocol Data Units (PDUs) that are carried in checksummed UDP packets. That is, any control packets flowing between the tunnel endpoints should be protected by UDP checksums. The control packets can also contain any negotiation required to enable the endpoint/adapters to accept UDP packets with a zero checksum. The control packets may also carry any negotiation required to enable the endpoint/adapters to identify the set of ports that need to enable reception UDP datagrams with a zero checksum.
- o A system shall not set the UDP checksum to zero in packets that do not contain tunneled packets.
- o UDP keep-alive packets with checksum zero can be sent to validate paths, given that paths between tunnel endpoints can change and so middleboxes in the path may vary during the life of the association. Paths with middleboxes that are intolerant of a UDP checksum of zero will drop the keep-alives and the endpoints will discover that. Note that this need only be done per tunnel endpoint pair, not per tunnel context. Keep-alive traffic should include both packets with tunnel checksums and packets with checksums equal to zero to enable the remote end to distinguish between path failures and the blockage of packets with checksum equal to zero.
- o Corruption of the encapsulating IPv6 source address, destination address and/or the UDP source port, destination port fields : If the 9 restrictions in [I-D.ietf-6man-udpzero] are followed, the inner packets (tunneled packets) should be protected and run the usual (presumably small) risk of having undetected corruption(s). If tunneling protocol contexts contain (at a minimum) source and destination IP addresses and source and destination ports, there are 16 possible corruption outcomes. We note that these outcomes

are not equally likely. The possible corruption outcomes may be:

- \* Half of the 16 possible corruption combinations have a corrupted destination address. If the incorrect destination is reached and the node doesn't have an application for the destination port, the packet will be dropped. If the application at the incorrect destination is the same tunneling protocol and if it has a matching context (which can be assumed to be a very low probability event) the inner packet will be decapsulated and forwarded. Application developers should verify the context of the packets they receive using UDP, as described in [RFC5405]. Applications that verify the context of a datagram are expected to have a high probability of discarding corrupted data. [I-D.ietf-6man-udpzero] presents examples of cases where corruption can inadvertently impact application state.
- \* Half of the 8 possible corruption combinations with a correct destination address have a corrupted source address. If the tunnel contexts contain all elements of the address-port 4-tuple, then the likelihood is that this corruption will be detected.
- \* Of the remaining 4 possibilities, with valid source and destination IPv6 addresses, 1 has all 4 fields valid, the other three have one or both ports corrupted. Again, if the tunneling endpoint context contains sufficient information, these errors should be detected with high probability.
- o Corruption of source-fragmented encapsulating packets: In this case, a tunneling protocol may reassemble fragments associated with the wrong context at the right tunnel endpoint, or it may reassemble fragments associated with a context at the wrong tunnel endpoint, or corrupted fragments may be reassembled at the right context at the right tunnel endpoint. In each of these cases, the IPv6 length of the encapsulating header may be checked (though [I-D.ietf-6man-udpzero] points out the weakness in this check). In addition, if the encapsulated packet is protected by a transport (or other) checksum, these errors can be detected (with some probability).

While they do not guarantee correctness, these mechanisms can reduce the risks of relaxing the UDP checksum requirement for IPv6.

## 5. The Zero-Checksum Update

This specification updates IPv6 to allow a UDP checksum of zero for



the outer encapsulating packet of a tunneling protocol. UDP endpoints that implement this update MUST change their behavior for any destination port explicitly configured for zero checksum and not discard UDP packets received with a checksum value of zero on the outer packet. When this is done, it requires the constraints in Section 5.1 of [I-D.ietf-6man-udpzero].

Specifically, the text in [RFC2460] Section 8.1, 4th bullet is updated. We refer to the following text:

"Unlike IPv4, when UDP packets are originated by an IPv6 node, 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."

This item should be taken out of the bullet list and should be replaced by:

Whenever originating a UDP packet, an IPv6 node SHOULD 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 SHOULD discard UDP packets containing a zero checksum, and SHOULD log the error. However, some protocols, such as tunneling protocols that use UDP as a tunnel encapsulation, MAY omit computing the UDP checksum of the encapsulating UDP header and set it to zero, subject to the constraints described in RFCXXXX. In cases where the encapsulating protocol uses a zero checksum for UDP, the receiver of packets sent to a port enabled to receive zero-checksum packets MUST NOT discard packets solely for having a UDP checksum of zero. Note that these constraints apply only to encapsulating protocols that omit calculating the UDP checksum and set it to zero. An encapsulating protocol can always choose to compute the UDP checksum, in which case, its behavior is not updated and uses the method specified in RFC2460.

1. IPv6 protocol stack implementations SHOULD NOT by default allow the new method. The default node receiver behavior MUST discard all IPv6 packets carrying UDP packets with a zero checksum.
2. Implementations MUST provide a way to signal the set of ports that will be enabled to receive UDP datagrams with a zero

checksum. An IPv6 node that enables reception of UDP packets with a zero-checksum, MUST enable this only for a specific port or port-range. This may be implemented via a socket API call, or similar mechanism.

3. RFC 2460 specifies that IPv6 nodes should log UDP datagrams with a zero-checksum. A port for which zero-checksum has been enabled MUST NOT log zero-checksum datagrams for that reason (of course, there might be other reasons to log such packets).
4. A stack may separately identify UDP datagrams that are discarded with a zero checksum. It SHOULD NOT add these to the standard log, since the endpoint has not been verified.
5. UDP Tunnels that encapsulate IP MAY rely on the inner packet integrity checks provided that the tunnel will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel MUST provide an additional integrity verification mechanism. An integrity mechanism is always recommended at the tunnel layer to ensure that corruption rates of the inner most packet are not increased.
6. Tunnels that encapsulate Non-IP packets MUST have a CRC or other mechanism for checking packet integrity, unless the Non-IP packet specifically is designed for transmission over lower layers that do not provide any packet integrity guarantee. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.
7. UDP applications that support use of a zero-checksum, SHOULD NOT rely upon correct reception of the IP and UDP protocol information (including the length of the packet) when decoding and processing the packet payload. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.
8. If a method proposes recursive tunnels, it MUST provide guidance that is appropriate for all use-cases. Restrictions may be needed to the use of a tunnel encapsulations and the use of recursive tunnels (e.g. Necessary when the endpoint is not verified).
9. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum MUST provide appropriate checks

concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and port numbers in the ICMPv6 message body).

Middleboxes MUST allow IPv6 packets with UDP checksum equal to zero to pass. Implementations of middleboxes MAY allow configuration of specific port ranges for which a zero UDP checksum is valid and may drop IPv6 UDP packets outside those ranges.

The path between tunnel endpoints can change, thus also the middleboxes in the path may vary during the life of the association. Paths with middleboxes that are intolerant of a UDP checksum of zero will drop any keep-alives sent to validate the path using checksum zero and the endpoints will discover that. Therefore keep-alive traffic SHOULD include both packets with tunnel checksums and packets with checksums equal to zero to enable the remote end to distinguish between path failures and the blockage of packets with checksum equal to zero. Note that path validation need only be done per tunnel endpoint pair, not per tunnel context.

RFC-Editor Note: Please replace RFCXXXX above with the RFC number this specification receives and remove this note.

## 6. Additional Observations

The existence of this issue among a significant number of protocols being developed in the IETF motivates this specified change. The authors would also like to make the following observations:

- o An empirically-based analysis of the probabilities of packet corruptions (with or without checksums) has not (to our knowledge) been conducted since about 2000. It is now 2012. We strongly suggest that an empirical study is in order, along with an extensive analysis of IPv6 header corruption probabilities.
- o A key cause to the increased usage of UDP in tunneling is the lack of protocol support in middleboxes. Specifically, new protocols, such as LISP [I-D.ietf-lisp], prefer to use UDP tunnels to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If this were not the case, the use of UDP-lite [RFC3828] might become more viable for some (but not necessarily all) tunneling protocols.

- o Another issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as is the TCP checksum for TCP flows). Protocols that do not use a pseudo-header approach to computing a checksum or CRC have essentially no protection from mis-delivered packets.

## 7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

## 8. Security Considerations

It requires less work to generate zero-checksum attack packets than ones with full UDP checksums. However, this does not lead to any significant new vulnerabilities as checksums are not a security measure and can be easily generated by any attacker, as properly configured tunnels should check the validity of the inner packet and perform any needed security checks, regardless of the checksum status, and finally as most attacks are generated from compromised hosts which automatically create checksummed packets (in other words, it would generally be more, not less, effort for most attackers to generate zero UDP checksums on the host).

## 9. Acknowledgements

We would like to thank Brian Haberman and Gorrry Fairhurst for discussions and reviews.

## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

- [RFC5619] Yamamoto, S., Williams, C., Yokota, H., and F. Parent,  
"Softwire Security Analysis and Requirements", RFC 5619,  
August 2009.

## 10.2. Informative References

- [I-D.ietf-6man-udpzero]  
Fairhurst, G. and M. Westerlund, "IPv6 UDP Checksum  
Considerations", draft-ietf-6man-udpzero-06 (work in  
progress), June 2012.
- [I-D.ietf-lisp]  
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis,  
"Locator/ID Separation Protocol (LISP)",  
draft-ietf-lisp-23 (work in progress), May 2012.
- [I-D.ietf-mboned-auto-multicast]  
Bumgardner, G., "Automatic Multicast Tunneling",  
draft-ietf-mboned-auto-multicast-14 (work in progress),  
June 2012.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines  
for Application Designers", BCP 145, RFC 5405,  
November 2008.

## Authors' Addresses

Marshall Eubanks  
AmericaFree.TV LLC  
P.O. Box 141  
Clifton, Virginia 20124  
USA

Phone: +1-703-501-4376  
Fax:  
Email: marshall.eubanks@gmail.com

P.F. Chimento  
Johns Hopkins University Applied Physics Laboratory  
11100 Johns Hopkins Road  
Laurel, MD 20723  
USA

Phone: +1-443-778-1743  
Fax:  
Email: Philip.Chimento@jhuapl.edu  
URI:

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

Phone: +46 10 714 82 87  
Email: magnus.westerlund@ericsson.com



Network Working Group  
Internet-Draft  
Updates: 2460 (if approved)  
Intended status: Standards Track  
Expires: August 25, 2013

M. Eubanks  
AmericaFree.TV LLC  
P. Chimento  
Johns Hopkins University Applied  
Physics Laboratory  
M. Westerlund  
Ericsson  
February 21, 2013

IPv6 and UDP Checksums for Tunneled Packets  
draft-ietf-6man-udpchecksums-08

Abstract

This document provides an update of the Internet Protocol version 6 (IPv6) specification (RFC2460) to improve the performance in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for any suitable tunnel protocol where header information is protected on the "inner" packet being carried. This relaxation removes the overhead associated with the computation of UDP checksums on IPv6 packets used to carry tunnel protocols. The specification describes how the IPv6 UDP checksum requirement can be relaxed for the situation where the encapsulated packet itself contains a checksum. The limitations and risks of this approach are described, and restrictions specified on the use of the method.

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 August 25, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the



document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Some Terminology . . . . .	4
2.1. Requirements Language . . . . .	4
3. Problem Statement . . . . .	4
4. Discussion . . . . .	4
4.1. Analysis of Corruption in Tunnel Context . . . . .	5
4.2. Limitation to Tunnel Protocols . . . . .	7
4.3. Middleboxes . . . . .	8
5. The Zero-Checksum Update . . . . .	8
6. Additional Observations . . . . .	10
7. IANA Considerations . . . . .	10
8. Security Considerations . . . . .	10
9. Acknowledgements . . . . .	11
10. References . . . . .	11
10.1. Normative References . . . . .	11
10.2. Informative References . . . . .	12
Authors' Addresses . . . . .	12

## 1. Introduction

This work constitutes an update of the Internet Protocol Version 6 (IPv6) Specification [RFC2460], in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. With the rapid growth of the Internet, tunnel protocols have become increasingly important to enable the deployment of new protocols. Tunnel protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, middleboxes and hosts on the global Internet for a new protocol is now measured in decades. At the same time, the increasing use of firewalls and other security-related middleboxes means that truly new tunnel protocols, with new protocol numbers, are also unlikely to be deployable in a reasonable time frame, which has resulted in an increasing interest in and use of UDP-based tunnel protocols. In such protocols, there is an encapsulated "inner" packet, and the "outer" packet carrying the tunneled inner packet is a UDP packet, which can pass through firewalls and other middleboxes that perform filtering that is a fact of life on the current Internet.

Tunnel endpoints may be routers or middleboxes aggregating traffic from a number of tunnel users, therefore the computation of an additional checksum on the outer UDP packet may be seen as an unwarranted burden on nodes that implement a tunnel protocol, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum over the IP packet header, and the checksum on the outer UDP packet may be set to zero. However in IPv6 there is no checksum in the IP header and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a zero checksum. So, while sending a UDP datagram with a zero checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. To improve support for IPv6 UDP tunnels, this document updates RFC 2460 to allow endpoints to use a zero UDP checksum under constrained situations (primarily IPv6 tunnel transports that carry checksum-protected packets), following the applicability statements and constraints in [I-D.ietf-6man-udpzero].

"Unicast UDP Usage Guidelines for Application Designers" [RFC5405] should be consulted when reading this specification. It discusses both UDP tunnels (Section 3.1.3) and the usage of checksums (Section 3.4).

While the origin of this specification is the problem raised by the draft titled "Automatic Multicast Tunnels", also known as "AMT" [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability. Since the first version of this document, the need for an efficient UDP tunneling mechanism has increased. Other IETF Working Groups, notably LISP [RFC6830] and Softwires [RFC5619] have

expressed a need to update the UDP checksum processing in RFC 2460. We therefore expect this update to be applicable in the future to other tunnel protocols specified by these and other IETF Working Groups.

## 2. Some Terminology

This document discusses only IPv6, since this problem does not exist for IPv4. Therefore all reference to 'IP' should be understood as a reference to IPv6.

The document uses the terms "tunneling" and "tunneled" as adjectives when describing packets. When we refer to 'tunneling packets' we refer to the outer packet header that provides the tunneling function. When we refer to 'tunneled packets' we refer to the inner packet, i.e., the packet being carried in the tunnel.

### 2.1. 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].

## 3. Problem Statement

When using tunnel protocols based on UDP, there can be both a benefit and a cost to computing and checking the UDP checksum of the outer (encapsulating) UDP transport header. In certain cases, reducing the forwarding cost is important, e.g., for nodes that perform the checksum in software the cost may outweigh the benefit. This document provides an update for usage of the UDP checksum with IPv6. The update is specified for use by a tunnel protocol that transports packets that are themselves protected by a checksum.

## 4. Discussion

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] describes issues related to allowing UDP over IPv6 to have a valid zero UDP checksum and is the starting point for this discussion. Sections 4 and 5 of [I-D.ietf-6man-udpzero], respectively identify node implementation and usage requirements for datagrams sent and received with a zero UDP checksum. These introduce constraints on the usage of a zero checksum for UDP over IPv6. The remainder of this section analyses the use of general tunnels and motivates why tunnel protocols are

being permitted to use the method described in this update. Issues with middleboxes are also discussed.

#### 4.1. Analysis of Corruption in Tunnel Context

This section analyzes the impact of the different corruption modes in the context of a tunnel protocol. It indicates what needs to be considered by the designer and user of a tunnel protocol to be robust. It also summarizes why use of a zero UDP checksum is thought to be safe for deployment.

1. Context (i.e., tunneling state) should be established by exchanging application Protocol Data Units (PDUs) carried in checksummed UDP datagrams or by other protocols with integrity protection against corruption. These control packets should also carry any negotiation required to enable the tunnel endpoint to accept UDP datagrams with a zero checksum and identify the set of ports that are used. It is important that the control traffic is robust against corruption because undetected errors can lead to long-lived and significant failures that may affect much more than the single packet that was corrupted.
2. Keep-alive datagrams with a zero UDP checksum should be sent to validate the network path, because the path between tunnel endpoints can change and therefore the set of middleboxes along the path may change during the life of an association. Paths with middleboxes that drop datagrams with a zero UDP checksum will drop these keep-alives. To enable the tunnel endpoints to discover and react to this behavior in a timely way, the keep-alive traffic should include datagrams with a non-zero checksum and datagrams with a zero checksum.
3. Receivers should attempt to detect corruption of the address information in an encapsulating packet. A robust tunnel protocol should track tunnel context based on the 5-tuple (tunneled protocol number, IPv6 source address, IPv6 destination address, UDP source port, UDP destination port). A corrupted datagram that arrives at a destination may be filtered based on this check.
  - \* If the datagram header matches the 5-tuple and the node has the zero checksum enabled for this port, the payload is matched to the wrong context. The tunneled packet will then be decapsulated and forwarded by the tunnel egress.
  - \* If a corrupted datagram matches a different 5-tuple and the zero checksum was enabled for the port, the datagram payload is matched to the wrong context, and may be processed by the

wrong tunnel protocol, if it also passes the verification of that protocol.

- \* If a corrupted datagram matches a 5-tuple and the zero checksum has not been enabled for this port, the datagram will be discarded.

When only the source information is corrupted, the datagram could arrive at the intended applications/protocol, which will process the datagram and try to match it against an existing tunnel context. The likelihood that a corrupted packet enters a valid context is reduced when the protocol restricts processing to only the source addresses with established contexts. When both source and destination fields are corrupted, this increases the likelihood of failing to match a context, with the exception of errors replacing one packet header with another one. In this case, it is possible that both packets are tunneled and therefore the corrupted packet could match a previously defined context.

4. Receivers should attempt to detect corruption of source-fragmented encapsulating packets. A tunnel protocol may reassemble fragments associated with the wrong context at the right tunnel endpoint, or it may reassemble fragments associated with a context at the wrong tunnel endpoint, or corrupted fragments may be reassembled at the right context at the right tunnel endpoint. In each of these cases, the IPv6 length of the encapsulating header may be checked (though [I-D.ietf-6man-udpzero] points out the weakness in this check). In addition, if the encapsulated packet is protected by a transport (or other) checksum, these errors can be detected (with some probability).
5. Tunnel protocols using UDP have some advantages that reduce the risk for a corrupted tunnel packet reaching a destination that will receive it, compared to other applications. This results from processing by the network of the inner (tunneled) packet after being forwarded from the tunnel egress using a wrong context:
  - \* A tunneled packet may be forwarded to the wrong address domain, for example, a private address domain where the inner packet's address is not routable, or may fail a source address check, such as Unicast Reverse Path Forwarding [RFC2827], resulting in the packet being dropped.
  - \* The destination address of a tunneled packet may not at all be reachable from the delivered domain. For example, an Ethernet

frame where the destination MAC address is not present on the LAN segment that was reached.

- \* The type of the tunneled packet may prevent delivery. For example, an attempt to interpret an IP packet payload as an Ethernet frame, would likely to result in the packet being dropped as invalid.
- \* The tunneled packet checksum or integrity mechanism may detect corruption of the inner packet caused at the same time as corruption to the outer packet header. The resulting packet would likely be dropped as invalid.

These checks each significantly reduce the likelihood that a corrupted inner tunneled packet is finally delivered to a protocol listener that can be affected by the packet. While the methods do not guarantee correctness, they can reduce the risk of relaxing the UDP checksum requirement for a tunnel application using IPv6.

#### 4.2. Limitation to Tunnel Protocols

This document describes the applicability of using a zero UDP checksum to support tunnel protocols. There are good motivations behind this and the arguments are provided here.

- o Tunnels carry inner packets that have their own semantics, which may make any corruption less likely to reach the indicated destination and be accepted as a valid packet. This is true for IP packets with the addition of verification that can be made by the tunnel protocol, the network processing of the inner packet headers as discussed above, and verification of the inner packet checksums. Non-IP inner packets are likely to be subject to similar effects that may reduce the likelihood of a misdelivered packet being delivered to a protocol listener that can be affected by the packet.
- o Protocols that directly consume the payload must have sufficient robustness against misdelivered packets from any context, including the ones that are corrupted in tunnels and any other usage of the zero checksum. This will require an integrity mechanism. Using a standard UDP checksum reduces the computational load in the receiver to verify this mechanism.
- o The design for stateful protocols or protocols where corruption causes cascade effects requires extra care. In tunnel usage, each encapsulating packet provides only a transport mechanism from tunnel ingress to tunnel egress. A corruption will commonly only affect the single tunneled packet, not the established protocol

state. One common effect is that the inner packet flow will only see a corruption and misdelivery of the outer packet as a lost packet.

- o Some non-tunnel protocols operate with general servers that do not know the source from which they will receive a packet. In such applications, a zero UDP checksum is unsuitable because there is a need to provide the first level of verification that the packet was intended for the receiving server. A verification prevents the server from processing the datagram payload and without this it may spend significant resources processing the packet, including sending replies or error messages.

Tunnel protocols that encapsulate IP will generally be safe for deployment, since all IPv4 and IPv6 packets include at least one checksum at either the network or transport layer. The network delivery of the inner packet will then further reduce the effects of corruption. Tunnel protocols carrying non-IP packets may offer equivalent protection when the non-IP networks reduce the risk of misdelivery to applications. However, there is a need for further analysis to understand the implications of misdelivery of corrupted packets for that each non-IP protocol. The analysis above suggests that non-tunnel protocols can be expected to have significantly more cases where a zero checksum would result in misdelivery or negative side-effects.

One unfortunate side-effect of increased use of a zero-checksum is that it also increases the likelihood of acceptance when a datagram with a zero UDP checksum is misdelivered. This requires all tunnel protocols using this method to be designed to be robust to misdelivery.

#### 4.3. Middleboxes

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] notes that middleboxes that conform to RFC 2460 will discard datagrams with a zero UDP checksum and should log this as an error. Tunnel protocols intending to use a zero UDP checksum need to ensure that they have defined a method for handling cases when a middlebox prevents the path between the tunnel ingress and egress from supporting transmission of datagrams with a zero UDP checksum.

#### 5. The Zero-Checksum Update

This specification updates IPv6 to allow a zero UDP checksum in the outer encapsulating datagram of a tunnel protocol. UDP endpoints

that implement this update MUST follow the node requirements in "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

The following text in [RFC2460] Section 8.1, 4th bullet should be deleted:

"Unlike IPv4, when UDP packets are originated by an IPv6 node, 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."

This text should be replaced by:

An IPv6 node associates a mode with each used UDP port (for sending and/or receiving packets).

Whenever originating a UDP packet for a port in the default mode, 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 as specified in [RFC2460]. IPv6 receivers MUST by default discard UDP packets containing a zero checksum, and SHOULD log the error.

As an alternative, certain protocols that use UDP as a tunnel encapsulation, MAY enable the zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing the zero-checksum mode MUST follow the node requirements specified in Section 4 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

Any protocol that enables the zero-checksum mode for a specific port or ports MUST follow the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

Middleboxes supporting IPv6 MUST follow requirements 9, 10 and 11 of the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].



## 6. Additional Observations

This update was motivated by the existence of a number of protocols being developed in the IETF that are expected to benefit from the change. The following observations are made:

- o An empirically-based analysis of the probabilities of packet corruption (with or without checksums) has not (to our knowledge) been conducted since about 2000. At the time of publication, it is now 2012. We strongly suggest a new empirical study, along with an extensive analysis of the corruption probabilities of the IPv6 header. This can potentially allow revising the recommendations in this document.
- o A key motivation for the increase in use of UDP in tunneling is a lack of protocol support in middleboxes. Specifically, new protocols, such as LISP [RFC6830], may prefer to use UDP tunnels to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If middleboxes were updated to support UDP-Lite [RFC3828], UDP-Lite would provide better protection than offered by this update. This may be suited to a variety of applications and would be expected to be preferred over this method for many tunnel protocols.
- o Another issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as is the TCP checksum for TCP flows). Protocols that do not use a pseudo-header approach to computing a checksum or CRC have essentially no protection from misdelivered packets.

## 7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

## 8. Security Considerations

Less work is required to generate an attack using a zero UDP checksum than one using a standard full UDP checksum. However, this does not lead to significant new vulnerabilities because checksums are not a security measure and can be easily generated by any attacker.

In general any user of zero UDP checksums should apply the checks and context verification that are possible to minimize the risk of

unintended traffic to reach a particular context. This will however not protect against an intended attack that create packet with the correct information. Source address validation can help prevent injection of traffic into contexts by an attacker.

Depending on the hardware design, the processing requirements may differ for tunnels that have a zero UDP checksum and those that calculate a checksum. This processing overhead may need to be considered when deciding whether to enable a tunnel and to determine an acceptable rate for transmission. This can become a security risk for designs that can handle a significantly larger number of packets with zero UDP checksums compared to datagrams with a non-zero checksum, such as tunnel egress. An attacker could attempt to inject non-zero checksummed UDP packets into a tunnel forwarding zero checksum UDP packets and cause overload in the processing of the non-zero checksums, e.g. if this happens in a routers slow path. Protection mechanisms should therefore be employed when this threat exists. Protection may include source address filtering to prevent an attacker injecting traffic, as well as throttling the amount of non-zero checksum traffic. The latter may impact the function of the tunnel protocol.

## 9. Acknowledgements

We would like to thank Brian Haberman, Dan Wing, Joel Halpern, David Waltermire, J.W. Atwood, Peter Yee, Joe Touch and the IESG of 2012 for discussions and reviews. Gorrry Fairhurst has been very diligent in reviewing and help ensuring alignment between this document and [I-D.ietf-6man-udpzero].

## 10. References

### 10.1. Normative References

- [I-D.ietf-6man-udpzero]  
Fairhurst, G. and M. Westerlund, "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums", draft-ietf-6man-udpzero-10 (work in progress), January 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

## 10.2. Informative References

- [I-D.ietf-mboned-auto-multicast]  
Bumgardner, G., "Automatic Multicast Tunneling",  
draft-ietf-mboned-auto-multicast-14 (work in progress),  
June 2012.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:  
Defeating Denial of Service Attacks which employ IP Source  
Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and  
G. Fairhurst, "The Lightweight User Datagram Protocol  
(UDP-Lite)", RFC 3828, July 2004.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines  
for Application Designers", BCP 145, RFC 5405,  
November 2008.
- [RFC5619] Yamamoto, S., Williams, C., Yokota, H., and F. Parent,  
"Software Security Analysis and Requirements", RFC 5619,  
August 2009.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The  
Locator/ID Separation Protocol (LISP)", RFC 6830,  
January 2013.

## Authors' Addresses

Marshall Eubanks  
AmericaFree.TV LLC  
P.O. Box 141  
Clifton, Virginia 20124  
USA

Phone: +1-703-501-4376  
Fax:  
Email: marshall.eubanks@gmail.com

P.F. Chimento  
Johns Hopkins University Applied Physics Laboratory  
11100 Johns Hopkins Road  
Laurel, MD 20723  
USA

Phone: +1-443-778-1743  
Email: Philip.Chimento@jhuapl.edu

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

Phone: +46 10 714 82 87  
Email: magnus.westerlund@ericsson.com



Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: December 20, 2012

G. Fairhurst  
University of Aberdeen  
M. Westerlund  
Ericsson  
June 20, 2012

IPv6 UDP Checksum Considerations  
draft-ietf-6man-udpzero-06

Abstract

This document examines the role of the UDP transport checksum when used with IPv6, as defined in RFC2460. It presents a summary of the trade-offs for evaluating the safety of updating RFC 2460 to permit an IPv6 UDP endpoint to use a zero value in the checksum field as an indication that no checksum is present. This method is compared with some other possibilities. The document also describes the issues and design principles that need to be considered when UDP is used with IPv6 to support tunnel encapsulations. It concludes that UDP with a zero checksum in IPv6 can safely be used for this purpose, provided that this usage is governed by a set of constraints.

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 December 20, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Document Structure . . . . .	3
1.2. Background . . . . .	4
1.2.1. The Role of a Transport Endpoint . . . . .	4
1.2.2. The UDP Checksum . . . . .	4
1.2.3. Differences between IPv6 and IPv4 . . . . .	6
1.3. Use of UDP Tunnels . . . . .	6
1.3.1. Motivation for new approaches . . . . .	6
1.3.2. Reducing forwarding cost . . . . .	7
1.3.3. Need to inspect the entire packet . . . . .	8
1.3.4. Interactions with middleboxes . . . . .	8
1.3.5. Support for load balancing . . . . .	8
2. Standards-Track Transports . . . . .	9
2.1. UDP with Standard Checksum . . . . .	9
2.2. UDP-Lite . . . . .	9
2.2.1. Using UDP-Lite as a Tunnel Encapsulation . . . . .	10
2.3. General Tunnel Encapsulations . . . . .	10
3. Issues Requiring Consideration . . . . .	11
3.1. Effect of packet modification in the network . . . . .	11
3.1.1. Corruption of the destination IP address . . . . .	12
3.1.2. Corruption of the source IP address . . . . .	13
3.1.3. Corruption of Port Information . . . . .	14
3.1.4. Delivery to an unexpected port . . . . .	14
3.1.5. Corruption of Fragmentation Information . . . . .	15
3.2. Validating the network path . . . . .	17
3.3. Applicability of method . . . . .	18
3.4. Impact on non-supporting devices or applications . . . . .	19
4. Evaluation of proposal to update RFC 2460 to support zero checksum . . . . .	19
4.1. Alternatives to the Standard Checksum . . . . .	19
4.2. Comparison . . . . .	21
4.2.1. Middlebox Traversal . . . . .	21
4.2.2. Load Balancing . . . . .	22
4.2.3. Ingress and Egress Performance Implications . . . . .	22
4.2.4. Deployability . . . . .	22
4.2.5. Corruption Detection Strength . . . . .	23
4.2.6. Comparison Summary . . . . .	23
5. Requirements on the specification of transported protocols . . . . .	25
5.1. Constraints required on usage of a zero checksum . . . . .	25
6. Summary . . . . .	27
7. Acknowledgements . . . . .	28
8. IANA Considerations . . . . .	28
9. Security Considerations . . . . .	28
10. References . . . . .	28
10.1. Normative References . . . . .	28
10.2. Informative References . . . . .	29
Appendix A. Document Change History . . . . .	30

## 1. Introduction

The User Datagram Protocol (UDP) [RFC0768] transport is defined for the Internet Protocol (IPv4) [RFC0791] and is defined in Internet Protocol, Version 6 (IPv6) [RFC2460] for IPv6 hosts and routers. The UDP transport protocol has a minimal set of features. This limited set has enabled a wide range of applications to use UDP, but these application do need to provide many important transport functions on top of UDP. The UDP Usage Guidelines [RFC5405] provides overall guidance for application designers, including the use of UDP to support tunneling. The key difference between UDP usage with IPv4 and IPv6 is that IPv6 mandates use of the UDP checksum, i.e. a non-zero value, due to the lack of an IPv6 header checksum.

The lack of a possibility to use UDP with a zero-checksum in IPv6 has been observed as a real problem for certain classes of application, primarily tunnel applications. This class of application has been deployed with a zero checksum using IPv4. The design of IPv6 raises different issues when considering the safety of using a zero checksum for UDP with IPv6. These issues can significantly affect applications, both when an endpoint is the intended user and when an innocent bystander (received by a different endpoint to that intended). The document examines these issues and compares the strengths and weaknesses of a number of proposed solutions. This analysis presents a set of issues that must be considered and mitigated to be able to safely deploy UDP with a zero checksum over IPv6. The provided comparison of methods is expected to also be useful when considering applications that have different goals from the ones that initiated the writing of this document, especially the use of already standardized methods.

The analysis concludes that using UDP with a zero checksum is the best method of the proposed alternatives to meet the goals for certain tunnel applications. Unfortunately, this usage is expected to have some deployment issues related to middleboxes, limiting the usability more than desired in the currently deployed internet. However, this limitation will be largest initially and will reduce as updates for support of UDP zero checksum for IPv6 are provided to middleboxes. The document therefore derives a set of constraints required to ensure safe deployment of zero checksum in UDP. It also identifies some issues that require future consideration and possibly additional research.

### 1.1. Document Structure

Section 1 provides a background to key issues, and introduces the use of UDP as a tunnel transport protocol.

Section 2 describes a set of standards-track datagram transport protocols that may be used to support tunnels.

Section 3 discusses issues with a zero checksum in UDP for IPv6. It considers the impact of corruption, the need for validation of the



path and when it is suitable to use a zero checksum.

Section 4 evaluates a set of proposals to update the UDP transport behaviour and other alternatives intended to improve support for tunnel protocols. It focuses on a proposal to allow a zero checksum for this use-case with IPv6 and assesses the trade-offs that would arise.

Section 5.1 lists the constraints perceived for safe deployment of zero-checksum.

Section 6 provides the recommendations for standardization of zero-checksum with a summary of the findings and notes remaining issues needing future work.

## 1.2. Background

This section provides a background on topics relevant to the following discussion.

### 1.2.1. The Role of a Transport Endpoint

An Internet transport endpoint should concern itself with the following issues:

- o Protection of the endpoint transport state from unnecessary extra state (e.g. Invalid state from rogue packets).
- o Protection of the endpoint transport state from corruption of internal state.
- o Pre-filtering by the endpoint of erroneous data, to protect the transport from unnecessary processing and from corruption that it can not itself reject.
- o Pre-filtering of incorrectly addressed destination packets, before responding to a source address.

### 1.2.2. The UDP Checksum

UDP, as defined in [RFC0768], supports two checksum behaviours when used with IPv4. The normal behaviour is for the sender to calculate a checksum over a block of data that includes a pseudo header and the UDP datagram payload. The UDP header includes a 16-bit one's complement checksum that provides a statistical guarantee that the payload was not corrupted in transit. This also allows a receiver to verify that the endpoint was the intended destination of the datagram, because the transport pseudo header covers the IP addresses, port numbers, transport payload length, and Next Header/Protocol value corresponding to the UDP transport protocol [RFC1071]. The length field verifies that the datagram is not truncated or padded. The checksum therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that were sent. Although the IPv4 UDP [RFC0768] checksum may be disabled, applications are recommended to enable UDP checksums [RFC5405].

The network-layer fields that are validated by a transport checksum are:

- o Endpoint IP source address (always included in the pseudo header of the checksum)
- o Endpoint IP destination address (always included in the pseudo header of the checksum)
- o Upper layer payload type (always included in the pseudo header of the checksum)
- o IP length of payload (always included in the pseudo header of the checksum)
- o Length of the network layer extension headers (i.e. by correct position of the checksum bytes)

The transport-layer fields that are validated by a transport checksum are:

- o Transport demultiplexing, i.e. ports (always included in the checksum)
- o Transport payload size (always included in the checksum)

Transport endpoints also need to verify the correctness of reassembly of any fragmented datagram. For UDP, this is normally provided as a part of the integrity check. Disabling the IPv4 checksum prevents this check. A lack of the UDP header and checksum in fragments can lead to issues in a translator or middlebox. For example, many IPv4 Network Address Translators, NATs, rely on port numbers to find the mappings, packet fragments do not carry port numbers, so fragments get dropped. IP/ICMP Translation Algorithm [RFC6145] provides some guidance on the processing of fragmented IPv4 UDP datagrams that do not carry a UDP checksum.

IPv4 UDP checksum control is often a kernel-wide configuration control (e.g. In Linux and BSD), rather than a per socket call. There are also Networking Interface Cards (NICs) that automatically calculate TCP [RFC0793] and UDP checksums on transmission when a checksum of zero is sent to the NIC, using a method known as checksum offloading.

### 1.2.3. Differences between IPv6 and IPv4

IPv6 does not provide a network-layer integrity check. The removal of the header checksum from the IPv6 specification released routers from a need to update a network-layer checksum for each router hop as the IPv6 Hop Count is changed (in contrast to the checksum update needed when an IPv4 router modifies the Time-To-Live (TTL)).

The IP header checksum calculation was seen as redundant for most traffic (with UDP or TCP checksums enabled), and people wanted to avoid this extra processing. However, there was concern that the removal of the IP header checksum in IPv6 combined with a UDP checksum set to zero would lessen the protection of the source/destination IP addresses and result in a significant (a multiplier of ~32,000) increase in the number of times that a UDP packet was accidentally delivered to the wrong destination address and/or apparently sourced from the wrong source address. This would have had implications on the detectability of mis-delivery of a packet to an incorrect endpoint/socket, and the robustness of the Internet infrastructure. The use of the UDP checksum is therefore required [RFC2460] when endpoint applications transmit UDP datagrams over IPv6.

### 1.3. Use of UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol, where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by the middleboxes that may exist along the path, because many middleboxes support transmission using UDP. In this use, the receiving endpoint decapsulates the UDP datagrams and forwards the original packets contained in the payload [RFC5405]. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks.

#### 1.3.1. Motivation for new approaches

A number of tunnel encapsulations deployed over IPv4 have used the UDP transport with a zero checksum. Users of these protocols expect a similar solution for IPv6.

A number of tunnel protocols are also currently being defined (e.g. Automated Multicast Tunnels, AMT [I-D.ietf-mboned-auto-multicast], and the Locator/Identifier Separation Protocol, LISP [LISP]). These protocols have proposed an update to IPv6 UDP checksum processing. These tunnel protocols could benefit from simpler checksum processing for various reasons:

- o Reducing forwarding costs, motivated by redundancy present in the encapsulated packet header, since in tunnel encapsulations, payload integrity and length verification may be provided by higher layer encapsulations (often using the IPv4, UDP, UDP-Lite, or TCP checksums).
- o Eliminating a need to access the entire packet when forwarding the packet by a tunnel endpoint.
- o Enhancing ability to traverse middleboxes, especially Network Address Translators, NATs.
- o A desire to use the port number space to enable load-sharing.

#### 1.3.2. Reducing forwarding cost

It is a common requirement to terminate a large number of tunnels on a single router/host. Processing per tunnel concerns both state (memory requirements) and per-packet processing costs.

Automatic IP Multicast Tunneling, known as AMT [I-D.ietf-mboned-auto-multicast] currently specifies UDP as the transport protocol for packets carrying tunneled IP multicast packets. The current specification for AMT requires that the UDP checksum in the outer packet header should be 0 (see Section 6.6 of [I-D.ietf-mboned-auto-multicast]). It argues that the computation of an additional checksum, when an inner packet is already adequately protected, is an unwarranted burden on nodes implementing lightweight tunneling protocols. The AMT protocol needs to replicate a multicast packet to each gateway tunnel. In this case, the outer IP addresses are different for each tunnel and therefore require a different pseudo header to be built for each UDP replicated encapsulation.

The argument concerning redundant processing costs is valid regarding the integrity of a tunneled packet. In some architectures (e.g. PC-based routers), other mechanisms may also significantly reduce checksum processing costs: There are implementations that have optimised checksum processing algorithms, including the use of checksum-offloading. This processing is readily available for IPv4 packets at high line rates. Such processing may be anticipated for IPv6 endpoints, allowing receivers to reject corrupted packets without further processing. However, there are certain classes of tunnel end-points where this off-loading is not available and

unlikely to become available in the near future.

#### 1.3.3. Need to inspect the entire packet

The currently-deployed hardware in many routers uses a fast-path processing that only provides the first  $n$  bytes of a packet to the forwarding engine, where typically  $n \leq 128$ . This prevents fast processing of a transport checksum over an entire (large) packet. Hence the currently defined IPv6 UDP checksum is poorly suited to use within a router that is unable to access the entire packet and does not provide checksum-offloading. Thus enabling checksum calculation over the complete packet can impact router design, performance improvement, energy consumption and/or cost.

#### 1.3.4. Interactions with middleboxes

In IPv4, UDP-encapsulation may be desirable for NAT traversal, since UDP support is commonly provided. It is also necessary due to the almost ubiquitous deployment of IPv4 NATs. There has also been discussion of NAT for IPv6, although not for the same reason as in IPv4. If IPv6 NAT becomes a reality they hopefully do not present the same protocol issues as for IPv4. If NAT is defined for IPv6, it should take UDP zero checksum into consideration.

The requirements for IPv6 firewall traversal are likely to be similar to those for IPv4. In addition, it can be reasonably expected that a firewall conforming to RFC 2460 will not regard UDP datagrams with a zero checksum as valid packets. If a zero-checksum for UDP were to be allowed for IPv6, this would need firewalls to be updated before full utility of the change is available.

It can be expected that UDP with zero-checksum will initially not have the same middlebox traversal characteristics as regular UDP. However, if standardized we can expect an improvement over time of the traversal capabilities. We also note that deployment of IPv6-capable middleboxes is still in its initial phases. Thus, it might be that the number of non-updated boxes quickly become a very small percentage of the deployed middleboxes.

#### 1.3.5. Support for load balancing

The UDP port number fields have been used as a basis to design load-balancing solutions for IPv4. This approach has also been leveraged for IPv6. An alternate method would be to utilise the IPv6 Flow Label as basis for entropy for the load balancing. This would have the desirable effect of releasing IPv6 load-balancing devices from the need to assume semantics for the use of the transport port field and also works for all type of transport protocols. This use of the flow-label is consistent with the intended use, although further clarity may be needed to ensure the field can be consistently used for this purpose, (e.g. Equal-Cost Multi-Path routing, ECMP [ECMP]).

Router vendors could be encouraged to start using the IPv6 Flow Label as a part of the flow hash, providing support for ECMP without requiring use of UDP. However, the method for populating the outer IPv6 header with a value for the flow label is not trivial: If the inner packet uses IPv6, then the flow label value could be copied to the outer packet header. However, many current end-points set the flow label to a zero value (thus no entropy). The ingress of a tunnel seeking to provide good entropy in the flow label field would therefore need to create a random flow label value and keep corresponding state, so that all packets that were associated with a flow would be consistently given the same flow label. Although possible, this complexity may not be desirable in a tunnel ingress.

The end-to-end use of flow labels for load balancing is a long-term solution. Even if the usage of the flow label is clarified, there would be a transition time before a significant proportion of end-points start to assign a good quality flow label to the flows that they originate, with continued use of load balancing using the transport header fields until any widespread deployment is finally achieved.

## 2. Standards-Track Transports

The IETF has defined a set of transport protocols that may be applicable for tunnels with IPv6. There are also a set of network layer encapsulation tunnels such as IP-in-IP and GRE. These already standardized solutions are discussed here prior to the issues, as background for the issue description and some comparison of where the issue may already occur.

### 2.1. UDP with Standard Checksum

UDP [RFC0768] with standard checksum behaviour, as defined in RFC 2460, has already been discussed. UDP usage guidelines are provided in [RFC5405].

### 2.2. UDP-Lite

UDP-Lite [RFC3828] offers an alternate transport to UDP, specified as a proposed standard, RFC 3828. A MIB is defined in RFC 5097 and unicast usage guidelines in [RFC5405]. There is at least one open source implementation as a part of the Linux kernel since version 2.6.20.

UDP-Lite provides a checksum with optional partial coverage. When using this option, a datagram is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). When the checksum covers the entire packet, UDP-Lite is fully equivalent with UDP. Errors/corruption in the insensitive part will not cause the datagram to be discarded by the transport layer at the receiving endpoint. A minor side-effect of using UDP-Lite is that this was specified for damage-tolerant payloads, and some link-layers may employ different link encapsulations when forwarding UDP-Lite segments (e.g. radio access bearers). Most link-layers will cover the insensitive part with the same strong layer 2 frame CRC that covers the sensitive part.

#### 2.2.1. Using UDP-Lite as a Tunnel Encapsulation

Tunnel encapsulations can use UDP-Lite (e.g. Control And Provisioning of Wireless Access Points, CAPWAP [RFC5415]), since UDP-Lite provides a transport-layer checksum, including an IP pseudo header checksum, in IPv6, without the need for a router/middlebox to traverse the entire packet payload. This provides most of the verification required for delivery and still keeps the complexity of the checksumming operation low. UDP-Lite may set the length of checksum coverage on a per packet basis. This feature could be used if a tunnel protocol is designed to only verify delivery of the tunneled payload and uses full checksumming for control information.

There is currently poor support for middlebox traversal using UDP-Lite, because UDP-Lite uses a different IPv6 network-layer Next Header value to that of UDP, and few middleboxes are able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited to protocols needing general Internet support, until such time that UDP-Lite has achieved better support in middleboxes and end-points.

#### 2.3. General Tunnel Encapsulations

The IETF has defined a set of tunneling protocols or network layer encapsulations, e.g., IP-in-IP and GRE. These either do not include a checksum or use a checksum that is optional, since tunnel encapsulations are typically layered directly over the Internet layer (identified by the upper layer type in the IPv6 Next Header field) and are also not used as endpoint transport protocols. There is little chance of confusing a tunnel-encapsulated packet with other application data that could result in corruption of application state or data.

From the end-to-end perspective, the principal difference is that the network-layer Next Header field identifies a separate transport, which reduces the probability that corruption could result in the packet being delivered to the wrong endpoint or application. Specifically, packets are only delivered to protocol modules that process a specific next header value. The next header field therefore provides a first-level check of correct demultiplexing. In contrast, the UDP port space is shared by many diverse applications and therefore UDP demultiplexing relies solely on the port numbers.

### 3. Issues Requiring Consideration

This section evaluates issues around the proposal to update IPv6 [RFC2460], to provide the option of using a UDP transport checksum set to zero. Some of the identified issues are shared with other protocols already in use.

The decision by IPv6 to omit an integrity check at the network level has meant that the transport check was overloaded with many functions, including validating:

- o the endpoint address was not corrupted within a router, i.e., a packet was intended to be received by this destination and validate that the packet does not consist of a wrong header spliced to a different payload;
- o that extension header processing is correctly delimited - i.e., the start of data has not been corrupted. In this case, reception of a valid next header value provides some protection;
- o reassembly processing, when used;
- o the length of the payload;
- o the port values - i.e., the correct application receives the payload (applications should also check the expected use of source ports/addresses);
- o the payload integrity.

In IPv4, the first four checks are performed using the IPv4 header checksum.

In IPv6, these checks occur within the endpoint stack using the UDP checksum information. An IPv6 node also relies on the header information to determine whether to send an ICMPv6 error message [RFC4443] and to determine the node to which this is sent. Corrupted information may lead to misdelivery to an unintended application socket on an unexpected host.

#### 3.1. Effect of packet modification in the network



IP packets may be corrupted as they traverse an Internet path. Evidence has been presented [Sigcomm2000] to show that this was once an issue with IPv4 routers, and occasional corruption could result from bad internal router processing in routers or hosts. These errors are not detected by the strong frame checksums employed at the link-layer [RFC3819]. There is no current evidence that such cases are rare in the modern Internet, nor that they may not be applicable to IPv6. It therefore seems prudent not to relax this constraint. The emergence of low-end IPv6 routers and the proposed use of NAT with IPv6 further motivate the need to protect from this type of error.

Corruption in the network may result in:

- o A datagram being mis-delivered to the wrong host/router or the wrong transport entity within an endpoint. Such a datagram needs to be discarded;
- o A datagram payload being corrupted, but still delivered to the intended host/router transport entity. Such a datagram needs to be either discarded or correctly processed by an application that provides its own integrity checks;
- o A datagram payload being truncated by corruption of the length field. Such a datagram needs to be discarded.

When a checksum is used, this significantly reduces the impact of errors, reducing the probability of undetected corruption of state (and data) on both the host stack and the applications using the transport service.

The following sections examine the impact of modifying each of these header fields.

#### 3.1.1. Corruption of the destination IP address

An IP endpoint destination address could be modified in the network (e.g. corrupted by an error). This is not a concern for IPv4, because the IP header checksum will result in this packet being discarded by the receiving IP stack. Such modification in the network can not be detected at the network layer when using IPv6.

There are two possible outcomes:

- o Delivery to a destination address that is not in use (the packet will not be delivered, but could result in an error report);

- o Delivery to a different destination address. This modification will normally be detected by the transport checksum, resulting in silent discard. Without this checksum, the packet would be passed to the endpoint port demultiplexing function. If an application is bound to the associated ports, the packet payload will be passed to the application (see the subsequent section on port processing).

### 3.1.2. Corruption of the source IP address

This section examines what happens when the source address is corrupted in transit. This is not a concern in IPv4, because the IP header checksum will normally result in this packet being discarded by the receiving IP stack.

Corruption of an IPv6 source address does not result in the IP packet being delivered to a different endpoint protocol or destination address. If only the source address is corrupted, the datagram will likely be processed in the intended context, although with erroneous origin information. The result will depend on the application or protocol that processes the packet. Some examples are:

- o An application that requires a per-established context may disregard the datagram as invalid, or could map this to another context (if a context for the modified source address was already activated).
- o A stateless application will process the datagram outside of any context, a simple example is the ECHO server, which will respond with a datagram directed to the modified source address. This would create unwanted additional processing load, and generate traffic to the modified endpoint address.
- o Some datagram applications build state using the information from packet headers. A previously unused source address would result in receiver processing and the creation of unnecessary transport-layer state at the receiver. For example, Real Time Protocol (RTP) [RFC3550] sessions commonly employ a source independent receiver port. State is created for each received flow.

Reception of a datagram with a corrupted source address will therefore result in accumulation of unnecessary state in the RTP state machine, including collision detection and response (since the same synchronization source, SSRC, value will appear to arrive from multiple source IP addresses).

In general, the effect of corrupting the source address will depend upon the protocol that processes the packet and its robustness to this error. For the case where the packet is received by a tunnel endpoint, the tunnel application is expected to correctly handle a corrupted source address.

The impact of source address modification is more difficult to quantify when the receiving application is not that originally intended and several fields have been modified in transit.

### 3.1.3. Corruption of Port Information

This section describes what happens if one or both of the UDP port values are corrupted in transit. This can also happen with IPv4 in the zero checksum case, but not when UDP checksums are enabled or with UDP-Lite. If the ports carried in the transport header of an IPv6 packet were corrupted in transit, packets may be delivered to the wrong process (on the intended machine) and/or responses or errors sent to the wrong application process (on the intended machine).

### 3.1.4. Delivery to an unexpected port

If one combines the corruption effects, such as destination address and ports, there is a number of potential outcomes when traffic arrives at an unexpected port. This section discusses these possibilities and their outcomes for a packet that does not use the UDP checksum validation:

- o Delivery to a port that is not in use. The packet is discarded, but could generate an ICMPv6 message (e.g. port unreachable).
- o It could be delivered to a different node that implements the same application, where the packet may be accepted, generating side-effects or accumulated state.
- o It could be delivered to an application that does not implement the tunnel protocol, where the packet may be incorrectly parsed, and may be misinterpreted, generating side-effects or accumulated state.

The probability of each outcome depends on the statistical probability that the address or the port information for the source or destination becomes corrupt in the datagram such that they match those of an existing flow or server port. Unfortunately, such a match may be more likely for UDP than for connection-oriented transports, because:

1. There is no handshake prior to communication and no sequence numbers (as in TCP, DCCP, or SCTP). Together, this makes it hard to verify that an application is given only the data associated with a transport session.
2. Applications writers often bind to wild-card values in endpoint identifiers and do not always validate correctness of datagrams they receive (guidance on this topic is provided in [RFC5405]).

While these rules could, in principle, be revised to declare naive applications as "Historic". This remedy is not realistic: the transport owes it to the stack to do its best to reject bogus datagrams.

If checksum coverage is suppressed, the application therefore needs to provide a method to detect and discard the unwanted data. A tunnel protocol would need to perform its own integrity checks on any control information if transported in UDP with zero-checksum. If the tunnel payload is another IP packet, the packets requiring checksums can be assumed to have their own checksums provided that the rate of corrupted packets is not significantly larger due to the tunnel encapsulation. If a tunnel transports other inner payloads that do not use IP, the assumptions of corruption detection for that particular protocol must be fulfilled, this may require an additional checksum/CRC and/or integrity protection of the payload and tunnel headers.

A protocol using UDP zero-checksum can never assume that it is the only protocol using a zero checksum. Therefore, it needs to gracefully handle misdelivery. It must be robust to reception of malformed packets received on a listening port and expect that these packets may contain corrupted data or data associated with a completely different protocol.

#### 3.1.5. Corruption of Fragmentation Information

The fragmentation information in IPv6 employs a 32-bit identity field, compared to only a 16-bit field in IPv4, a 13-bit fragment offset and a 1-bit flag, indicating if there are more fragments. Corruption of any of these field may result in one of two outcomes:

Reassembly failure: An error in the "More Fragments" field for the last fragment will for example result in the packet never being considered complete and will eventually be timed out and discarded. A corruption in the ID field will result in the fragment not being delivered to the intended context thus leaving the rest incomplete, unless that packet has been duplicated prior

to corruption. The incomplete packet will eventually be timed out and discarded.

Erroneous reassembly: The re-assembled packet did not match the original packet. This can occur when the ID field of a fragment is corrupted, resulting in a fragment becoming associated with another packet and taking the place of another fragment. Corruption in the offset information can cause the fragment to be misaligned in the reassembly buffer, resulting in incorrect reassembly. Corruption can cause the packet to become shorter or longer, however completion of reassembly is much less probable, since this would require consistent corruption of the IPv6 headers payload length field and the offset field. The possibility of mis-assembly requires the reassembling stack to provide strong checks that detect overlap or missing data, note however that this is not guaranteed and has recently been clarified in "Handling of Overlapping IPv6 Fragments" [RFC5722].

The erroneous reassembly of packets is a general concern and such packets should be discarded instead of being passed to higher layer processes. The primary detector of packet length changes is the IP payload length field, with a secondary check by the transport checksum. The Upper-Layer Packet length field included in the pseudo header assists in verifying correct reassembly, since the Internet checksum has a low probability of detecting insertion of data or overlap errors (due to misplacement of data). The checksum is also incapable of detecting insertion or removal of all zero-data that occurs in a multiple of a 16-bit chunk.

The most significant risk of corruption results following mis-association of a fragment with a different packet. This risk can be significant, since the size of fragments is often the same (e.g. fragments resulting when the path MTU results in fragmentation of a larger packet, common when addition of a tunnel encapsulation header expands the size of a packet). Detection of this type of error requires a checksum or other integrity check of the headers and the payload. Such protection is anyway desirable for tunnel encapsulations using IPv4, since the small fragmentation ID can easily result in wrap-around [RFC4963], this is especially the case for tunnels that perform flow aggregation [I-D.ietf-intarea-tunnels].

Tunnel fragmentation behavior matters. There can be outer or inner fragmentation "Tunnels in the Internet Architecture" [I-D.ietf-intarea-tunnels]. If there is inner fragmentation by the tunnel, the outer headers will never be fragmented and thus a zero-checksum in the outer header will not affect the reassembly process. When a tunnel performs outer header fragmentation, the tunnel egress needs to perform reassembly of the outer fragments into an inner packet. The inner packet is either a complete packet or a fragment. If it is a fragment, the destination endpoint of the fragment will perform reassembly of the received fragments. The complete packet or the reassembled fragments will then be processed according to the packet next header field. The receiver may only detect reassembly anomalies when it uses a protocol with a checksum. The larger the number of reassembly processes to which a packet has been subjected, the greater the probability of an error.

- o An IP-in-IP tunnel that performs inner fragmentation has similar properties to a UDP tunnel with a zero-checksum that also performs inner fragmentation.
- o An IP-in-IP tunnel that performs outer fragmentation has similar properties to a UDP tunnel with a zero checksum that performs outer fragmentation.
- o A tunnel that performs outer fragmentation can result in a higher level of corruption due to both inner and outer fragmentation, enabling more chances for reassembly errors to occur.
- o Recursive tunneling can result in fragmentation at more than one header level, even for inner fragmentation unless it goes to the inner most IP header.
- o Unless there is verification at each reassembly, the probability for undetected error will increase with the number of times fragmentation is recursively applied, making IP-in-IP and UDP with zero checksum both vulnerable to undetected errors.

In conclusion fragmentation of packets with a zero-checksum does not worsen the situation compared to some other commonly used tunnel encapsulations. However, caution is needed for recursive tunneling without any additional verification at the different tunnel layers.

### 3.2. Validating the network path

IP transports designed for use in the general Internet should not assume specific path characteristics. Network protocols may reroute packets that change the set of routers and middleboxes along a path. Therefore transports such as TCP, SCTP and DCCP have been designed to negotiate protocol parameters, adapt to different network path characteristics, and receive feedback to verify that the current path is suited to the intended application. Applications using UDP and

UDP-Lite need to provide their own mechanisms to confirm the validity of the current network path.

The zero-checksum in UDP is explicitly disallowed in RFC2460. Thus it may be expected that any device on the path that has a reason to look beyond the IP header will consider such a packet as erroneous or illegal and may likely discard it, unless the device is updated to support a new behavior. A pair of end-points intending to use a new behavior will therefore not only need to ensure support at each end-point, but also that the path between them will deliver packets with the new behavior. This may require negotiation or an explicit mandate to use the new behavior by all nodes intended to use a new protocol.

Support along the path between end points may be guaranteed in limited deployments by appropriate configuration. In general, it can be expected to take time for deployment of any updated behaviour to become ubiquitous. A sender will need to probe the path to verify the expected behavior. Path characteristics may change, and usage therefore should be robust and able to detect a failure of the path under normal usage and re-negotiate. This will require periodic validation of the path, adding complexity to any solution using the new behavior.

### 3.3. Applicability of method

The expectation of the present proposal defined in [I-D.ietf-6man-udpchecksums] is that this change would only apply to IPv6 router nodes that implement specific protocols that permit omission of UDP checksums. However, the distinction between a router and a host is not always clear, especially at the transport level. Systems (such as unix-based operating systems) routinely provide both functions. There is also no way to identify the role of a receiver from a received packet.

Any new method would therefore need a specific applicability statement indicating when the mechanism can (and can not) be used. Enabling this, and ensuring correct interactions with the stack, implies much more than simply disabling the checksum algorithm for specific packets at the transport interface.

The IETF should carefully consider constraints on sanctioning the use of any new transport mode. If this is specified and widely available, it may be expected to be used by applications that are perceived to gain benefit. Any solution that uses an end-to-end transport protocol, rather than an IP-in-IP encapsulation, needs to

minimise the possibility that end-hosts could confuse a corrupted or wrongly delivered packet with that of data addressed to an application running on their endpoint unless they accept that behavior.

### 3.4. Impact on non-supporting devices or applications

It is important to consider what potential impact the zero-checksum behavior may have on end-points, devices or applications that are not modified to support the new behavior or by default or preference, use the regular behavior. These applications must not be significantly impacted by the changes.

To illustrate a potential issue, consider the implications of a node that were to enable use of a zero-checksum at the interface level: This would result in all applications that listen to a UDP socket receiving datagram where the checksum was not verified. This could have a significant impact on an application that was not designed with the additional robustness needed to handle received packets with corruption, creating state or destroying existing state in the application.

In contrast, the use of a zero-checksum could be enabled only for individual ports using an explicit request by the application. In this case, applications using other ports would maintain the current IPv6 behavior, discarding incoming UDP datagrams with a zero-checksum. These other applications would not be effected by this changed behavior. An application that allows the changed behavior should be aware of the risk for corruption and the increased level of misdirected traffic, and can be designed robustly to handle this risk.

## 4. Evaluation of proposal to update RFC 2460 to support zero checksum

This section evaluates the proposal to update IPv6 [RFC2460], to provide the option that some nodes may suppress generation and checking of the UDP transport checksum. It also compares the proposal with other alternatives.

### 4.1. Alternatives to the Standard Checksum

There are several alternatives to the normal method for calculating the UDP Checksum that do not require a tunnel endpoint to inspect the



entire packet when computing a checksum. These include (in decreasing order of complexity):

- o Delta computation of the checksum from an encapsulated checksum field. Since the checksum is a cumulative sum [RFC1624], an encapsulating header checksum can be derived from the new pseudo header, the inner checksum and the sum of the other network-layer fields not included in the pseudo header of the encapsulated packet, in a manner resembling incremental checksum update [RFC1141]. This would not require access to the whole packet, but does require fields to be collected across the header, and arithmetic operations on each packet. The method would only work for packets that contain a 2's complement transport checksum (i.e. it would not be appropriate for SCTP or when IP fragmentation is used).
- o UDP-Lite with the checksum coverage set to only the header portion of a packet. This requires a pseudo header checksum calculation only on the encapsulating packet header. The computed checksum value may be cached (before adding the Length field) for each flow/destination and subsequently combined with the Length of each packet to minimise per-packet processing. This value is combined with the UDP payload length for the pseudo header, however this length is expected to be known when performing packet forwarding.
- o The proposed UDP Tunnel Transport, UDPTT [UDPTT] suggested a method where UDP would be modified to derive the checksum only from the encapsulating packet protocol header. This value does not change between packets in a single flow. The value may be cached per flow/destination to minimise per-packet processing.
- o There has been a proposal to simply ignore the UDP checksum value on reception at the tunnel egress, allowing a tunnel ingress to insert any value correct or false. For tunnel usage, a non standard checksum value may be used, forcing an RFC 2460 receiver to drop the packet. The main downside is that it would be impossible to identify a UDP packet (in the network or an endpoint) that is treated in this way compared to a packet that has actually been corrupted.
- o A method has been proposed that uses a new (to be defined) IPv6 Destination Options Header to provide an end-to-end validation check at the network layer. This would allow an endpoint to verify delivery to an appropriate end point, but would also require IPv6 nodes to correctly handle the additional header, and would require changes to middlebox behavior (e.g. when used with a NAT that always adjusts the checksum value).
- o UDP modified to disable checksum processing [I-D.ietf-6man-udpchecksums]. This requires no checksum calculation, but would require constraints on appropriate usage and updates to end-points and middleboxes.

- o IP-in-IP tunneling. As this method completely dispenses with a transport protocol in the outer-layer it has reduced overhead and complexity, but also reduced functionality. There is no outer checksum over the packet and also no ports to perform demultiplexing between different tunnel types. This reduces the information available upon which a load balancer may act.

These options are compared and discussed further in the following sections.

#### 4.2. Comparison

This section compares the above listed methods to support datagram tunneling. It includes proposals for updating the behaviour of UDP.

##### 4.2.1. Middlebox Traversal

Regular UDP with a standard checksum or the delta encoded optimization for creating correct checksums have the best possibilities for successful traversal of a middlebox. No new support is required.

A method that ignores the UDP checksum on reception is expected to have a good probability of traversal, because most middleboxes perform an incremental checksum update. UDPTT may also traverse a middlebox with this behaviour. However, a middlebox on the path that attempts to verify a standard checksum will not forward packets using either of these methods, preventing traversal. The methods that ignores the checksum has an additional downside in that middlebox traversal can not be improved, because there is no way to identify which packets use the modified checksum behaviour.

IP-in-IP or GRE tunnels offer good traversal of middleboxes that have not been designed for security, e.g. firewalls. However, firewalls may be expected to be configured to block general tunnels as they present a large attack surface.

A new IPv6 Destination Options header will suffer traversal issues with middleboxes, especially Firewalls and NATs, and will likely require them to be updated before the extension header is passed.

Packets using UDP with a zero checksum will not be passed by any middlebox that validates the checksum using RFC 2460 or updates the checksum field, such as NAT or firewalls. This would require an update to correctly handle the zero checksum packets.

UDP-Lite will require an update of almost all type of middleboxes, because it requires support for a separate network-layer protocol number. Once enabled, the method to support incremental checksum update would be identical to that for UDP, but different for checksum

validation.

#### 4.2.2. Load Balancing

The usefulness of solutions for load balancers depends on the difference in entropy in the headers for different flows that can be included in a hash function. All the proposals that use the UDP protocol number have equal behavior. UDP-Lite has the potential for equally good behavior as for UDP. However, UDP-Lite is currently likely to not be supported by deployed hashing mechanisms, which may cause a load balancer to not use the transport header in the computed hash. A load balancer that only uses the IP header will have low entropy, but could be improved by including the IPv6 the flow label, providing that the tunnel ingress ensures that different flow labels are assigned to different flows. However, a transition to the common use of good quality flow labels is likely to take time to deploy.

#### 4.2.3. Ingress and Egress Performance Implications

IP-in-IP tunnels are often considered efficient, because they introduce very little processing and low data overhead. The other proposals introduce a UDP-like header incurring associated data overhead. Processing is minimised for the zero-checksum method, ignoring the checksum on reception, and only slightly higher for UDPTT, the extension header and UDP-Lite. The delta-calculation scheme operates on a few more fields, but also introduces serious failure modes that can result in a need to calculate a checksum over the complete packet. Regular UDP is clearly the most costly to process, always requiring checksum calculation over the entire packet.

It is important to note that the zero-checksum method, ignoring checksum on reception, the Option Header, UDPTT and UDP-Lite will likely incur additional complexities in the application to incorporate a negotiation and validation mechanism.

#### 4.2.4. Deployability

The major factors influencing deployability of these solutions are a need to update both end-points, a need for negotiation and the need to update middleboxes. These are summarised below:

- o The solution with the best deployability is regular UDP. This requires no changes and has good middlebox traversal characteristics.
- o The next easiest to deploy is the delta checksum solution. This does not modify the protocol on the wire and only needs changes in tunnel ingress.
- o IP-in-IP tunnels should not require changes to the end-points, but

raise issues when traversing firewalls and other security-type devices, which are expected to require updates.

- o Ignoring the checksum on reception will require changes at both end-points. The never ceasing risk of path failure requires additional checks to ensure this solution is robust and will require changes or additions to the tunneling control protocol to negotiate support and validate the path.
- o The remaining solutions offer similar deployability. UDP-Lite requires support at both end-points and in middleboxes. UDPTT and Zero-checksum with or without an Extension header require support at both end-points and in middleboxes. UDP-Lite, UDPTT, and Zero-checksum and Extension header may additionally require changes or additions to the tunneling control protocol to negotiate support and path validation.

#### 4.2.5. Corruption Detection Strength

The standard UDP checksum and the delta checksum can both provide some verification at the tunnel egress. This can significantly reduce the probability that a corrupted inner packet is forwarded. UDP-Lite, UDPTT and the extension header all provide some verification against corruption, but do not verify the inner packet. They only provide a strong indication that the delivered packet was intended for the tunnel egress and was correctly delimited. The Zero-checksum, ignoring the checksum on reception and IP-and-IP encapsulation provide no verification that a received packet was intended to be processed by a specific tunnel egress or that the inner packet was correct.

#### 4.2.6. Comparison Summary

The comparisons above may be summarised as "there is no silver bullet that will slay all the issues". One has to select which down side(s) can best be lived with. Focusing on the existing solutions, this can be summarized as:

Regular UDP: Good middlebox traversal and load balancing and multiplexing, requiring a checksum in the outer headers covering the whole packet.

IP in IP: A low complexity encapsulation, with limited middlebox traversal, no multiplexing support, and currently poor load balancing support that could improve over time.

UDP-Lite: A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, currently poor load balancing support that could improve over time, in most cases requiring application level negotiation

and validation.

The delta-checksum is an optimization in the processing of UDP, as such it exhibits some of the drawbacks of using regular UDP.

The remaining proposals may be described in similar terms:

Zero-Checksum: A low complexity encapsulation, with good multiplexing support, limited middlebox traversal that could improve over time, good load balancing support, in most cases requiring application level negotiation and validation.

UDPTT: A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, good load balancing support, in most cases requiring application level negotiation and validation.

IPv6 Destination Option IP in IP tunneling: A medium complexity, with no multiplexing support, limited middlebox traversal, currently poor load balancing support that could improve over time, in most cases requiring application level negotiation and validation.

IPv6 Destination Option combined with UDP Zero-checksumming: A medium complexity encapsulation, with good multiplexing support, limited load balancing support that could improve over time, in most cases requiring application level negotiation and validation.

Ignore the checksum on reception: A low complexity encapsulation, with good multiplexing support, medium middlebox traversal that never can improve, good load balancing support, in most cases requiring application level negotiation and validation.

There is no clear single optimum solution. If the most important need is to traverse middleboxes, then the best choice is to stay with regular UDP and consider the optimizations that may be required to perform the checksumming. If one can live with limited middlebox traversal, low complexity is necessary and one does not require load balancing, then IP-in-IP tunneling is the simplest. If one wants strengthened error detection, but with currently limited middlebox traversal and load-balancing. UDP-Lite is appropriate. UDP Zero-checksum addresses another set of constraints, low complexity and a need for load balancing from the current Internet, providing it can live with currently limited middlebox traversal.

Techniques for load balancing and middlebox traversal do continue to evolve. Over a long time, developments in load balancing have good potential to improve. This time horizon is long since it requires both load balancer and end-point updates to get full benefit. The challenges of middlebox traversal are also expected to change with time, as device capabilities evolve. Middleboxes are very prolific with a larger proportion of end-user ownership, and therefore may be expected to take long time cycles to evolve. One potential advantage is that the deployment of IPv6 capable middleboxes are still in its initial phase and the quicker zero-checksum becomes standardized the fewer boxes will be non-compliant.

Thus, the question of whether to allow UDP with a zero-checksum for IPv6 under reasonable constraints, is therefore best viewed as a trade-off between a number of more subjective questions:

- o Is there sufficient interest in zero-checksum with the given constraints (summarised below)?
- o Are there other avenues of change that will resolve the issue in a better way and sufficiently quickly ?
- o Do we accept the complexity cost of having one more solution in the future?

The authors do think the answer to the above questions are such that zero-checksum should be standardized for use by tunnel encapsulations.

## 5. Requirements on the specification of transported protocols

This section identifies requirements for the protocols that are transported over a transport connection that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints.

### 5.1. Constraints required on usage of a zero checksum

If a zero checksum approach were to be adopted by the IETF, the specification should consider adding the following constraints on usage:

1. IPv6 protocol stack implementations should not by default allow the new method. The default node receiver behaviour must discard all IPv6 packets carrying UDP packets with a zero checksum.
2. Implementations must provide a way to signal the set of ports that will be enabled to receive UDP datagrams with a zero checksum. An IPv6 node that enables reception of UDP packets with a zero-checksum, must enable this only for a specific port or port-range. This may be implemented via a socket API call, or

similar mechanism.

3. RFC 2460 specifies that IPv6 nodes should log UDP datagrams with a zero-checksum. This should remain the case for any datagram received on a port that does not explicitly enable zero-checksum processing. A port for which zero-checksum has been enabled must not log the datagram.
4. A stack may separately identify UDP datagrams that are discarded with a zero checksum. It should not add these to the standard log, since the endpoint has not been verified.
5. Tunnels that encapsulate IP may rely on the inner packet integrity checks provided that the tunnel will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel must provide an additional integrity verification mechanism. An integrity mechanisms is always recommended at the tunnel layer to ensure that corruption rates of the inner most packet are not increased.
6. Tunnels that encapsulate Non-IP packets must have a CRC or other mechanism for checking packet integrity, unless the Non-IP packet specifically is designed for transmission over lower layers that do not provide any packet integrity guarantee. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.
7. UDP applications that support use of a zero-checksum, should not rely upon correct reception of the IP and UDP protocol information (including the length of the packet) when decoding and processing the packet payload. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.
8. If a method proposes recursive tunnels, it needs to provide guidance that is appropriate for all use-cases. Restrictions may be needed to the use of a tunnel encapsulations and the use of recursive tunnels (e.g. Necessary when the endpoint is not verified).
9. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum must provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and port numbers in the ICMPv6 message body).

Deployment of the new method needs to remain restricted to endpoints that explicitly enable this mode and adopt the above procedures. Any

middlebox that examines or interacts with the UDP header over IPv6 should support the new method.

## 6. Summary

This document examines the role of the transport checksum when used with IPv6, as defined in RFC2460.

It presents a summary of the trade-offs for evaluating the safety of updating RFC 2460 to permit an IPv6 UDP endpoint to use a zero value in the checksum field to indicate that no checksum is present. A decision not to include a UDP checksum in received IPv6 datagrams could impact a tunnel application that receives these packets. However, a well-designed tunnel application should include consistency checks to validate any header information encapsulated with a packet. In most cases tunnels encapsulating IP packets can rely on the inner packets own integrity protection. When correctly implemented, such a tunnel endpoint will not be negatively impacted by omission of the transport-layer checksum. Recursive tunneling and fragmentation is a potential issue that can raise corruption rates significantly, and requires careful consideration.

Other applications at the intended destination node or another IPv6 node can be impacted if they are allowed to receive datagrams without a transport-layer checksum. It is particularly important that already deployed applications are not impacted by any change at the transport layer. If these applications execute on nodes that implement RFC 2460, they will reject all datagrams with a zero UDP checksum, thus this is not an issue. For nodes that implement support for zero-checksum it is important to ensure that only UDP applications that desire zero-checksum can receive and originate zero-checksum packets. Thus, the enabling of zero-checksum needs to be at a port level, not for the entire host or for all use of an interface.

The implications on firewalls, NATs and other middleboxes need to be considered. It is not expected that IPv6 NATs handle IPv6 UDP datagrams in the same way that they handle IPv4 UDP datagrams. This possibly reduces the need to update the checksum. Firewalls are intended to be configured, and therefore may need to be explicitly updated to allow new services or protocols. IPv6 middlebox deployment is not yet as prolific as it is in IPv4. Thus, relatively few current middleboxes may actually block IPv6 UDP with a zero checksum.

In general, UDP-based applications need to employ a mechanism that allows a large percentage of the corrupted packets to be removed before they reach an application, both to protect the data stream of the application and the control plane of higher layer protocols. These checks are currently performed by the UDP checksum for IPv6, or the reduced checksum for UDP-Lite when used with IPv6.



The use of UDP with no checksum has merits for some applications, such as tunnel encapsulation, and is widely used in IPv4. However, there are dangers for IPv6: There is a bigger risk of corruption and miss-delivery when using zero-checksum in IPv6 compared to IPv4 due to the removed IP header checksum. Thus, applications need to make a new evaluation of the risks of enabling a zero-checksum. Some applications will need to re-consider their usage of zero-checksum, and possibly consider a solution that at least provides the same delivery protection as for IPv4, for example by utilizing UDP-Lite, or by enabling the UDP checksum. Tunnel applications using UDP for encapsulation can in many case use zero-checksum without significant impact on the corruption rate. In some cases, the use of checksum off-loading may help alleviate the checksum processing cost.

Recursive tunneling and fragmentation is a difficult issue relating to tunnels in general. There is an increased risk of an error in the inner-most packet when fragmentation when several layers of tunneling and several different reassembly processes are run without any verification of correctness. This issue requires future thought and consideration.

The conclusion is that UDP zero checksum in IPv6 should be standardized, as it satisfies usage requirements that are currently difficult to address. We do note that a safe deployment of zero-checksum will need to follow a set of constraints listed in Section 5.1.

## 7. Acknowledgements

Brian Haberman, Brian Carpenter, Magaret Wasserman, Lars Eggert, others in the TSV directorate.

Thanks also to: Remi Denis-Courmont, Pekka Savola and many others who contributed comments and ideas via the 6man, behave, lisp and mboned lists.

## 8. IANA Considerations

This document does not require any actions by IANA.

## 9. Security Considerations

Transport checksums provide the first stage of protection for the stack, although they can not be considered authentication mechanisms. These checks are also desirable to ensure packet counters correctly log actual activity, and can be used to detect unusual behaviours.

## 10. References

### 10.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC1071] Braden, R., Borman, D., Partridge, C. and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.
- [RFC2460] Deering, S.E. and R.M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

## 10.2. Informative References

- [ECMP] "Using the IPv6 flow label for equal cost multipath routing in tunnels (draft-carpenter-flow-ecmp)", .
- [I-D.ietf-6man-udpchecksums] Eubanks, M. and P. Chimento, "UDP Checksums for Tunneled Packets", Internet-Draft draft-ietf-6man-udpchecksums-02, March 2012.
- [I-D.ietf-intarea-tunnels] Touch, J. and M. Townsley, "Tunnels in the Internet Architecture", Internet-Draft draft-ietf-intarea-tunnels-00, March 2010.
- [I-D.ietf-mboned-auto-multicast] Bumgardner, G., "Automatic Multicast Tunneling", Internet-Draft draft-ietf-mboned-auto-multicast-14, June 2012.
- [LISP] D. Farinacci et al, , "Locator/ID Separation Protocol (LISP)", March 2009.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC1141] Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, January 1990.
- [RFC1624] Rijssinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J. and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E. and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

- [RFC4443] Conta, A., Deering, S. and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4963] Heffner, J., Mathis, M. and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5415] Calhoun, P., Montemurro, M. and D. Stanley, "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, March 2009.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, December 2009.
- [RFC6145] Li, X., Bao, C. and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.
- [Sigcomm2000] Jonathan Stone and Craig Partridge , , "When the CRC and TCP Checksum Disagree", 2000.
- [UDPTT] G Fairhurst, , "The UDP Tunnel Transport mode", Feb 2010.

#### Appendix A. Document Change History

{RFC EDITOR NOTE: This section must be deleted prior to publication}

Individual Draft 00 This is the first DRAFT of this document - It contains a compilation of various discussions and contributions from a variety of IETF WGs, including: mboned, tsv, 6man, lisp, and behave. This includes contributions from Magnus with text on RTP, and various updates.

Individual Draft 01

- \* This version corrects some typos and editorial NiTs and adds discussion of the need to negotiate and verify operation of a new mechanism (3.3.4).

Individual Draft 02

- \* Version -02 corrects some typos and editorial NiTs.
- \* Added reference to ECMP for tunnels.
- \* Clarifies the recommendations at the end of the document.

Working Group Draft 00

- \* Working Group Version -00 corrects some typos and removes much of rationale for UDPTT. It also adds some discussion of IPv6 extension header.

Working Group Draft 01

- \* Working Group Version -01 updates the rules and incorporates off-list feedback. This version is intended for wider review within the 6man working group.

Working Group Draft 02

- \* This version is the result of a major rewrite and re-ordering of the document.
- \* A new section comparing the results have been added.
- \* The constraints list has been significantly altered by removing some and rewording other constraints.
- \* This contains other significant language updates to clarify the intent of this draft.

Working Group Draft 03

- \* Editorial updates

Working Group Draft 04

- \* Resubmission only updating the AMT and RFC2765 references.

Working Group Draft 05

- \* Resubmission to correct editorial NiTs - thanks to Bill Atwood for noting these. Group Draft 05.

Working Group Draft 06

- \* Resubmission to keep draft alive (spelling updated from 05).

Authors' Addresses

Godred Fairhurst  
University of Aberdeen  
School of Engineering  
Aberdeen, AB24 3UE,  
Scotland, UK

Email: [gorry@erg.abdn.ac.uk](mailto:gorry@erg.abdn.ac.uk)  
URI: <http://www.erg.abdn.ac.uk/users/gorry>

Magnus Westerlund  
Ericsson  
Farogatan 6  
Stockholm, SE-164 80  
Sweden

Phone: +46 8 719 0000  
Email: magnus.westerlund@ericsson.com

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: August 29, 2013

G. Fairhurst  
University of Aberdeen  
M. Westerlund  
Ericsson  
February 25, 2013

Applicability Statement for the use of IPv6 UDP Datagrams with Zero  
Checksums  
draft-ietf-6man-udpzero-12

Abstract

This document provides an applicability statement for the use of UDP transport checksums with IPv6. It defines recommendations and requirements for the use of IPv6 UDP datagrams with a zero UDP checksum. It describes the issues and design principles that need to be considered when UDP is used with IPv6 to support tunnel encapsulations and examines the role of the IPv6 UDP transport checksum. The document also identifies issues and constraints for deployment on network paths that include middleboxes. An appendix presents a summary of the trade-offs that were considered in evaluating the safety of the update to RFC 2460 that updates use of the UDP checksum with IPv6.

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 August 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

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

## Table of Contents

1. Introduction . . . . .	4
1.1. Document Structure . . . . .	5
1.2. Terminology . . . . .	5
1.3. Use of UDP Tunnels . . . . .	5
1.3.1. Motivation for new approaches . . . . .	6
1.3.2. Reducing forwarding cost . . . . .	6
1.3.3. Need to inspect the entire packet . . . . .	7
1.3.4. Interactions with middleboxes . . . . .	7
1.3.5. Support for load balancing . . . . .	8
2. Standards-Track Transports . . . . .	9
2.1. UDP with Standard Checksum . . . . .	9
2.2. UDP-Lite . . . . .	9
2.2.1. Using UDP-Lite as a Tunnel Encapsulation . . . . .	10
2.3. General Tunnel Encapsulations . . . . .	10
2.4. Relation to UDP-Lite and UDP with checksum . . . . .	10
3. Issues Requiring Consideration . . . . .	12
3.1. Effect of packet modification in the network . . . . .	13
3.1.1. Corruption of the destination IP address . . . . .	14
3.1.2. Corruption of the source IP address . . . . .	15
3.1.3. Corruption of Port Information . . . . .	16
3.1.4. Delivery to an unexpected port . . . . .	16
3.1.5. Corruption of Fragmentation Information . . . . .	17
3.2. Where Packet Corruption Occurs . . . . .	19
3.3. Validating the network path . . . . .	20
3.4. Applicability of method . . . . .	21
3.5. Impact on non-supporting devices or applications . . . . .	21
4. Constraints on implementation of IPv6 nodes supporting zero checksum . . . . .	22
5. Requirements on usage of the zero UDP checksum . . . . .	24
6. Summary . . . . .	26
7. Acknowledgements . . . . .	28
8. IANA Considerations . . . . .	28
9. Security Considerations . . . . .	28
10. References . . . . .	29
10.1. Normative References . . . . .	29
10.2. Informative References . . . . .	29

Appendix A. Evaluation of proposal to update RFC 2460 to	
support zero checksum . . . . .	31
A.1. Alternatives to the Standard Checksum . . . . .	31
A.2. Comparison . . . . .	33
A.2.1. Middlebox Traversal . . . . .	33
A.2.2. Load Balancing . . . . .	34
A.2.3. Ingress and Egress Performance Implications . . . . .	34
A.2.4. Deployability . . . . .	34
A.2.5. Corruption Detection Strength . . . . .	35
A.2.6. Comparison Summary . . . . .	35
Appendix B. Document Change History . . . . .	38
Authors' Addresses . . . . .	41



## 1. Introduction

The User Datagram Protocol (UDP) [RFC0768] transport is defined for the Internet Protocol (IPv4) [RFC0791] and is defined in "Internet Protocol, Version 6 (IPv6) [RFC2460] for IPv6 hosts and routers. The UDP transport protocol has a minimal set of features. This limited set has enabled a wide range of applications to use UDP, but these application do need to provide many important transport functions on top of UDP. The UDP Usage Guidelines [RFC5405] provides overall guidance for application designers, including the use of UDP to support tunneling. The key difference between UDP usage with IPv4 and IPv6 is that RFC 2460 mandates use of a calculated UDP checksum, i.e. a non-zero value, due to the lack of an IPv6 header checksum. The inclusion of the pseudo header in the checksum computation provides a statistical check that datagrams have been delivered to the intended IPv6 destination node. Algorithms for checksum computation are described in [RFC1071].

The lack of a possibility to use an IPv6 datagram with a zero UDP checksum has been observed as a real problem for certain classes of application, primarily tunnel applications. This class of application has been deployed with a zero UDP checksum using IPv4. The design of IPv6 raises different issues when considering the safety of using a UDP checksum with IPv6. These issues can significantly affect applications, both when an endpoint is the intended user and when an innocent bystander (when a packet is received by a different endpoint to that intended).

This document examines the issues and an appendix compares the strengths and weaknesses of a number of proposed solutions. This identifies a set of issues that must be considered and mitigated to be able to safely deploy IPv6 applications that use a zero UDP checksum. The provided comparison of methods is expected to also be useful when considering applications that have different goals from the ones that initiated the writing of this document, especially the use of already standardized methods. The analysis concludes that using a zero UDP checksum is the best method of the proposed alternatives to meet the goals for certain tunnel applications.

This document defines recommendations and requirements for use of IPv6 datagrams with a zero UDP checksum. This usage is expected to have initial deployment issues related to middleboxes, limiting the usability more than desired in the currently deployed Internet. However, this limitation will be largest initially and will reduce as updates are provided in middleboxes that support the zero UDP checksum for IPv6. The document therefore derives a set of constraints required to ensure safe deployment of a zero UDP checksum.

Finally, the document also identifies some issues that require future consideration and possibly additional research.

#### 1.1. Document Structure

Section 1 provides a background to key issues, and introduces the use of UDP as a tunnel transport protocol.

Section 2 describes a set of standards-track datagram transport protocols that may be used to support tunnels.

Section 3 discusses issues with a zero UDP checksum for IPv6. It considers the impact of corruption, the need for validation of the path and when it is suitable to use a zero UDP checksum.

Section 4 is an applicability statement that defines requirements and recommendations on the implementation of IPv6 nodes that support the use of a zero UDP checksum.

Section 5 provides an applicability statement that defines requirements and recommendations for protocols and tunnel encapsulations that are transported over an IPv6 transport that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints.

Section 6 provides the recommendations for standardization of zero UDP checksum with a summary of the findings and notes remaining issues needing future work.

Appendix A evaluates the set of proposals to update the UDP transport behaviour and other alternatives intended to improve support for tunnel protocols. It concludes by assessing the trade-offs of the various methods, identifying advantages and disadvantages for each method.

#### 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].

#### 1.3. Use of UDP Tunnels

One increasingly popular use of UDP is as a tunneling protocol, where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by the middleboxes that may exist along the path,

because many middleboxes support transmission using UDP. In this use, the receiving endpoint decapsulates the UDP datagrams and forwards the original packets contained in the payload [RFC5405]. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks.

#### 1.3.1. Motivation for new approaches

A number of tunnel encapsulations deployed over IPv4 have used the UDP transport with a zero checksum. Users of these protocols expect a similar solution for IPv6.

A number of tunnel protocols are also currently being defined (e.g. Automated Multicast Tunnels, AMT [I-D.ietf-mboned-auto-multicast], and the Locator/Identifier Separation Protocol, LISP [LISP]). These protocols motivated an update to IPv6 UDP checksum processing to benefit from simpler checksum processing for various reasons:

- o Reducing forwarding costs, motivated by redundancy present in the encapsulated packet header, since in tunnel encapsulations, payload integrity and length verification may be provided by higher layer encapsulations (often using the IPv4, UDP, UDP-Lite, or TCP checksums).
- o Eliminating a need to access the entire packet when forwarding the packet by a tunnel endpoint.
- o Enhancing ability to traverse and function with middleboxes.
- o A desire to use the port number space to enable load-sharing.

#### 1.3.2. Reducing forwarding cost

It is a common requirement to terminate a large number of tunnels on a single router/host. The processing cost per tunnel includes both state (memory requirements) and per-packet processing at the tunnel ingress and egress.

Automatic IP Multicast Tunneling, known as AMT [I-D.ietf-mboned-auto-multicast] currently specifies UDP as the transport protocol for packets carrying tunneled IP multicast packets. The current specification for AMT states that the UDP checksum in the outer packet header should be zero (see Section 6.6 of [I-D.ietf-mboned-auto-multicast]). This argues that the computation of an additional checksum is an unwarranted burden on nodes implementing lightweight tunneling protocols when an inner packet is already adequately protected, . The AMT protocol needs to

replicate a multicast packet to each gateway tunnel. In this case, the outer IP addresses are different for each tunnel and therefore require a different pseudo header to be built for each UDP replicated encapsulation.

The argument concerning redundant processing costs is valid regarding the integrity of a tunneled packet. In some architectures (e.g. PC-based routers), other mechanisms may also significantly reduce checksum processing costs: There are implementations that have optimised checksum processing algorithms, including the use of checksum-offloading. This processing is readily available for IPv4 packets at high line rates. Such processing may be anticipated for IPv6 endpoints, allowing receivers to reject corrupted packets without further processing. However, there are certain classes of tunnel end-points where this off-loading is not available and unlikely to become available in the near future.

#### 1.3.3. Need to inspect the entire packet

The currently-deployed hardware in many routers uses a fast-path processing that only provides the first  $n$  bytes of a packet to the forwarding engine, where typically  $n \leq 128$ .

When this design is used to support a tunnel ingress and egress, it prevents fast processing of a transport checksum over an entire (large) packet. Hence the currently defined IPv6 UDP checksum is poorly suited to use within a router that is unable to access the entire packet and does not provide checksum-offloading. Thus enabling checksum calculation over the complete packet can impact router design, performance improvement, energy consumption and/or cost.

#### 1.3.4. Interactions with middleboxes

Many paths in the Internet include one or more middleboxes of various types. There exist large classes of middleboxes that will handle zero UDP checksum packets, which would not support UDP-Lite or the other investigated proposals. These middleboxes includes load balancers (see Section 1.3.5) including Equal Cost Multipath Routing, traffic classifiers and other functions that reads some fields in the UDP headers but does not validate the UDP checksum.

There are also middleboxes that either validates or modify the UDP checksum. The two most common classes are Firewalls and NATs. In IPv4, UDP-encapsulation may be desirable for NAT traversal, since UDP support is commonly provided. It is also necessary due to the almost ubiquitous deployment of IPv4 NATs. There has also been discussion of NAT for IPv6, although not for the same reason as in IPv4. If

IPv6 NAT becomes a reality they hopefully do not present the same protocol issues as for IPv4. If NAT is defined for IPv6, it should take into consideration the use of a zero UDP checksum.

The requirements for IPv6 firewall traversal are likely to be similar to those for IPv4. In addition, it can be reasonably expected that a firewall conforming to RFC 2460 will not regard datagrams with a zero UDP checksum as valid. Use of a zero UDP checksum with IPv6 requires firewalls to be updated before the full utility of the change is available.

It can be expected that datagrams with zero UDP checksum will initially not have the same middlebox traversal characteristics as regular UDP (RFC 2460). However when implementations follow the requirements specified in this document, we expect the traversal capabilities to improve over time. We also note that deployment of IPv6-capable middleboxes is still in its initial phases. Thus, it might be that the number of non-updated boxes quickly become a very small percentage of the deployed middleboxes.

#### 1.3.5. Support for load balancing

The UDP port number fields have been used as a basis to design load-balancing solutions for IPv4. This approach has also been leveraged for IPv6. An alternate method would be to utilise the IPv6 Flow Label [RFC6437] as a basis for entropy for load balancing. This would have the desirable effect of releasing IPv6 load-balancing devices from the need to assume semantics for the use of the transport port field and also works for all type of transport protocols.

This use of the flow-label for load balancing is consistent with the intended use, although further clarity was needed to ensure the field can be consistently used for this purpose, therefore an updated IPv6 Flow Label [RFC6437] and Equal-Cost Multi-Path routing usage, (ECMP) [RFC6438] was produced. Router vendors could be encouraged to start using the IPv6 Flow Label as a part of the flow hash, providing support for ECMP without requiring use of UDP.

However, the method for populating the outer IPv6 header with a value for the flow label is not trivial: If the inner packet uses IPv6, then the flow label value could be copied to the outer packet header. However, many current end-points set the flow label to a zero value (thus no entropy). The ingress of a tunnel seeking to provide good entropy in the flow label field would therefore need to create a random flow label value and keep corresponding state, so that all packets that were associated with a flow would be consistently given the same flow label. Although possible, this complexity may not be

desirable in a tunnel ingress.

The end-to-end use of flow labels for load balancing is a long-term solution. Even if the usage of the flow label is clarified, there would be a transition time before a significant proportion of end-points start to assign a good quality flow label to the flows that they originate, with continued use of load balancing using the transport header fields until any widespread deployment is finally achieved.

## 2. Standards-Track Transports

The IETF has defined a set of transport protocols that may be applicable for tunnels with IPv6. There are also a set of network layer encapsulation tunnels such as IP-in-IP and GRE. These already standardized solutions are discussed here prior to the issues, as background for the issue description and some comparison of where the issue may already occur.

### 2.1. UDP with Standard Checksum

UDP [RFC0768] with standard checksum behaviour, as defined in RFC 2460, has already been discussed. UDP usage guidelines are provided in [RFC5405].

### 2.2. UDP-Lite

UDP-Lite [RFC3828] offers an alternate transport to UDP, specified as a proposed standard, RFC 3828. A MIB is defined in [RFC5097] and unicast usage guidelines in [RFC5405]. There is at least one open source implementation as a part of the Linux kernel since version 2.6.20.

UDP-Lite provides a checksum with optional partial coverage. When using this option, a datagram is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). When the checksum covers the entire packet, UDP-Lite is fully equivalent with UDP, with the exception that it uses a different value in the Next Header field in the IPv6 header. Errors/corruption in the insensitive part will not cause the datagram to be discarded by the transport layer at the receiving endpoint. A minor side-effect of using UDP-Lite is that this was specified for damage-tolerant payloads and some link-layers may employ different link encapsulations when forwarding UDP-Lite segments (e.g. radio access bearers). Most link-layers will cover the insensitive part with the same strong layer 2 frame CRC that covers the sensitive part.

#### 2.2.1. Using UDP-Lite as a Tunnel Encapsulation

Tunnel encapsulations can use UDP-Lite (e.g. Control And Provisioning of Wireless Access Points, CAPWAP [RFC5415]), since UDP-Lite provides a transport-layer checksum, including an IP pseudo header checksum, in IPv6, without the need for a router/middlebox to traverse the entire packet payload. This provides most of the verification required for delivery and still keeps a low complexity for the checksumming operation. UDP-Lite may set the length of checksum coverage on a per packet basis. This feature could be used if a tunnel protocol is designed to only verify delivery of the tunneled payload and uses a calculated checksum for control information.

There is currently poor support for middlebox traversal using UDP-Lite, because UDP-Lite uses a different IPv6 network-layer Next Header value to that of UDP, and few middleboxes are able to interpret UDP-Lite and take appropriate actions when forwarding the packet. This makes UDP-Lite less suited to protocols needing general Internet support, until such time that UDP-Lite has achieved better support in middleboxes and end-points.

#### 2.3. General Tunnel Encapsulations

The IETF has defined a set of tunneling protocols or network layer encapsulations, e.g., IP-in-IP and GRE. These either do not include a checksum or use a checksum that is optional, since tunnel encapsulations are typically layered directly over the Internet layer (identified by the upper layer type in the IPv6 Next Header field) and are also not used as endpoint transport protocols. There is little chance of confusing a tunnel-encapsulated packet with other application data that could result in corruption of application state or data.

From the end-to-end perspective, the principal difference is that the network-layer Next Header field identifies a separate transport, which reduces the probability that corruption could result in the packet being delivered to the wrong endpoint or application. Specifically, packets are only delivered to protocol modules that process a specific Next Header value. The Next Header field therefore provides a first-level check of correct demultiplexing. In contrast, the UDP port space is shared by many diverse applications and therefore UDP demultiplexing relies solely on the port numbers.

#### 2.4. Relation to UDP-Lite and UDP with checksum

The operation of IPv6 with UDP with a zero-checksum is not the same as IPv4 with UDP with a zero-checksum. Protocol designers should not

be fooled into thinking the two are the same. The requirements below list a set of additional considerations.

Where possible, existing general tunnel encapsulations, such as GRE, IP-in-IP, should be used. This section assumes that such existing tunnel encapsulations do not offer the functionality required to satisfy the protocol designer's goals. The section considers the standardized alternative solutions, rather than the full set of ideas evaluated in Appendix A. The alternatives to UDP with a zero checksum are UDP with a (calculated) checksum, and UDP-Lite.

UDP with a checksum has the advantage of close to universal support in both endpoints and middleboxes. It also provides statistical verification of delivery to the intended destination (address and port). However, some classes of device have limited support for calculation of a checksum that covers a full datagram. For these devices, this can incur significant processing cost (e.g. requiring processing in the router slow-path) and can hence reduce capacity or fail to function.

UDP-Lite has the advantage of using a checksum that is calculated only over the pseudo header and the UDP header. This provides a statistical verification of delivery to the intended destination (address and port). The checksum can be calculated without access to the datagram payload, only requiring access to the part to be protected. A drawback is that UDP-Lite has currently limited support in both end-points (i.e. is not supported on all operating system platforms) and middleboxes (that require support for the UDP-Lite header type). A path verification method is therefore recommended.

IPv6 and UDP with a zero-checksum can also be used by nodes that do not permit calculation of a payload checksum. Many existing classes of middleboxes do not verify or change the transport checksum. For these middleboxes, IPv6 with a zero UDP checksum is expected to function where UDP-Lite would not. However, support for the zero UDP checksum in middleboxes that do change or verify the checksum is currently limited, and this may result in datagrams with a zero UDP checksum being discarded, therefore a path verification method is recommended.

There are sets of constraints for which no solution exist: A protocol designer that needs to originate or receive datagrams on a device that can not efficiently calculate a checksum over a full datagram and also needs these packets to pass through a middlebox that verifies or changes a UDP checksum, but does not support a zero UDP checksum, can not use the zero UDP checksum method. Similarly, one that originates datagrams on a device with UDP-Lite support, but needs the packets to pass through a middlebox that does not support



UDP-Lite, can not use UDP-Lite. For such cases, there is no optimal solution and the current recommendation is to use or fall-back to using UDP with full checksum coverage.

### 3. Issues Requiring Consideration

This informative section evaluates issues around the proposal to update IPv6 [RFC2460], to enable the UDP transport checksum to be set to zero. Some of the identified issues are shared with other protocols already in use. The section also provides background to the requirements and recommendations that follow.

The decision in RFC 2460 to omit an integrity check at the network level meant that the IPv6 transport checksum was overloaded with many functions, including validating:

- o the endpoint address was not corrupted within a router, i.e., a packet was intended to be received by this destination and validate that the packet does not consist of a wrong header spliced to a different payload;
- o that extension header processing is correctly delimited - i.e., the start of data has not been corrupted. In this case, reception of a valid Next Header value provides some protection;
- o reassembly processing, when used;
- o the length of the payload;
- o the port values - i.e., the correct application receives the payload (applications should also check the expected use of source ports/addresses);
- o the payload integrity.

In IPv4, the first four checks are performed using the IPv4 header checksum.

In IPv6, these checks occur within the endpoint stack using the UDP checksum information. An IPv6 node also relies on the header information to determine whether to send an ICMPv6 error message [RFC4443] and to determine the node to which this is sent. Corrupted information may lead to misdelivery to an unintended application socket on an unexpected host.

### 3.1. Effect of packet modification in the network

IP packets may be corrupted as they traverse an Internet path. Older evidence in "When the CRC and TCP Checksum Disagree" [Sigcomm2000] show that this was once an issue in year 2000 with IPv4 routers, and occasional corruption could result from bad internal router processing in routers or hosts. These errors are not detected by the strong frame checksums employed at the link-layer [RFC3819]. During the development of this document in 2009, individuals provided reports of observed rates for received UDP datagrams using IPv4 where the UDP checksum had been detected as corrupt. These rates were as high as 1.39E-4 for some paths, but also close to zero for some other paths.

There is extensive experience of deployment using tunnel protocols in well-managed networks (e.g. corporate networks or service provider core networks). This has shown the robustness of methods such as PWE and MPLS that do not employ a transport protocol checksum and have not specified mechanisms to protect from corruption of the unprotected headers (such as the VPN Identifier in MPLS). Reasons for the robustness may include:

- o A reduced probability of corruption on paths through well-managed networks.
- o IP forms the majority of the inner traffic carried by these tunnels. Hence from a transport perspective, endpoint verification is already being performed when processing a received IPv4 packet or by the transport pseudo-header for an IPv6 packet. This update to UDP does not change this behaviour.
- o In certain cases, a combination of additional filtering (e.g. filter of a MAC destination address in a L2 tunnel) significantly reduces the probability of final mis-delivery to the IP stack.
- o The tunnel protocols did not use a UDP transport header, any corruption is therefore unlikely to result in misdelivery to another UDP-based application. This concern is specific to the use of UDP with IPv6.

While this experience can guide the present recommendations, any update to UDP must preserve operation in the general Internet. This is heterogeneous and can include links and systems of very varying characteristics. Transport protocols used by hosts need to be designed with this in mind, especially when there is need to traverse edge networks, where middlebox deployments are common.

For the general Internet, there is no current evidence that

corruption is rare, nor that this may not be applicable to IPv6. It therefore seems prudent not to relax checks on misdelivery. The emergence of low-end IPv6 routers and the proposed use of NAT with IPv6 further motivate the need to protect from misdelivery.

Corruption in the network may result in:

- o A datagram being misdelivered to the wrong host/router or the wrong transport entity within an endpoint. Such a datagram needs to be discarded;
- o A datagram payload being corrupted, but still delivered to the intended host/router transport entity. Such a datagram needs to be either discarded or correctly processed by an application that provides its own integrity checks;
- o A datagram payload being truncated by corruption of the length field. Such a datagram needs to be discarded.

When a checksum is used, this significantly reduces the impact of errors, reducing the probability of undetected corruption of state (and data) on both the host stack and the applications using the transport service.

The following sections examine the impact of modifying each of these header fields.

#### 3.1.1. Corruption of the destination IP address

An IPv6 endpoint destination address could be modified in the network (e.g. corrupted by an error). This is not a concern for IPv4, because the IP header checksum will result in this packet being discarded by the receiving IP stack. Such modification in the network can not be detected at the network layer when using IPv6. Detection of this corruption by a UDP receiver relies on the IPv6 pseudo header incorporated in the transport checksum.

There are two possible outcomes:

- o Delivery to a destination address that is not in use (the packet will not be delivered, but could result in an error report);
- o Delivery to a different destination address. This modification will normally be detected by the transport checksum, resulting in silent discard. Without a computed checksum, the packet would be passed to the endpoint port demultiplexing function. If an application is bound to the associated ports, the packet payload will be passed to the application (see the subsequent section on

port processing).

### 3.1.2. Corruption of the source IP address

This section examines what happens when the source address is corrupted in transit. This is not a concern in IPv4, because the IP header checksum will normally result in this packet being discarded by the receiving IP stack. Detection of this corruption by a UDP receiver relies on the IPv6 pseudo header incorporated in the transport checksum.

Corruption of an IPv6 source address does not result in the IP packet being delivered to a different endpoint protocol or destination address. If only the source address is corrupted, the datagram will likely be processed in the intended context, although with erroneous origin information. When using Unicast Reverse Path Forwarding [RFC2827], a change in address may result in the router discarding the packet when the route to the modified source address is different to that of the source address of the original packet.

The result will depend on the application or protocol that processes the packet. Some examples are:

- o An application that requires a per-established context may disregard the datagram as invalid, or could map this to another context (if a context for the modified source address was already activated).
- o A stateless application will process the datagram outside of any context, a simple example is the ECHO server, which will respond with a datagram directed to the modified source address. This would create unwanted additional processing load, and generate traffic to the modified endpoint address.
- o Some datagram applications build state using the information from packet headers. A previously unused source address would result in receiver processing and the creation of unnecessary transport-layer state at the receiver. For example, Real Time Protocol (RTP) [RFC3550] sessions commonly employ a source independent receiver port. State is created for each received flow. Reception of a datagram with a corrupted source address will therefore result in accumulation of unnecessary state in the RTP state machine, including collision detection and response (since the same synchronization source, SSRC, value will appear to arrive from multiple source IP addresses).
- o ICMP messages relating to a corrupted packet can be misdirected to the wrong source node.

In general, the effect of corrupting the source address will depend upon the protocol that processes the packet and its robustness to this error. For the case where the packet is received by a tunnel endpoint, the tunnel application is expected to correctly handle a corrupted source address.

The impact of source address modification is more difficult to quantify when the receiving application is not that originally intended and several fields have been modified in transit.

### 3.1.3. Corruption of Port Information

This section describes what happens if one or both of the UDP port values are corrupted in transit. This can also happen with IPv4 is used with a zero UDP checksum, but not when UDP checksums are calculated or when UDP-Lite is used. If the ports carried in the transport header of an IPv6 packet were corrupted in transit, packets may be delivered to the wrong application process (on the intended machine) and/or responses or errors sent to the wrong application process (on the intended machine).

### 3.1.4. Delivery to an unexpected port

If one combines the corruption effects, such as destination address and ports, there is a number of potential outcomes when traffic arrives at an unexpected port. This section discusses these possibilities and their outcomes for a packet that does not use the UDP checksum validation:

- o Delivery to a port that is not in use. The packet is discarded, but could generate an ICMPv6 message (e.g. port unreachable).
- o It could be delivered to a different node that implements the same application, where the packet may be accepted, generating side-effects or accumulated state.
- o It could be delivered to an application that does not implement the tunnel protocol, where the packet may be incorrectly parsed, and may be misinterpreted, generating side-effects or accumulated state.

The probability of each outcome depends on the statistical probability that the address or the port information for the source or destination becomes corrupt in the datagram such that they match those of an existing flow or server port. Unfortunately, such a match may be more likely for UDP than for connection-oriented transports, because:

1. There is no handshake prior to communication and no sequence numbers (as in TCP, DCCP, or SCTP). Together, this makes it hard to verify that an application process is given only the application data associated with a specific transport session.
2. Applications writers often bind to wild-card values in endpoint identifiers and do not always validate correctness of datagrams they receive (guidance on this topic is provided in [RFC5405]).

While these rules could, in principle, be revised to declare naive applications as "Historic". This remedy is not realistic: the transport owes it to the stack to do its best to reject bogus datagrams.

If checksum coverage is suppressed, the application therefore needs to provide a method to detect and discard the unwanted data. A tunnel protocol would need to perform its own integrity checks on any control information if transported in datagrams with a zero UDP checksum. If the tunnel payload is another IP packet, the packets requiring checksums can be assumed to have their own checksums provided that the rate of corrupted packets is not significantly larger due to the tunnel encapsulation. If a tunnel transports other inner payloads that do not use IP, the assumptions of corruption detection for that particular protocol must be fulfilled, this may require an additional checksum/CRC and/or integrity protection of the payload and tunnel headers.

A protocol that uses a zero UDP checksum can not assume that it is the only protocol using a zero UDP checksum. Therefore, it needs to gracefully handle misdelivery. It must be robust to reception of malformed packets received on a listening port and expect that these packets may contain corrupted data or data associated with a completely different protocol.

#### 3.1.5. Corruption of Fragmentation Information

The fragmentation information in IPv6 employs a 32-bit identity field, compared to only a 16-bit field in IPv4, a 13-bit fragment offset and a 1-bit flag, indicating if there are more fragments. Corruption of any of these field may result in one of two outcomes:

Reassembly failure: An error in the "More Fragments" field for the last fragment will for example result in the packet never being considered complete and will eventually be timed out and discarded. A corruption in the ID field will result in the fragment not being delivered to the intended context thus leaving the rest incomplete, unless that packet has been duplicated prior to corruption. The incomplete packet will eventually be timed out

and discarded.

Erroneous reassembly: The re-assembled packet did not match the original packet. This can occur when the ID field of a fragment is corrupted, resulting in a fragment becoming associated with another packet and taking the place of another fragment. Corruption in the offset information can cause the fragment to be misaligned in the reassembly buffer, resulting in incorrect reassembly. Corruption can cause the packet to become shorter or longer, however completion of reassembly is much less probable, since this would require consistent corruption of the IPv6 headers payload length field and the offset field. The possibility of mis-assembly requires the reassembling stack to provide strong checks that detect overlap or missing data, note however that this is not guaranteed and has been clarified in "Handling of Overlapping IPv6 Fragments" [RFC5722].

The erroneous reassembly of packets is a general concern and such packets should be discarded instead of being passed to higher layer processes. The primary detector of packet length changes is the IP payload length field, with a secondary check by the transport checksum. The Upper-Layer Packet length field included in the pseudo header assists in verifying correct reassembly, since the Internet checksum has a low probability of detecting insertion of data or overlap errors (due to misplacement of data). The checksum is also incapable of detecting insertion or removal of all zero-data that occurs in a multiple of a 16-bit chunk.

The most significant risk of corruption results following mis-association of a fragment with a different packet. This risk can be significant, since the size of fragments is often the same (e.g. fragments resulting when the path MTU results in fragmentation of a larger packet, common when addition of a tunnel encapsulation header expands the size of a packet). Detection of this type of error requires a checksum or other integrity check of the headers and the payload. Such protection is anyway desirable for tunnel encapsulations using IPv4, since the small fragmentation ID can easily result in wrap-around [RFC4963], this is especially the case for tunnels that perform flow aggregation [I-D.ietf-intarea-tunnels].

Tunnel fragmentation behavior matters. There can be outer or inner fragmentation "Tunnels in the Internet Architecture" [I-D.ietf-intarea-tunnels]. If there is inner fragmentation by the tunnel, the outer headers will never be fragmented and thus a zero UDP checksum in the outer header will not affect the reassembly process. When a tunnel performs outer header fragmentation, the tunnel egress needs to perform reassembly of the outer fragments into an inner packet. The inner packet is either a complete packet or a

fragment. If it is a fragment, the destination endpoint of the fragment will perform reassembly of the received fragments. The complete packet or the reassembled fragments will then be processed according to the packet Next Header field. The receiver may only detect reassembly anomalies when it uses a protocol with a checksum. The larger the number of reassembly processes to which a packet has been subjected, the greater the probability of an error.

- o An IP-in-IP tunnel that performs inner fragmentation has similar properties to a UDP tunnel with a zero UDP checksum that also performs inner fragmentation.
- o An IP-in-IP tunnel that performs outer fragmentation has similar properties to a UDP tunnel with a zero UDP checksum that performs outer fragmentation.
- o A tunnel that performs outer fragmentation can result in a higher level of corruption due to both inner and outer fragmentation, enabling more chances for reassembly errors to occur.
- o Recursive tunneling can result in fragmentation at more than one header level, even for inner fragmentation unless it goes to the inner-most IP header.
- o Unless there is verification at each reassembly, the probability for undetected error will increase with the number of times fragmentation is recursively applied, making IP-in-IP and UDP with zero UDP checksum both vulnerable to undetected errors.

In conclusion, fragmentation of datagrams with a zero UDP checksum does not worsen the performance compared to some other commonly used tunnel encapsulations. However, caution is needed for recursive tunneling without any additional verification at the different tunnel layers.

### 3.2. Where Packet Corruption Occurs

Corruption of IP packets can occur at any point along a network path, during packet generation, during transmission over the link, in the process of routing and switching, etc. Some transmission steps include a checksum or Cyclic Redundancy Check (CRC) that reduces the probability for corrupted packets being forwarded, but there still exists a probability that errors may propagate undetected.

Unfortunately the community lacks reliable information to identify the most common functions or equipment that result in packet corruption. However, there are indications that the place where corruption occurs can vary significantly from one path to another.



There is therefore a risk in applying evidence from one domain of usage to infer characteristics for another. Methods intended for general Internet usage must therefore assume that corruption can occur and deploy mechanisms to mitigate the effect of corruption and/or resulting misdelivery.

### 3.3. Validating the network path

IP transports designed for use in the general Internet should not assume specific path characteristics. Network protocols may reroute packets that change the set of routers and middleboxes along a path. Therefore transports such as TCP, SCTP and DCCP have been designed to negotiate protocol parameters, adapt to different network path characteristics, and receive feedback to verify that the current path is suited to the intended application. Applications using UDP and UDP-Lite need to provide their own mechanisms to confirm the validity of the current network path.

A zero value in the UDP checksum field is explicitly disallowed in RFC2460. Thus it may be expected that any device on the path that has a reason to look beyond the IP header, for example to validate the UDP checksum, will consider such a packet as erroneous or illegal and may discard it, unless the device is updated to support the new behavior. Any middlebox that modifies the UDP checksum, for example a NAT that changes the values of the IP and UDP header in such a way that the checksum over the pseudo header changes value, will need to be updated to support this behavior. Until then, a zero UDP checksum packet is likely to be discarded either directly in the middlebox or at the destination, when a zero UDP checksum has been modified to a non-zero by an incremental update.

A pair of end-points intending to use a new behavior will therefore not only need to ensure support at each end-point, but also that the path between them will deliver packets with the new behavior. This may require using negotiation or an explicit mandate to use the new behavior by all nodes that support the new protocol.

Enabling the use of a zero checksum places new requirements on equipment deployed within the network, such as middleboxes. A middlebox (e.g. Firewalls, Network Address Translators) may enable zero checksum usage for a particular range of ports. Note that checksum off-loading and operating system design may result in all IPv6 UDP traffic being sent with a calculated checksum. This requires middleboxes that are configured to enable a zero UDP checksum to continue to work with bidirectional UDP flows that use a zero UDP checksum in only one direction, and therefore they must not maintain separate state for a UDP flow based on its checksum usage.

Support along the path between end points can be guaranteed in limited deployments by appropriate configuration. In general, it can be expected to take time for deployment of any updated behaviour to become ubiquitous.

A sender will need to probe the path to verify the expected behavior. Path characteristics may change, and usage therefore should be robust and able to detect a failure of the path under normal usage and re-negotiate. Note that a bidirectional path does not necessarily support the same checksum usage in both the forward and return directions: Receipt of a datagram with a zero UDP checksum, does not imply that the remote endpoint can also receive a datagram with a zero UDP checksum. This will require periodic validation of the path, adding complexity to any solution using the new behavior.

#### 3.4. Applicability of method

The update to the IPv6 specification defined in [I-D.ietf-6man-udpchecksums] only modifies IPv6 nodes that implement specific protocols designed to permit omission of a UDP checksum. This document therefore provides an applicability statement for the updated method indicating when the mechanism can (and can not) be used. Enabling this, and ensuring correct interactions with the stack, implies much more than simply disabling the checksum algorithm for specific packets at the transport interface.

When the method is widely available, it may be expected to be used by applications that are perceived to gain benefit. Any solution that uses an end-to-end transport protocol, rather than an IP-in-IP encapsulation, needs to minimise the possibility that application processes could confuse a corrupted or wrongly delivered UDP datagram with that of data addressed to the application running on their endpoint.

The protocol or application that uses the zero checksum method must ensure that the lack of checksum does not affect the protocol operation. This includes being robust to receiving a unintended packet from another protocol or context following corruption of a destination or source address and/or port value. It also includes considering the need for additional implicit protection mechanisms required when using the payload of a UDP packet received with a zero checksum.

#### 3.5. Impact on non-supporting devices or applications

It is important to consider the potential impact of using a zero UDP checksum on end-point devices or applications that are not modified to support the new behavior or by default or preference, use the

regular behavior. These applications must not be significantly impacted by the update.

To illustrate why this necessary, consider the implications of a node that enables use of a zero UDP checksum at the interface level: This would result in all applications that listen to a UDP socket receiving datagrams where the checksum was not verified. This could have a significant impact on an application that was not designed with the additional robustness needed to handle received packets with corruption, creating state or destroying existing state in the application.

A zero UDP checksum therefore needs to be enabled only for individual ports using an explicit request by the application. In this case, applications using other ports would maintain the current IPv6 behavior, discarding incoming datagrams with a zero UDP checksum. These other applications would not be affected by this changed behavior. An application that allows the changed behavior should be aware of the risk of corruption and the increased level of misdirected traffic, and can be designed robustly to handle this risk.

#### 4. Constraints on implementation of IPv6 nodes supporting zero checksum

This section is an applicability statement that defines requirements and recommendations on the implementation of IPv6 nodes that support use of a zero value in the checksum field of a UDP datagram.

All implementations that support this zero UDP checksum method **MUST** conform to the requirements defined below.

1. An IPv6 sending node **MAY** use a calculated RFC 2460 checksum for all datagrams that it sends. This explicitly permits an interface that supports checksum offloading to insert an updated UDP checksum value in all UDP datagrams that it forwards, however note that sending a calculated checksum requires the receiver to also perform the checksum calculation. Checksum offloading can normally be switched off for a particular interface to ensure that datagrams are sent with a zero UDP checksum.
2. IPv6 nodes **SHOULD** by default **NOT** allow the zero UDP checksum method for transmission.
3. IPv6 nodes **MUST** provide a way for the application/protocol to indicate the set of ports that will be enabled to send datagrams with a zero UDP checksum. This may be implemented by enabling a

transport mode using a socket API call when the socket is established, or a similar mechanism. It may also be implemented by enabling the method for a pre-assigned static port used by a specific tunnel protocol.

4. IPv6 nodes MUST provide a method to allow an application/protocol to indicate that a particular UDP datagram is required to be sent with a UDP checksum. This needs to be allowed by the operating system at any time (e.g. to send keep-alive datagrams), not just when a socket is established in the zero checksum mode.
5. The default IPv6 node receiver behaviour MUST discard all IPv6 packets carrying datagrams with a zero UDP checksum.
6. IPv6 nodes MUST provide a way for the application/protocol to indicate the set of ports that will be enabled to receive datagrams with a zero UDP checksum. This may be implemented via a socket API call, or similar mechanism. It may also be implemented by enabling the method for a pre-assigned static port used by a specific tunnel protocol.
7. IPv6 nodes supporting usage of zero UDP checksums MUST also allow reception using a calculated UDP checksum on all ports configured to allow zero UDP checksum usage. (The sending endpoint, e.g. encapsulating ingress, may choose to compute the UDP checksum, or may calculate this by default.) The receiving endpoint MUST use the reception method specified in RFC2460 when the checksum field is not zero.
8. RFC 2460 specifies that IPv6 nodes SHOULD log received datagrams with a zero UDP checksum. This remains the case for any datagram received on a port that does not explicitly enable processing of a zero UDP checksum. A port for which the zero UDP checksum has been enabled MUST NOT log the datagram solely because the checksum value is zero.
9. IPv6 nodes MAY separately identify received UDP datagrams that are discarded with a zero UDP checksum. It SHOULD NOT add these to the standard log, since the endpoint has not been verified. This may be used to support other functions (such as a security policy).
10. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum MUST provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and

port numbers in the ICMPv6 message body).

#### 5. Requirements on usage of the zero UDP checksum

This section is an applicability statement that identifies requirements and recommendations for protocols and tunnel encapsulations that are transported over an IPv6 transport flow (e.g. tunnel) that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints. Before deciding to use the zero UDP checksum and loose the integrity verification provided, a protocol developer should seriously consider if they can use checksummed UDP packets or UDP-Lite [RFC3828], because IPv6 with a zero UDP checksum is not equivalent in behavior to IPv4 with zero UDP checksum.

The requirements and recommendations for protocols and tunnel encapsulations using an IPv6 transport flow that does not perform a UDP checksum calculation to verify the integrity at the transport endpoints are:

1. Transported protocols that enable the use of zero UDP checksum MUST only enable this for a specific port or port-range. This needs to be enabled at the sending and receiving endpoints for a UDP flow.
2. An integrity mechanism is always RECOMMENDED at the transported protocol layer to ensure that corruption rates of the delivered payload is not increased (e.g. the inner-most packet of a UDP tunnel). A mechanism that isolates the causes of corruption (e.g. identifying misdelivery, IPv6 header corruption, tunnel header corruption) is expected to also provide additional information about the status of the tunnel (e.g. to suggest a security attack).
3. A transported protocol that encapsulates Internet Protocol (IPv4 or IPv6) packets MAY rely on the inner packet integrity checks, provided that the tunnel protocol will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel protocol MUST provide an additional integrity verification mechanism. Early detection is desirable to avoid wasting unnecessary computation, transmission capacity or storage for packets that will subsequently be discarded.
4. A transported protocol that supports use of a zero UDP checksum, MUST be designed so that corruption of this information does not result in accumulated state for the protocol.

5.    A transported protocol with a non-tunnel payload or one that encapsulates non-IP packets **MUST** have a CRC or other mechanism for checking packet integrity, unless the non-IP packet is specifically designed for transmission over a lower layer that does not provide a packet integrity guarantee.
6.    A transported protocol with control feedback **SHOULD** be robust to changes in the network path, since the set of middleboxes on a path may vary during the life of an association. The UDP endpoints need to discover paths with middleboxes that drop packets with a zero UDP checksum. Therefore, transported protocols **SHOULD** send keep-alive messages with a zero UDP checksum. An endpoint that discovers an appreciable loss rate for keep-alive packets **MAY** terminate the UDP flow (e.g. tunnel). Section 3.1.3 of RFC 5405 describes requirements for congestion control when using a UDP-based transport.
7.    A protocol with control feedback that can fall-back to using UDP with a calculated RFC 2460 checksum is expected to be more robust to changes in the network path. Therefore, keep-alive messages **SHOULD** include both UDP datagrams with a checksum and datagrams with a zero UDP checksum. This will enable the remote endpoint to distinguish between a path failure and dropping of datagrams with a zero UDP checksum.
8.    A middlebox implementation **MUST** allow forwarding of an IPv6 UDP datagram with both a zero and standard UDP checksum using the same UDP port.
9.    A middlebox **MAY** configure a restricted set of specific port ranges that forward UDP datagrams with a zero UDP checksum. The middlebox **MAY** drop IPv6 datagrams with a zero UDP checksum that are outside a configured range.
10.   When a middlebox forwards an IPv6 UDP flow containing datagrams with both a zero and standard UDP checksum, the middlebox **MUST** NOT maintain separate state for flows depending on the value of their UDP checksum field. (This requirement is necessary to enable a sender that always calculates a checksum to communicate via a middlebox with a remote endpoint that uses a zero UDP checksum.)

Special considerations are required when designing a UDP tunnel protocol, where the tunnel ingress or egress may be a router that may not have access to the packet payload. When the node is acting as a host (i.e., sending or receiving a packet addressed to itself), the checksum processing is similar to other hosts. However, when the node (e.g. a router) is acting as a tunnel ingress or egress that

forwards a packet to or from a UDP tunnel, there may be restricted access to the packet payload. This prevents calculating (or verifying) a UDP checksum. In this case, the tunnel protocol may use a zero UDP checksum and must:

- o Ensure that tunnel ingress and tunnel egress router are both configured to use a zero UDP checksum. For example, this may include ensuring that hardware checksum offloading is disabled.
- o The tunnel operator must ensure that middleboxes on the network path are updated to support use of a zero UDP checksum.
- o A tunnel egress should implement appropriate security techniques to protect from overload, including source address filtering to prevent traffic injection by an attacker, and rate-limiting of any packets that incur additional processing, such as UDP datagrams used for control functions that require verification of a calculated checksum to verify the network path. Usage of common control traffic for multiple tunnels between a pair of nodes can assist in reducing the number of packets to be processed.

## 6. Summary

This document provides an applicability statement for the use of UDP transport checksums with IPv6.

It examines the role of the UDP transport checksum when used with IPv6 and presents a summary of the trade-offs in evaluating the safety of updating RFC 2460 to permit an IPv6 endpoint to use a zero UDP checksum field to indicate that no checksum is present.

Application designers should first examine whether their transport goals may be met using standard UDP (with a calculated checksum) or by using UDP-Lite. The use of UDP with a zero UDP checksum has merits for some applications, such as tunnel encapsulation, and is widely used in IPv4. However, there are different dangers for IPv6: There is an increased risk of corruption and misdelivery when using zero UDP checksum in IPv6 compared to using IPv4 due to the lack of an IPv6 header checksum. Thus, applications need to evaluate the risks of enabling use of a zero UDP checksum and consider a solution that at least provides the same delivery protection as for IPv4, for example by utilizing UDP-Lite, or by enabling the UDP checksum. The use of checksum off-loading may help alleviate the cost of checksum processing and permit use of a checksum using method defined in RFC 2460.

Tunnel applications using UDP for encapsulation can in many cases use

a zero UDP checksum without significant impact on the corruption rate. A well-designed tunnel application should include consistency checks to validate the header information encapsulated with a received packet. In most cases, tunnels encapsulating IP packets can rely on the integrity protection provided by the transported protocol (or tunneled inner packet). When correctly implemented, such an endpoint will not be negatively impacted by omission of the transport-layer checksum. Recursive tunneling and fragmentation is a potential issue that can raise corruption rates significantly, and requires careful consideration.

Other UDP applications at the intended destination node or another node can be impacted if they are allowed to receive datagrams that have a zero UDP checksum. It is important that already deployed applications are not impacted by a change at the transport layer. If these applications execute on nodes that implement RFC 2460, they will discard (and log) all datagrams with a zero UDP checksum. This is not an issue.

In general, UDP-based applications need to employ a mechanism that allows a large percentage of the corrupted packets to be removed before they reach an application, both to protect the data stream of the application and the control plane of higher layer protocols. These checks are currently performed by the UDP checksum for IPv6, or the reduced checksum for UDP-Lite when used with IPv6.

The transport of recursive tunneling and the use of fragmentation pose difficult issues that need to be considered in the design of tunnel protocols. There is an increased risk of an error in the inner-most packet when fragmentation when several layers of tunneling and several different reassembly processes are run without verification of correctness. This requires extra thought and careful consideration in the design of transported tunnels.

Any use of the updated method must consider the implications on firewalls, NATs and other middleboxes. It is not expected that IPv6 NATs handle IPv6 UDP datagrams in the same way that they handle IPv4 UDP datagrams. In many deployed cases this will require an update to support an IPv6 zero UDP checksum. Firewalls are intended to be configured, and therefore may need to be explicitly updated to allow new services or protocols. IPv6 middlebox deployment is not yet as prolific as it is in IPv4, and therefore new devices are expected to follow the methods specified in this document.

Each application should consider the implications of choosing an IPv6 transport that uses a zero UDP checksum, and consider whether other standard methods may be more appropriate, and may simplify application design.



## 7. Acknowledgements

Brian Haberman, Brian Carpenter, Margaret Wasserman, Lars Eggert, others in the TSV directorate. Barry Leiba, Ronald Bonica, Pete Resnick, and Stewart Bryant are thanked for resulting in a document with much greater applicability. Thanks to P.F. Chimento for careful review and editorial corrections.

Thanks also to: Remi Denis-Courmont, Pekka Savola, Glen Turner, and many others who contributed comments and ideas via the 6man, behave, lisp and mboned lists.

## 8. IANA Considerations

This document does not require any actions by IANA.

## 9. Security Considerations

Transport checksums provide the first stage of protection for the stack, although they can not be considered authentication mechanisms. These checks are also desirable to ensure packet counters correctly log actual activity, and can be used to detect unusual behaviours.

Depending on the hardware design, the processing requirements may differ for tunnels that have a zero UDP checksum and those that calculate a checksum. This processing overhead may need to be considered when deciding whether to enable a tunnel and to determine an acceptable rate for transmission. This can become a security risk for designs that can handle a significantly larger number of packets with zero UDP checksums compared to datagrams with a non-zero checksum, such as tunnel egress. An attacker could attempt to inject non-zero checksummed UDP packets into a tunnel forwarding zero checksum UDP packets and cause overload in the processing of the non-zero checksums, e.g. if this happens in a routers slow path. Protection mechanisms should therefore be employed when this threat exists. Protection may include source address filtering to prevent an attacker injecting traffic, as well as throttling the amount of non-zero checksum traffic. The latter may impact the function of the tunnel protocol.

Transmission of IPv6 packets with a zero UDP checksum could reveal additional information to an on-path attacker to identify the operating system or configuration of a sending node. There is a need to probe the network path to determine whether the current path supports using IPv6 packets with a zero UDP checksum. The details of the probing mechanism may differ for different tunnel encapsulations

and if visible in the network (e.g. if not using IPsec in encryption mode) could reveal additional information to an on-path attacker to identify the type of tunnel being used.

IP-in-IP or GRE tunnels offer good traversal of middleboxes that have not been designed for security, e.g. firewalls. However, firewalls may be expected to be configured to block general tunnels as they present a large attack surface. This applicability statement therefore permits this method to be enabled only for specific ranges of ports.

When the zero UDP checksum mode is enabled for a range of ports, nodes and middleboxes must forward received UDP datagrams that have either a calculated checksum or a zero checksum.

## 10. References

### 10.1. Normative References

- [I-D.ietf-6man-udpchecksums]  
Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", draft-ietf-6man-udpchecksums-08 (work in progress), February 2013.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

### 10.2. Informative References

- [I-D.ietf-intarea-tunnels]  
Touch, J. and M. Townsley, "Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-00 (work in progress), March 2010.
- [I-D.ietf-mboned-auto-multicast]  
Bumgardner, G., "Automatic Multicast Tunneling", draft-ietf-mboned-auto-multicast-14 (work in progress),

June 2012.

- [LISP]        D. Farinacci et al, "Locator/ID Separation Protocol (LISP)", November 2012.
- [RFC1071]    Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.
- [RFC1141]    Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, January 1990.
- [RFC1624]    Rijssinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.
- [RFC2827]    Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC3550]    Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3819]    Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC3828]    Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.
- [RFC4443]    Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4963]    Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5097]    Renker, G. and G. Fairhurst, "MIB for the UDP-Lite protocol", RFC 5097, January 2008.
- [RFC5405]    Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5415]    Calhoun, P., Montemurro, M., and D. Stanley, "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol

Specification", RFC 5415, March 2009.

[RFC5722]    Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, December 2009.

[RFC6437]    Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6438]    Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011.

[Sigcomm2000]    Jonathan Stone and Craig Partridge , "When the CRC and TCP Checksum Disagree", 2000.

[UDPTT]    G Fairhurst, "The UDP Tunnel Transport mode", Feb 2010.

#### Appendix A.    Evaluation of proposal to update RFC 2460 to support zero checksum

This informative appendix documents the evaluation of the proposal to update IPv6 [RFC2460], to provide the option that some nodes may suppress generation and checking of the UDP transport checksum. It also compares the proposal with other alternatives, and notes that for a particular application some standard methods may be more appropriate than using IPv6 with a zero UDP checksum.

##### A.1.    Alternatives to the Standard Checksum

There are several alternatives to the normal method for calculating the UDP Checksum [RFC1071] that do not require a tunnel endpoint to inspect the entire packet when computing a checksum. These include (in decreasing order of complexity):

- o    Delta computation of the checksum from an encapsulated checksum field. Since the checksum is a cumulative sum [RFC1624], an encapsulating header checksum can be derived from the new pseudo header, the inner checksum and the sum of the other network-layer fields not included in the pseudo header of the encapsulated packet, in a manner resembling incremental checksum update [RFC1141]. This would not require access to the whole packet, but does require fields to be collected across the header, and arithmetic operations on each packet. The method would only work for packets that contain a 2's complement transport checksum (i.e., it would not be appropriate for SCTP or when IP fragmentation is used).

- o UDP-Lite with the checksum coverage set to only the header portion of a packet. This requires a pseudo header checksum calculation only on the encapsulating packet header. The computed checksum value may be cached (before adding the Length field) for each flow/destination and subsequently combined with the Length of each packet to minimise per-packet processing. This value is combined with the UDP payload length for the pseudo header, however this length is expected to be known when performing packet forwarding.
- o The proposed UDP Tunnel Transport [UDPTT] suggested a method where UDP would be modified to derive the checksum only from the encapsulating packet protocol header. This value does not change between packets in a single flow. The value may be cached per flow/destination to minimise per-packet processing.
- o There has been a proposal to simply ignore the UDP checksum value on reception at the tunnel egress, allowing a tunnel ingress to insert any value correct or false. For tunnel usage, a non standard checksum value may be used, forcing an RFC 2460 receiver to drop the packet. The main downside is that it would be impossible to identify a UDP datagram (in the network or an endpoint) that is treated in this way compared to a packet that has actually been corrupted.
- o A method has been proposed that uses a new (to be defined) IPv6 Destination Options Header to provide an end-to-end validation check at the network layer. This would allow an endpoint to verify delivery to an appropriate end point, but would also require IPv6 nodes to correctly handle the additional header, and would require changes to middlebox behavior (e.g. when used with a NAT that always adjusts the checksum value).
- o UDP modified to disable checksum processing [I-D.ietf-6man-udpchecksums]. This eliminates the need for a checksum calculation, but would require constraints on appropriate usage and updates to end-points and middleboxes.
- o IP-in-IP tunneling. As this method completely dispenses with a transport protocol in the outer-layer it has reduced overhead and complexity, but also reduced functionality. There is no outer checksum over the packet and also no ports to perform demultiplexing between different tunnel types. This reduces the information available upon which a load balancer may act.

These options are compared and discussed further in the following sections.

## A.2. Comparison

This section compares the above listed methods to support datagram tunneling. It includes proposals for updating the behaviour of UDP.

While this comparison focuses on applications that are expected to execute on routers, the distinction between a router and a host is not always clear, especially at the transport level. Systems (such as unix-based operating systems) routinely provide both functions. There is no way to identify the role of the receiving node from a received packet.

### A.2.1. Middlebox Traversal

Regular UDP with a standard checksum or the delta encoded optimization for creating correct checksums have the best possibilities for successful traversal of a middlebox. No new support is required.

A method that ignores the UDP checksum on reception is expected to have a good probability of traversal, because most middleboxes perform an incremental checksum update. UDPTT would also have been able to traverse a middlebox with this behaviour. However, a middlebox on the path that attempts to verify a standard checksum will not forward packets using either of these methods, preventing traversal. A method that ignores the checksum has an additional downside in that it prevents improvement of middlebox traversal, because there is no way to identify UDP datagrams that use the modified checksum behaviour.

IP-in-IP or GRE tunnels offer good traversal of middleboxes that have not been designed for security, e.g. firewalls. However, firewalls may be expected to be configured to block general tunnels as they present a large attack surface.

A new IPv6 Destination Options header will suffer traversal issues with middleboxes, especially Firewalls and NATs, and will likely require them to be updated before the extension header is passed.

Datagrams with a zero UDP checksum will not be passed by any middlebox that validates the checksum using RFC 2460 or updates the checksum field, such as NAT or firewalls. This would require an update to correctly handle a datagram with a zero UDP checksum.

UDP-Lite will require an update of almost all type of middleboxes, because it requires support for a separate network-layer protocol number. Once enabled, the method to support incremental checksum update would be identical to that for UDP, but different for checksum

validation.

#### A.2.2. Load Balancing

The usefulness of solutions for load balancers depends on the difference in entropy in the headers for different flows that can be included in a hash function. All the proposals that use the UDP protocol number have equal behavior. UDP-Lite has the potential for equally good behavior as for UDP. However, UDP-Lite is currently unlikely to be supported by deployed hashing mechanisms, which could cause a load balancer to not use the transport header in the computed hash. A load balancer that only uses the IP header will have low entropy, but could be improved by including the IPv6 the flow label, providing that the tunnel ingress ensures that different flow labels are assigned to different flows. However, a transition to the common use of good quality flow labels is likely to take time to deploy.

#### A.2.3. Ingress and Egress Performance Implications

IP-in-IP tunnels are often considered efficient, because they introduce very little processing and low data overhead. The other proposals introduce a UDP-like header incurring associated data overhead. Processing is minimised for the method that uses a zero UDP checksum, ignoring the UDP checksum on reception, and only slightly higher for UDPTT, the extension header and UDP-Lite. The delta-calculation scheme operates on a few more fields, but also introduces serious failure modes that can result in a need to calculate a checksum over the complete datagram. Regular UDP is clearly the most costly to process, always requiring checksum calculation over the entire datagram.

It is important to note that the zero UDP checksum method, ignoring checksum on reception, the Option Header, UDPTT and UDP-Lite will likely incur additional complexities in the application to incorporate a negotiation and validation mechanism.

#### A.2.4. Deployability

The major factors influencing deployability of these solutions are a need to update both end-points, a need for negotiation and the need to update middleboxes. These are summarised below:

- o The solution with the best deployability is regular UDP. This requires no changes and has good middlebox traversal characteristics.
- o The next easiest to deploy is the delta checksum solution. This does not modify the protocol on the wire and only needs changes in

tunnel ingress.

- o IP-in-IP tunnels should not require changes to the end-points, but raise issues when traversing firewalls and other security devices, which are expected to require updates.
- o Ignoring the checksum on reception will require changes at both end-points. The never ceasing risk of path failure requires additional checks to ensure this solution is robust and will require changes or additions to the tunnel control protocol to negotiate support and validate the path.
- o The remaining solutions (including the zero checksum method) offer similar deployability. UDP-Lite requires support at both end-points and in middleboxes. UDPTT and the zero UDP checksum method with or without an extension header require support at both end-points and in middleboxes. UDP-Lite, UDPTT, and the zero UDP checksum method and use of extension headers may additionally require changes or additions to the tunnel control protocol to negotiate support and path validation.

#### A.2.5. Corruption Detection Strength

The standard UDP checksum and the delta checksum can both provide some verification at the tunnel egress. This can significantly reduce the probability that a corrupted inner packet is forwarded. UDP-Lite, UDPTT and the extension header all provide some verification against corruption, but do not verify the inner packet. They only provide a strong indication that the delivered packet was intended for the tunnel egress and was correctly delimited.

The methods using a zero UDP checksum, ignoring the UDP checksum on reception and IP-and-IP encapsulation all provide no verification that a received datagram was intended to be processed by a specific tunnel egress or that the inner encapsulated packet was correct. Section 3.1 discusses experience using specific protocols in well-managed networks.

#### A.2.6. Comparison Summary

The comparisons above may be summarised as "there is no silver bullet that will slay all the issues". One has to select which down side(s) can best be lived with. Focusing on the existing solutions, this can be summarized as:



Regular UDP:    The method defined in RFC 2460 has good middlebox traversal and load balancing and multiplexing, requiring a checksum in the outer headers covering the whole packet.

IP in IP:    A low complexity encapsulation, with limited middlebox traversal, no multiplexing support, and currently poor load balancing support that could improve over time.

UDP-Lite:    A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, currently poor load balancing support that could improve over time, in most cases requiring application level negotiation to select the protocol and validation to confirm the path forwards UDP-Lite.

The delta-checksum is an optimization in the processing of UDP, as such it exhibits some of the drawbacks of using regular UDP.

The remaining proposals may be described in similar terms:

Zero-Checksum:    A low complexity encapsulation, with good multiplexing support, limited middlebox traversal that could improve over time, good load balancing support, in most cases requiring application level negotiation and validation to confirm the path forwards a zero UDP checksum.

UDPTT:    A medium complexity encapsulation, with good multiplexing support, limited middlebox traversal, but possible to improve over time, good load balancing support, in most cases requiring application level negotiation to select the transport and validation to confirm the path forwards UDPTT datagrams.

IPv6 Destination Option IP in IP tunneling:    A medium complexity, with no multiplexing support, limited middlebox traversal, currently poor load balancing support that could improve over time, in most cases requiring negotiation to confirm the option is supported and validation to confirm the path forwards the option.

IPv6 Destination Option combined with UDP Zero-checksumming:    A medium complexity encapsulation, with good multiplexing support, limited load balancing support that could improve over time, in most cases requiring negotiation to confirm the option is supported and validation to confirm the path forwards the option.

Ignore the checksum on reception:    A low complexity encapsulation, with good multiplexing support, medium middlebox traversal that never can improve, good load balancing support, in most cases requiring negotiation to confirm the option is supported by the

remote endpoint and validation to confirm the path forwards a zero UDP checksum.

There is no clear single optimum solution. If the most important need is to traverse middleboxes, then the best choice is to stay with regular UDP and consider the optimizations that may be required to perform the checksumming. If one can live with limited middlebox traversal, low complexity is necessary and one does not require load balancing, then IP-in-IP tunneling is the simplest. If one wants strengthened error detection, but with currently limited middlebox traversal and load-balancing. UDP-Lite is appropriate. Zero UDP checksum addresses another set of constraints, low complexity and a need for load balancing from the current Internet, providing it can live with currently limited middlebox traversal.

Techniques for load balancing and middlebox traversal do continue to evolve. Over a long time, developments in load balancing have good potential to improve. This time horizon is long since it requires both load balancer and end-point updates to get full benefit. The challenges of middlebox traversal are also expected to change with time, as device capabilities evolve. Middleboxes are very prolific with a larger proportion of end-user ownership, and therefore may be expected to take long time cycles to evolve.

One potential advantage is that the deployment of IPv6-capable middleboxes are still in its initial phase and the quicker a new method becomes standardized, the fewer boxes will be non-compliant.

Thus, the question of whether to permit use of datagrams with a zero UDP checksum for IPv6 under reasonable constraints, is therefore best viewed as a trade-off between a number of more subjective questions:

- o Is there sufficient interest in using a zero UDP checksum with the given constraints (summarised below)?
- o Are there other avenues of change that will resolve the issue in a better way and sufficiently quickly ?
- o Do we accept the complexity cost of having one more solution in the future?

The analysis concludes that the IETF should carefully consider constraints on sanctioning the use of any new transport mode. The 6man working group of the IETF has determined that the answer to the above questions are sufficient to update IPv6 to standardise use of a zero UDP checksum for use by tunnel encapsulations for specific applications.

Each application should consider the implications of choosing an IPv6 transport that uses a zero UDP checksum. In many cases, standard methods may be more appropriate, and may simplify application design. The use of checksum off-loading may help alleviate the checksum processing cost and permit use of a checksum using method defined in RFC 2460.

## Appendix B. Document Change History

{RFC EDITOR NOTE: This section must be deleted prior to publication}

Individual Draft 00    This is the first DRAFT of this document - It contains a compilation of various discussions and contributions from a variety of IETF WGs, including: mboned, tsv, 6man, lisp, and behave. This includes contributions from Magnus with text on RTP, and various updates.

### Individual Draft 01

- \* This version corrects some typos and editorial NiTs and adds discussion of the need to negotiate and verify operation of a new mechanism (3.3.4).

### Individual Draft 02

- \* Version -02 corrects some typos and editorial NiTs.
- \* Added reference to ECMP for tunnels.
- \* Clarifies the recommendations at the end of the document.

### Working Group Draft 00

- \* Working Group Version -00 corrects some typos and removes much of rationale for UDPTT. It also adds some discussion of IPv6 extension header.

### Working Group Draft 01

- \* Working Group Version -01 updates the rules and incorporates off-list feedback. This version is intended for wider review within the 6man working group.

Working Group Draft 02

- \* This version is the result of a major rewrite and re-ordering of the document.
- \* A new section comparing the results have been added.
- \* The constraints list has been significantly altered by removing some and rewording other constraints.
- \* This contains other significant language updates to clarify the intent of this draft.

Working Group Draft 03

- \* Editorial updates

Working Group Draft 04

- \* Resubmission only updating the AMT and RFC2765 references.

Working Group Draft 05

- \* Resubmission to correct editorial NiTs - thanks to Bill Atwood for noting these. Group Draft 05.

Working Group Draft 06

- \* Resubmission to keep draft alive (spelling updated from 05).

Working Group Draft 07

- \* Interim Version
- \* Submission after IESG Feedback Added
- \* Updates to enable the document to become a PS Applicability Statement

Working Group Draft 08

- \* First Version written as a PS Applicability Statement
- \* Changes to reflect decision to update RFC 2460, rather than recommend decision
- \* Updates to requirements for middleboxes

- \* Inclusion of requirements for security, API, and tunnel
- \* Move of the rationale for the update to an Annex (former section 4)

Working Group Draft 09

- \* Submission after second WGLC (note mistake corrected in -09).
- \* Clarified role of API for supporting full checksum.
- \* Clarified that full checksum is required in security considerations, and therefore noting that full checksum should not be treated as an attack - consistent with remainder of document.
- \* Added mention that API can set a mode in transport stack - to link to similar statement in RFC 2460 update.
- \* Fixed typos.

Working Group Draft 10

- \* Submission to correct unwanted removal of text from section 5 bullets 5-7 by GF.
- \* Replaced section 5 text with the text from 08, and reapplied the editorial correction.
- \* Note to reviewers: Please compare this revision with -08 used in the IETF LC).

Working Group Draft 11

- \* Added REF for 5097 (Noted by S.Turner)
- \* Added text in response to P. Resnick on place where checksum is calculated.
- \* Added text to note experience with MPLS/PWE; Appendix updated to refer to this (S. Bryant)
- \* Added text in response to P.Resnick's 2nd comments.
- \* Request to make UDP-Lite more clearly recommended (J Touch, P.Resnick)

- \* Added considerations around usage of zero checksum in routers.
- \* Added text in response to Stewart Bryant's comments on router requirements.

#### Authors' Addresses

Godred Fairhurst  
University of Aberdeen  
School of Engineering  
Aberdeen, AB24 3UE  
Scotland, UK

Email: [gorry@erg.abdn.ac.uk](mailto:gorry@erg.abdn.ac.uk)  
URI: <http://www.erg.abdn.ac.uk/users/gorry>

Magnus Westerlund  
Ericsson  
Farogatan 6  
Stockholm, SE-164 80  
Sweden

Phone: +46 8 719 0000  
Email: [magnus.westerlund@ericsson.com](mailto:magnus.westerlund@ericsson.com)



6MAN  
Internet-Draft  
Updates: 3986 (if approved)  
Intended status: Standards Track  
Expires: March 25, 2013

B. Carpenter  
Univ. of Auckland  
S. Cheshire  
Apple Inc.  
R. Hinden  
Check Point  
September 21, 2012

Representing IPv6 Zone Identifiers in Address Literals and Uniform  
Resource Identifiers  
draft-ietf-6man-uri-zoneid-04

Abstract

This document describes how the Zone Identifier of an IPv6 scoped address can be represented in a literal IPv6 address and in a Uniform Resource Identifier that includes such a literal address. It updates RFC 3986 accordingly.

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 March 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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



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

## Table of Contents

1. Introduction . . . . .	3
2. Specification . . . . .	4
3. Web Browsers . . . . .	5
4. Security Considerations . . . . .	5
5. IANA Considerations . . . . .	6
6. Acknowledgements . . . . .	6
7. Change log [RFC Editor: Please remove] . . . . .	6
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	7
Appendix A. Alternatives Considered . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

The Uniform Resource Identifier (URI) syntax [RFC3986] defined how a literal IPv6 address can be represented in the "host" part of a URI. A subsequent specification [RFC4007] extended the text representation of limited-scope IPv6 addresses such that a zone identifier may be concatenated to a literal address, for purposes described in that RFC. Zone identifiers are especially useful in contexts where literal addresses are typically used, for example during fault diagnosis, when it may be essential to specify which interface is used for sending to a link local address. It should be noted that zone identifiers have purely local meaning within the host where they are defined, and they are completely meaningless for any other host. Today, they are only meaningful when attached to addresses with less than global scope, but it is possible that other uses might be defined in the future.

RFC 4007 does not specify how zone identifiers are to be represented in URIs. Practical experience has shown that this feature is useful, in particular when using a web browser for debugging with link local addresses, but as it is undefined, it is not implemented consistently in URI parsers or in browsers.

Some versions of some browsers accept the RFC 4007 syntax for scoped IPv6 addresses embedded in URIs, i.e., they have been coded to interpret the "%" sign according to RFC 4007 instead of RFC 3986. Clearly this approach is very convenient for users, although it formally breaches the syntax rules of RFC 3986. The present document defines an alternative approach that respects and extends the rules of URI syntax, and IPv6 literals in general, to be consistent.

Thus, this document updates [RFC3986] by adding syntax to allow a zone identifier to be included in a literal IPv6 address within a URI.

It should be noted that in other contexts than a user interface, a zone identifier is mapped into a numeric zone index or interface number. The MIB textual convention [RFC4001] and the socket interface [RFC3493] define this as a 32 bit unsigned integer. The mapping between the human-readable zone identifier string and the numeric value is a host-specific function that varies between operating systems. The present document is concerned only with the human-readable string.

Several alternative solutions were considered while this document was developed. The Appendix briefly describes the alternatives and their advantages and disadvantages.

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

## 2. Specification

According to RFC 4007, a zone identifier is attached to the textual representation of an IPv6 address by concatenating "%" followed by <zone\_id>, where <zone\_id> is a string identifying the zone of the address. However, RFC 4007 gives no precise definition of the character set allowed in <zone\_id>. There are no rules or de facto standards for this. For example, the first Ethernet interface in a host might be called %0, %1, %en1, %eth0, or whatever the implementer happened to choose.

In a URI, a literal IPv6 address is always embedded between "[" and "]". This document specifies how a <zone\_id> can be appended to the address. A <zone\_id> SHOULD contain only ASCII characters classified in RFC 3986 as "unreserved", which conveniently excludes "]" in order to simplify parsing.

Unfortunately "%" is always treated as an escape character in a URI, and according to RFC 3986 it MUST therefore itself be escaped in a URI, in the form "%25". Thus, the scoped address fe80::a%en1 would appear in a URI as http://[fe80::a%25en1].

If an operating system uses any other characters in zone or interface identifiers that are not in the "unreserved" character set, they MUST be escaped with a "%" sign according to RFC 3986.

We now present the necessary formal syntax.

In RFC 3986, the IPv6 literal format is formally defined in ABNF [RFC5234] by the following rule:

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
```

To provide support for a zone identifier, the existing syntax of IPv6address is retained, and a zone identifier may be added optionally to any literal address. This allows flexibility for unknown future uses. The rule quoted above from RFC 3986 is replaced by three rules:

```
IP-literal = "[" ( IPv6addrz / IPvFuture  ) "]"
```

```
ZoneID = 1*( unreserved / pct-encoded )
```

IPv6addrz = IPv6address [ "%" ZoneID ]

The rules in [RFC5952] SHOULD be applied in producing URIs.

RFC 3986 states that URIs have a global scope, but that in some cases their interpretation depends on the end-user's context. URIs including a ZoneID are to be interpreted only in the context of the host where they originate, since the ZoneID is of local significance only.

The 6man WG discussed and rejected an alternative in which the existing syntax of IPv6address would be extended by an option to add the ZoneID only for the case of link-local addresses. It was felt that the present solution offers more flexibility for future uses and is more straightforward to implement.

RFC 4007 offers guidance on how the ZoneID affects interface/address selection inside the IPv6 stack. Note that the behaviour of an IPv6 stack if passed a non-zero zone index for an address other than link-local is undefined.

### 3. Web Browsers

Due to the lack of a standard in this area, web browsers have been inconsistent in providing for ZoneIDs. Many have no support, but there are examples of ad hoc support. For example, older versions of Firefox allowed the use of a ZoneID preceded by an unescaped "%" character, but this was removed for consistency with RFC 3986. As another example, recent versions of Internet Explorer allow use of a ZoneID preceded by a "%" character escaped as "%25", still beyond the syntax allowed by RFC 3986. This syntax extension is in fact used internally in the Windows operating system and some of its APIs.

This document implies that all browsers should recognise a ZoneID preceded by an escaped "%". In the spirit of "be liberal with what you accept", we also recommend that URI parsers accept bare "%" signs (i.e., a "%" not followed by two valid hexadecimal characters). This makes it easy for a user to copy and paste a string such as "fe80::a%en1" from the output of a "ping" command and have it work.

### 4. Security Considerations

The security considerations of [RFC3986] and [RFC4007] apply. In particular, this URI format creates a specific pathway by which a deceitful zone index might be communicated, as mentioned in the final security consideration of RFC 4007. It is emphasised that the format

is intended only for debugging purposes, but of course this intention does not prevent misuse.

To limit this risk, implementations SHOULD NOT allow use of this format except for well-defined usages such as sending to link local addresses under prefix fe80::/10.

An HTTP server or proxy MUST ignore any ZoneID attached to an incoming URI, as it only has local significance at the sending host.

## 5. IANA Considerations

This document requests no action by IANA.

## 6. Acknowledgements

The lack of this format was first pointed out by Margaret Wasserman some years ago, and more recently by Kerry Lynn. A previous draft document by Martin Duerst and Bill Fenner [I-D.fenner-literal-zone] discussed this topic but was not finalised.

Valuable comments and contributions were made by Karl Auer, Carsten Bormann, Brian Haberman, Tatuya Jinmei, Tom Petch, Tomoyuki Sahara, Juergen Schoenwaelder, Dave Thaler, and Ole Troan.

Brian Carpenter was a visitor at the Computer Laboratory, Cambridge University during part of this work.

This document was produced using the xml2rfc tool [RFC2629].

## 7. Change log [RFC Editor: Please remove]

draft-ietf-6man-uri-zoneid-03: additional author, 2012-21-10.

draft-ietf-6man-uri-zoneid-03: reverted to percent-encoded model following WGLC, 2012-09-10.

draft-ietf-6man-uri-zoneid-02: additional WG comments, 2012-07-11.

draft-ietf-6man-uri-zoneid-01: use "-" instead of %25, listed alternatives in Appendix, according to WG debate, added suggestion for browser developers, 2012-05-29.

draft-ietf-6man-uri-zoneid-00: adopted by WG, fixed syntax to allow for % encoded characters, 2012-02-17.

draft-carpenter-6man-uri-zoneid-01: chose Option 2, removed 15 character limit, added explanation of ID/number mapping and other clarifications, 2012-02-08.

draft-carpenter-6man-uri-zoneid-00: original version, 2011-12-07.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

### 8.2. Informative References

- [I-D.fenner-literal-zone]  
Fenner, B. and M. Duerst, "Formats for IPv6 Scope Zone Identifiers in Literal Address Formats",  
draft-fenner-literal-zone-02 (work in progress),  
October 2005.
- [I-D.iab-identifier-comparison]  
Thaler, D., "Issues in Identifier Comparison for Security Purposes", draft-iab-identifier-comparison-03 (work in progress), July 2012.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [chrome] Google, "Use the address bar (omnibox)", 2012, <<http://support.google.com/chrome/bin/answer.py?answer=95440>>.

## Appendix A. Alternatives Considered

### 1. Leave the problem unsolved.

This would mean that per-interface diagnostics would still have to be performed using ping or ping6:

```
ping fe80::a%en1
```

Advantage: works today.

Disadvantage: less convenient than using a browser.

### 2. Simply using the percent character.

```
http://[fe80::a%en1]
```

Advantage: allows use of browser, allows cut and paste.

Disadvantage: invalid syntax under RFC 3986; not acceptable to URI community.

### 3. Escaping the escape character as allowed by RFC 3986:

```
http://[fe80::a%25en1]
```

Advantage: allows use of browser, consistent with general URI syntax.

Disadvantage: somewhat ugly and confusing, doesn't allow simple cut and paste.

### 4. Alternative separator

```
http://[fe80::a-en1]
```

Advantage: allows use of browser, simple syntax

Disadvantage: Requires all IPv6 address literal parsers and generators to be updated in order to allow simple cut and paste;

inconsistent with existing tools and practice.

Note: the initial proposal for this choice was to use an underscore as the separator, but it was noted that this becomes effectively invisible when a user interface automatically underlines URLs.

5. With the "IPvFuture" syntax left open in RFC 3986:

`http://[v6.fe80::a_en1]`

Advantage: allows use of browser.

Disadvantage: ugly and redundant, doesn't allow simple cut and paste.

#### Authors' Addresses

Brian Carpenter  
Department of Computer Science  
University of Auckland  
PB 92019  
Auckland, 1142  
New Zealand  
  
Email: brian.e.carpenter@gmail.com

Stuart Cheshire  
Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
US  
  
Email: cheshire@apple.com

Robert M. Hinden  
Check Point Software Technologies, Inc.  
800 Bridge Parkway  
Redwood City, CA 94065  
US  
  
Email: bob.hinden@gmail.com





6MAN  
Internet-Draft  
Updates: 3986 (if approved)  
Intended status: Standards Track  
Expires: June 10, 2013

B. Carpenter  
Univ. of Auckland  
S. Cheshire  
Apple Inc.  
R. Hinden  
Check Point  
December 7, 2012

Representing IPv6 Zone Identifiers in Address Literals and Uniform  
Resource Identifiers  
draft-ietf-6man-uri-zoneid-06

Abstract

This document describes how the Zone Identifier of an IPv6 scoped address, as defined in RFC 4007, can be represented in a literal IPv6 address and in a Uniform Resource Identifier that includes such a literal address. It updates RFC 3986 accordingly.

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 June 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Specification . . . . .	4
3. Web Browsers . . . . .	5
4. Security Considerations . . . . .	6
5. IANA Considerations . . . . .	6
6. Acknowledgements . . . . .	6
7. Change log [RFC Editor: Please remove] . . . . .	7
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	8
Appendix A. Options Considered . . . . .	8
Authors' Addresses . . . . .	10

## 1. Introduction

The Uniform Resource Identifier (URI) syntax [RFC3986] defined how a literal IPv6 address can be represented in the "host" part of a URI. A subsequent specification [RFC4007] extended the text representation of limited-scope IPv6 addresses such that a zone identifier may be concatenated to a literal address, for purposes described in that RFC. Zone identifiers are especially useful in contexts where literal addresses are typically used, for example during fault diagnosis, when it may be essential to specify which interface is used for sending to a link local address. It should be noted that zone identifiers have purely local meaning within the node where they are defined, often being the same as IPv6 interface names. They are completely meaningless for any other node. Today, they are only meaningful when attached to addresses with less than global scope, but it is possible that other uses might be defined in the future.

RFC 4007 does not specify how zone identifiers are to be represented in URIs. Practical experience has shown that this feature is useful, in particular when using a web browser for debugging with link local addresses, but as it is undefined, it is not implemented consistently in URI parsers or in browsers.

Some versions of some browsers accept the RFC 4007 syntax for scoped IPv6 addresses embedded in URIs, i.e., they have been coded to interpret the "%" sign according to RFC 4007 instead of RFC 3986. Clearly this approach is very convenient for users, although it formally breaches the syntax rules of RFC 3986. The present document defines an alternative approach that respects and extends the rules of URI syntax, and IPv6 literals in general, to be consistent.

Thus, this document updates [RFC3986] by adding syntax to allow a zone identifier to be included in a literal IPv6 address within a URI.

It should be noted that in other contexts than a user interface, a zone identifier is mapped into a numeric zone index or interface number. The MIB textual convention InetZoneIndex [RFC4001] and the socket interface [RFC3493] define this as a 32 bit unsigned integer. The mapping between the human-readable zone identifier string and the numeric value is a host-specific function that varies between operating systems. The present document is concerned only with the human-readable string.

Several alternative solutions were considered while this document was developed. The Appendix briefly describes the various options and their advantages and disadvantages.

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

## 2. Specification

According to RFC 4007, a zone identifier is attached to the textual representation of an IPv6 address by concatenating "%" followed by <zone\_id>, where <zone\_id> is a string identifying the zone of the address. However, RFC 4007 gives no precise definition of the character set allowed in <zone\_id>. There are no rules or de facto standards for this. For example, the first Ethernet interface in a host might be called %0, %1, %en1, %eth0, or whatever the implementer happened to choose.

In a URI, a literal IPv6 address is always embedded between "[" and "]". This document specifies how a <zone\_id> can be appended to the address. Unfortunately "%" is always treated as an escape character in a URI, and according to RFC 3986 it MUST therefore itself be percent-encoded in a URI, in the form "%25". Thus, the scoped address fe80::a%en1 would appear in a URI as http://[fe80::a%25en1].

A <zone\_id> SHOULD contain only ASCII characters classified in RFC 3986 as "unreserved". This excludes characters such as "]" or even "%" which would complicate parsing. However, the syntax below does allow such characters to be percent-encoded, for compatibility with existing devices that use them.

If an operating system uses any other characters in zone or interface identifiers that are not in the "unreserved" character set, they MUST be escaped with a "%" sign according to RFC 3986.

We now present the necessary formal syntax.

In RFC 3986, the IPv6 literal format is formally defined in ABNF [RFC5234] by the following rule:

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
```

To provide support for a zone identifier, the existing syntax of IPv6address is retained, and a zone identifier may be added optionally to any literal address. This allows flexibility for unknown future uses. The rule quoted above from RFC 3986 is replaced by three rules:

```
IP-literal = "[" ( IPv6address / IPv6addrz / IPvFuture  ) "]"
```

```
ZoneID = 1*( unreserved / pct-encoded )
```

```
IPv6addrz = IPv6address "%25" ZoneID
```

This syntax fills the gap that is described at the end of Section 11.7 of RFC 4007.

The rules in [RFC5952] SHOULD be applied in producing URIs.

RFC 3986 states that URIs have a global scope, but that in some cases their interpretation depends on the end-user's context. URIs including a ZoneID are to be interpreted only in the context of the host where they originate, since the ZoneID is of local significance only.

RFC 4007 offers guidance on how the ZoneID affects interface/address selection inside the IPv6 stack. Note that the behaviour of an IPv6 stack if passed a non-null zone index for an address other than link-local is undefined.

### 3. Web Browsers

This section discusses how web browsers might handle this syntax extension. Unfortunately there is no formal distinction between the syntax allowed in a browser's input dialogue box and the syntax allowed in URIs. For this reason, no normative statements are made in this section.

Due to the lack of defined syntax, web browsers have been inconsistent in providing for ZoneIDs. Many have no support, but there are examples of ad hoc support. For example, some versions of Firefox allowed the use of a ZoneID preceded by an unescaped "%" character, but this was removed for consistency with RFC 3986. As another example, some versions of Internet Explorer allow use of a ZoneID preceded by a "%" character escaped as "%25", still beyond the syntax allowed by RFC 3986. This syntax extension is in fact used internally in the Windows operating system and some of its APIs.

It is desirable for all browsers to recognise a ZoneID preceded by an escaped "%". In the spirit of "be liberal with what you accept", we also suggest that URI parsers accept bare "%" signs when possible (i.e., a "%" not followed by two valid and meaningful hexadecimal characters). This would make it possible for a user to copy and paste a string such as "fe80::a%e1" from the output of a "ping" command and have it work. On the other hand, "%e1" would need to be

manually escaped as "fe80::a%25ee1" to avoid any risk of misinterpretation.

Such bare "%" signs are for user interface convenience, and need to be turned into properly escaped characters (where "%25" encodes "%") before the URI is used in any protocol or HTML document. However, URIs including a ZoneID have no meaning outside the originating node. It would therefore be highly desirable for a browser to remove the ZoneID from a URI before including that URI in an HTTP request.

The normal diagnostic usage for the ZoneID syntax will cause it to be entered in the browser's input dialogue box. Thus, URIs including a ZoneID are unlikely to be encountered in HTML documents. However, if they do (for example, in a diagnostic script coded in HTML) it would be appropriate to treat them exactly as above.

#### 4. Security Considerations

The security considerations of [RFC3986] and [RFC4007] apply. In particular, this URI format creates a specific pathway by which a deceitful zone index might be communicated, as mentioned in the final security consideration of RFC 4007. It is emphasised that the format is intended only for debugging purposes, but of course this intention does not prevent misuse.

To limit this risk, implementations MUST NOT allow use of this format except for well-defined usages such as sending to link local addresses under prefix fe80::/10. At the time of writing, this is the only well-defined usage known.

An HTTP client, proxy or other intermediary MUST remove any ZoneID attached to an outgoing URI, as it only has local significance at the sending host.

#### 5. IANA Considerations

This document requests no action by IANA.

#### 6. Acknowledgements

The lack of this format was first pointed out by Margaret Wasserman some years ago, and more recently by Kerry Lynn. A previous draft document by Martin Duerst and Bill Fenner [I-D.fenner-literal-zone] discussed this topic but was not finalised.

Valuable comments and contributions were made by Karl Auer, Carsten Bormann, Benoit Claise, Stephen Farrell, Brian Haberman, Ted Hardie, Tatuya Jinmei, Yves Lafon, Barry Leiba, Radia Perlman, Tom Petch, Tomoyuki Sahara, Juergen Schoenwaelder, Dave Thaler, Martin Thomson, and Ole Troan.

Brian Carpenter was a visitor at the Computer Laboratory, Cambridge University during part of this work.

This document was produced using the xml2rfc tool [RFC2629].

## 7. Change log [RFC Editor: Please remove]

draft-ietf-6man-uri-zoneid-06: responding to IETF Last Call and IESG comments, 2012-12-07.

draft-ietf-6man-uri-zoneid-05: tuned ABNF, clarified RFC 4007 text, 2012-11-06.

draft-ietf-6man-uri-zoneid-04: additional author, 2012-09-21.

draft-ietf-6man-uri-zoneid-03: reverted to percent-encoded model following WGLC, 2012-09-10.

draft-ietf-6man-uri-zoneid-02: additional WG comments, 2012-07-11.

draft-ietf-6man-uri-zoneid-01: use "-" instead of %25, listed alternatives in Appendix, according to WG debate, added suggestion for browser developers, 2012-05-29.

draft-ietf-6man-uri-zoneid-00: adopted by WG, fixed syntax to allow for % encoded characters, 2012-02-17.

draft-carpenter-6man-uri-zoneid-01: chose Option 2, removed 15 character limit, added explanation of ID/number mapping and other clarifications, 2012-02-08.

draft-carpenter-6man-uri-zoneid-00: original version, 2011-12-07.

## 8. References

### 8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.



- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

## 8.2. Informative References

- [I-D.fenner-literal-zone]  
Fenner, B. and M. Duerst, "Formats for IPv6 Scope Zone Identifiers in Literal Address Formats", draft-fenner-literal-zone-02 (work in progress), October 2005.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [chrome] Google, "Use the address bar (omnibox)", 2012, <<http://support.google.com/chrome/bin/answer.py?answer=95440>>.

## Appendix A. Options Considered

The syntax defined above allows a ZoneID to be added to any IPv6 address. The 6man WG discussed and rejected an alternative in which the existing syntax of IPv6address would be extended by an option to add the ZoneID only for the case of link-local addresses. It was felt that the present solution offers more flexibility for future uses and is more straightforward to implement.

The various syntax options considered are now briefly described.

1. Leave the problem unsolved.

This would mean that per-interface diagnostics would still have to be performed using ping or ping6:

```
ping fe80::a%en1
```

Advantage: works today.

Disadvantage: less convenient than using a browser.

2. Simply using the percent character.

```
http://[fe80::a%en1]
```

Advantage: allows use of browser, allows cut and paste.

Disadvantage: invalid syntax under RFC 3986; not acceptable to URI community.

3. Escaping the escape character as allowed by RFC 3986:

```
http://[fe80::a%25en1]
```

Advantage: allows use of browser, consistent with general URI syntax.

Disadvantage: somewhat ugly and confusing, doesn't allow simple cut and paste.

This is the option chosen for standardization.

4. Alternative separator

```
http://[fe80::a-en1]
```

Advantage: allows use of browser, simple syntax

Disadvantage: Requires all IPv6 address literal parsers and generators to be updated in order to allow simple cut and paste; inconsistent with existing tools and practice.

Note: the initial proposal for this choice was to use an underscore as the separator, but it was noted that this becomes effectively invisible when a user interface automatically underlines URLs.

5. With the "IPvFuture" syntax left open in RFC 3986:

`http://[v6.fe80::a_en1]`

Advantage: allows use of browser.

Disadvantage: ugly and redundant, doesn't allow simple cut and paste.

#### Authors' Addresses

Brian Carpenter  
Department of Computer Science  
University of Auckland  
PB 92019  
Auckland, 1142  
New Zealand

Email: [brian.e.carpenter@gmail.com](mailto:brian.e.carpenter@gmail.com)

Stuart Cheshire  
Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
US

Email: [cheshire@apple.com](mailto:cheshire@apple.com)

Robert M. Hinden  
Check Point Software Technologies, Inc.  
800 Bridge Parkway  
Redwood City, CA 94065  
US

Email: [bob.hinden@gmail.com](mailto:bob.hinden@gmail.com)



Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: April 18, 2013

A. Kaiser  
S. Decremps  
A. Petrescu  
CEA  
October 15, 2012

Prefix Delegation extension to Neighbor Discovery protocol  
draft-kaiser-nd-pd-00

Abstract

This document describes an extension to the Neighbor Discovery protocol (ND). The proposed extension enhances ND by providing it a new option: the Prefix Delegation option (ND\_PD). ND\_PD offers the possibility for a router to ask for prefixes to be delegated, even if no DHCPv6 Server (or Relay) are present on link.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Requirements . . . . .	5
4. Related works . . . . .	6
5. Protocol overview . . . . .	7
5.1. Example use case . . . . .	7
6. Format of the Prefix Delegation option . . . . .	9
6.1. Format of the Prefix Collection structure . . . . .	10
6.2. Format of the Prefix Information structure . . . . .	11
7. Prefix Delegation packets format . . . . .	13
7.1. Requesting prefixes . . . . .	13
7.2. Delegating prefixes . . . . .	13
7.3. Renewing/rebinding prefixes . . . . .	14
7.4. Releasing prefixes . . . . .	15
8. Advertisement of the ND_PD service . . . . .	16
9. Local operations on requesting and delegating routers . . . . .	17
9.1. Delegating router behaviour . . . . .	17
9.2. Requesting router behaviour . . . . .	18
10. Status codes . . . . .	19
11. Security Considerations . . . . .	20
12. IANA Considerations . . . . .	21
13. Acknowledgements . . . . .	22
14. References . . . . .	23
Authors' Addresses . . . . .	24

## 1. Introduction

This document describes a new option that extends ND with a PD mechanism. Using this mechanism, a requesting router can ask for a global IPv6 prefix to a delegating router even if any DHCPv6 Server (or Relay) is present on the link.

The goal of the ND\_PD mechanism described in this document is the same as the DHCPv6 Prefix Delegation mechanism (DHCPv6\_PD) described in [DHCPV6\_PD]. Therefore, the ND\_PD mechanism can be seen as a substitute of the DHCPv6\_PD mechanism that can be used on links lacking in DHCPv6 Servers (or Relays). Indeed, there exists a various number of situations where the DHCPv6 services may not be enabled on a link. In the context of vehicular networks for instance, a vehicle (called Leaf Vehicle (LV)) may access the Internet through another vehicle (called Internet Vehicle (IV)) that shares its Internet connexion. In order to provide Internet access to the nodes present in the LV, the latter needs a global IPv6 prefix. If the IV does not provide DHCPv6 services, the LV will not be able to get a global IPv6 prefix. In this kind of situations, the LV can still ask for a prefix using the ND\_PD mechanism. Indeed, as the ND protocol is present on each IPv6 capable node (which is not the case for DHCPv6), providing ND with a PD extension ensures that any IPv6 router is able to request for a prefix in any situation. Moreover, as the proposed ND\_PD mechanism relies only on the exchange of two messages (compared with DHCPv6\_PD that needs the exchange of 4 messages), it can be more suitable in the case of highly mobile networks (e.g. vehicular networks).

## 2. Terminology

This document uses the terminology defined in [DHCPV6\_PD], [NEIGHDISC], and [SLAAC]. Also the following additional terms are used:

Requesting router:	A router that is asking for prefixes to be delegated.
Delegating router:	A router that provides prefixes to requesting routers.
Prefix Collection:	A logical structure that stores a list of Prefix Informations. A Prefix Collection is identified by its unique identifier, namely PC_ID.
Prefix Information:	A logical structure that stores all informations related to a prefix. A Prefix Information is always associated with a Prefix Collection.



### 3. Requirements

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [KEYWORDS].

#### 4. Related works

A few drafts about providing a PD mechanism to the ND protocol have already been proposed in the past.

In [DRAFT\_LUTCHANSKY] the author proposes to add a PD option to the Router Solicitation (RS) and Router Advertisement (RA) messages generated by routers. A router that needs a global prefix can ask for one by including a PD option in a RS message. Then, a router that owns prefixes for delegation replies to the request with a RA that includes a PD option. The main advantage of this proposal is that it is very simple and does not require any additional message to work. However it lacks of completeness: the handling as well as the renewing and releasing of the delegated prefixes are not taken into consideration.

In [DRAFT\_HABERMAN] a more complete PD mechanism for ND protocol is proposed. The mechanism is based on two new ICMP messages: the Prefix Request and the Prefix Delegation. The former is used by a requesting router to ask for a prefix. Conversely, the latter is used by a delegating router to reply to the request. The proposal also includes the possibility for a requesting router to renew a delegated prefix that has not expired yet and to return a delegated prefix that is no longer required.

The proposed ND\_PD mechanism that is described in this document is close to the one described in [DRAFT\_HABERMAN]. However, our mechanism relies on the creation of new RS/RA options rather than the creation of new ICMP messages. Also, the ND\_PD service provided by a router is advertised in the header of its RA. This information enable requesting routers to be aware of the presence of routers that provide the ND\_PD service on link without asking for it by, for instance, sending a RS on the all-routers link-local multicast address.

## 5. Protocol overview

The ND\_PD mechanism presented in this document defines a new option for the RS and the RA messages. Using this option, routers on a same link can request and/or delegate prefixes to other routers even if any DHCPv6\_PD Server/Relay is present on the link.

The Prefix Delegation option proposed in this document adds the following functionalities to the Neighbor Discovery protocol:

- o Request of prefixes
- o Delegation of prefixes
- o Renew of already delegated prefixes
- o Release of delegated prefixes

These functionalities are described in more details in Section 6.

### 5.1. Example use case

In Figure 1 a vehicular scenario is shown. The figure depicts two vehicles: a Leaf Vehicle (LV) and an Internet Vehicle (IV). Both of them embed several Local Fixed Nodes (LFN), like sensors for example, and a Mobile Router (MR) that acts as a gateway between the network inside the vehicle (made of LFNs) and the outdoor. The main difference between the LV and the IV is the number of interfaces present in the MR and its capacity to connect to an infrastructure network. In this figure, the MR-IV has two egress interfaces: one connected to the infrastructure (E2) using a long range technology (e.g. 3G or LTE) and the other one connected to the LV (E1) in ad-hoc using a short range technology (e.g. wifi). Let us assume that a DHCPv6 server exists in the infrastructure. Therefore, using DHCPv6\_PD the MR-IV can obtain a global prefix to announce on its ingress interface (I1) to configure its LFNs. In order to enable the LFNs of the LV to access the Internet, the LV also needs a global prefix to assign on its ingress interface I1. As the IV is neither a DHCPv6 Server nor a DHCPv6 Relay, the LV can still ask for a prefix to the IV using the ND\_PD mechanism. If the IV has a pool of prefixes to assign, it will delegate a prefix to the LV. Otherwise, the IV can ask the DHCPv6 Server in the infrastructure for an additional prefix that it will delegate to the LV.

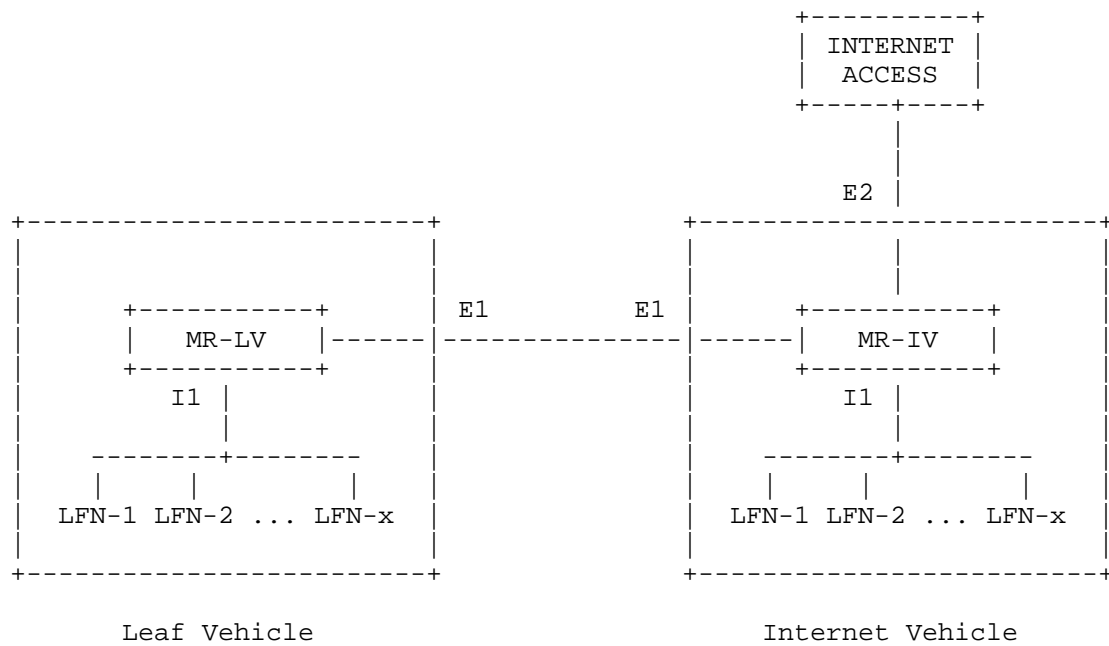


Figure 1: A vehicular scenario

## 6. Format of the Prefix Delegation option

This section details the format of the option used in the ND\_PD mechanism. This option is only valid if included in RS or RA messages and MUST NOT be included in NS, NA, or Redirect messages.

The Prefix Delegation option is used to manage everything related to delegated prefixes. The following operations are considered:

- REQ: Request operation. A requesting router asks for a prefix to a delegating router.
- REP: Reply operation. A delegating router replies to a requesting router. Depending on the type of request, a reply message acts as a delegating message that contains the delegated prefix (i.e. in response of a REQ, REN, or REB message) or as an acknowledgment message that contains the prefix that has been successfully released (i.e. in response of a REL message).
- REN: Renew operation. A requesting router asks the delegating router that provided it the prefix to extend its lifetime.
- REB: Rebind operation. A requesting router asks any delegating router present on the link to extend the lifetime of its delegated prefix.
- REL: Release operation. A requesting router that does not use anymore a delegated prefix informs the delegating router that provided it the prefix its intention of releasing it.

The Prefix Delegation option is composed of a Prefix Delegation header followed by a list of Prefix Collection structures. The format of the Prefix Delegation header is as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |   Length   | Transac. ID |   Msg. Type   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Reserved                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
.                                     List of PC                                     .
.                                     ...                                     .
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type: Value that describes the Prefix Delegation option (TBD: IANA).

Length: Size of the option in blocks of 64 bits (according to [NEIGHDISC]) including the fields "Type" and "Length".

Transac. ID: Identifier of the current messages exchange between the requesting router and the delegating router.

Msg. Type: Describes the type of operation related to the message (REQ, REP, REN, REB, or REL).

Reserved: Unused field. MUST be set to 0 by sender and ignored by recipient.

List of PC: A list of Prefix Collection structures.

#### 6.1. Format of the Prefix Collection structure

A Prefix Collection is a structure that holds a list of prefixes. It is composed of a Prefix Collection header followed by a list of Prefix Information structures. It can be seen as the equivalent of an IA\_PD option in DHCPv6\_PD. The format of the Prefix Collection header is as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Length   | Status Code |           Reserved           |
+-----+-----+-----+-----+-----+-----+-----+
|                                     PC_ID                                     |
+-----+-----+-----+-----+-----+-----+-----+
|                                     Renew Time                               |
+-----+-----+-----+-----+-----+-----+-----+
|                                     Rebind Time                             |
+-----+-----+-----+-----+-----+-----+-----+
|                                     List of PI                               |
.                                     .
.                                     .
+-----+-----+-----+-----+-----+-----+-----+

```

Length: The length in blocks of 64 bits of the Prefix Collection including the Prefix Collection header and all the Prefix Informations associated with it.

Status Code: An unsigned integer value that gives informations about the success or failure of an operation along with other additionnal informations. When processing RS messages, this value MUST be set to FULL\_SUCCESS by the sender and ignored by the recipient. More details are presented in Section 10.

Reserved: Unused field. MUST be set to 0 by sender and ignored by recipient.

PC\_ID: An unique identifier of the Prefix Collection. The PC\_ID MUST be unique among all PC\_ID known by the requesting router.

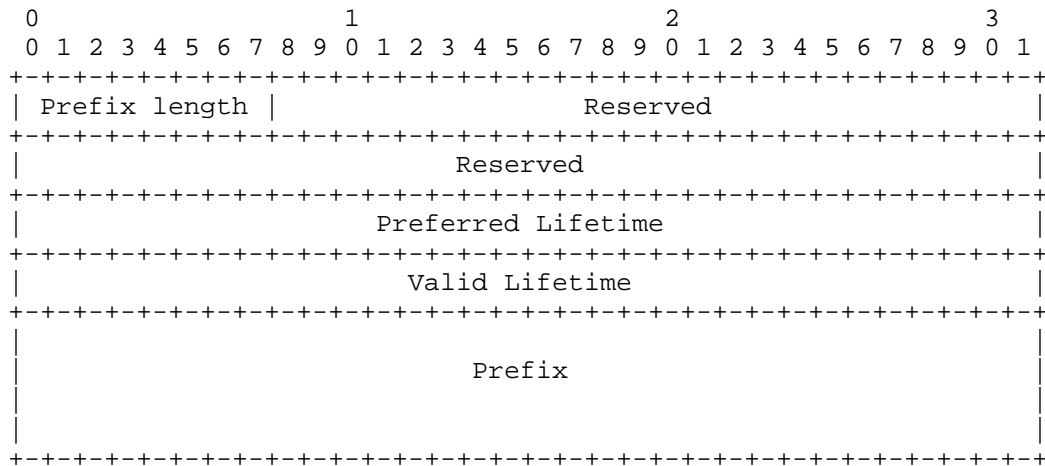
Renew Time: Time length in seconds (relative to the time the packet is sent) before which the requesting router should contact the delegating router from which the Prefix Collection has been received to extend the lifetime of all the prefixes associated with this Prefix Collection.

Rebind Time: Time length in seconds (relative to the time the packet is sent) before which the requesting router should contact any delegating router that is present on the link to extend the lifetime of all the prefixes associated with the Prefix Collection.

List of PI: A list of Prefix Information structures.

## 6.2. Format of the Prefix Information structure

A Prefix Information is a structure that holds all informations related to a prefix. A Prefix Information can be seen as the equivalent of an IA\_PD Prefix option in DHCPv6\_PD. The format of the Prefix Information is as follow:



- Prefix length:** The length of the prefix.
- Reserved:** Unused field. MUST be set to 0 by sender and ignored by recipient.
- Preferred Lifetime:** Time length in seconds (relative to the time the packet is sent) during which addresses generated from this prefix remain preferred (see [SLAAC]). A value of all one bits represents infinity.
- Valid Lifetime:** Time length in seconds (relative to the time the packet is sent) during which the prefix is valid and can be used by nodes for auto configuration (see [SLAAC]). A value of all one bits represents infinity.
- Prefix:** The IPv6 delegated prefix. All bits in the prefix positionned after the prefix length MUST be set to 0.



## 7. Prefix Delegation packets format

### 7.1. Requesting prefixes

The request for prefixes is done using the REQ operation. Thus, the "Msg. Type" field in the Prefix Delegation option of the RS message MUST be set to REQ.

When requesting prefixes a requesting router MUST add for each requested prefix a Prefix Information in the Prefix Delegation option of the RS message. For example, if a router requests three prefixes, three Prefix Information MUST be included in the RS message. The requested prefixes MUST also be associated to a Prefix Collection. The number of Prefix Information associated to a Prefix Collection is left to the choice of the requesting router. However a Prefix Information MUST be associated with only one Prefix Collection. The format of the RS message is as follows:

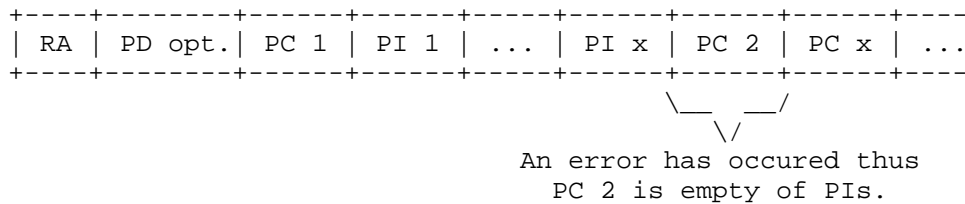
```
+-----+-----+-----+-----+-----+-----+-----+
| RS | PD opt. | PC 1 | PI 1 | ... | PI x | PC x | ...
+-----+-----+-----+-----+-----+-----+-----+
```

The fields "Renew Time" and "Rebind Time" in the Prefix Collection header and all the fields in the Prefix Information MAY be set to values that the requesting router prefers. If the requesting router has no preference, these fields MUST be set to 0. Also notice that the PC\_ID value in the Prefix Collection header is left to the choice of the requesting router, following the rule that a PC\_ID MUST be unique among all PC\_ID known by the requesting router.

### 7.2. Delegating prefixes

The delegation of prefixes is done using the REP operation. Thus, the "Msg. Type" field in the Prefix Delegation option of the RA message MUST be set to REP.

Upon reception of a RS message, a delegating router replies with a RA message that includes all delegated prefixes. If, for some reasons, the request (or part of the request) cannot be fulfilled, the delegating router replies with only the Prefix Collection header and the associated Status Code. The format of the RA message is as follows:



### 7.3. Renewing/rebinding prefixes

The renew or rebind of prefixes is done using the REN or REB operation respectively. Thus, the "Msg. Type" field in the Prefix Delegation option of the RS message MUST be set to REN or REB.

When the Renew Time of a Prefix Collection expires, a requesting router SHOULD renew the Prefix Collection in order to continue using the associated delegated prefixes. To this end, the requesting router MUST send a RS message to the delegating router that delegated it the Prefix Collection. The RS message MUST contain all Prefix Collections (including all the Prefix Informations associated to them) that need to be renewed. The format of the RS message is the same as the one presented in Section 7.1.

When the Rebind Time of a Prefix Collection expires, the behaviour of the requesting router is the same as in the renew case. The only difference is that the RS message MUST be sent to all routers present in the link.

The behaviour of a delegating router that receives a Renew or Rebind request is as follows:

1. The delegating router can identify a Prefix Collection (using its PC\_ID): it updates the Renew and Rebind Times of the Prefix Collection and the Preferred and Valid Lifetimes of all the prefixes associated with this Prefix Collection and adds in its RA message ("Msg. Type" = REP) the Prefix Collection (including all the Prefix Information associated with it).
2. The delegating router cannot identify a Prefix Collection: it adds in the RA message only the Prefix Collection header with an appropriate Status Code.

The format of the RA message is the same as the one presented in Section 7.2.

#### 7.4. Releasing prefixes

The release of prefixes is done using the REL operation. Thus, the "Msg. Type" field in the Prefix Delegation option of the RS message MUST be set to REL.

When a requesting router does not use anymore a prefix, it SHOULD release it. To this end, the requesting router SHOULD send a RS message to the delegating router that delegated it the prefix. The RS message MUST include the Prefix Collection header to which the releasing prefix is associated with and MUST only include the Prefix Informations related to the prefixes that should be released. For example, if a requesting router has a Prefix Collection with four prefixes associated with it and it wants to release one of the four, only the Prefix Information of this prefix MUST be added in the RS message. The format of the RS message is the same as the one presented in Section 7.1.

When a delegating router receives a request for releasing prefixes, it replies with a RA message ("Msg. type" = REP) that includes either only the Prefix Collection header and an appropriate Status Code if an error occurred and the release cannot be fulfilled, or the Prefix Collection header and the Prefix Informations associated with it that have been successfully released. Thus, this reply acts as an acknowledgment of the prefixes that have been successfully released. The format of the RA message is the same as the one presented in Section 7.2.

## 8. Advertisement of the ND\_PD service

Each router that runs the ND\_PD mechanism as described in this document MUST advertise the nodes about this service. To this end the new D flag ("prefix Delegation" flag) is added in the header of the RA messages (according to [RAFLAGS] and [IANAWEB] the flag space could be allocated at position 6). If set, this flag indicates that the router that originated the RA message provides the ND\_PD service. Figure 2 shows the modified header of the RA messages that includes this new flag.

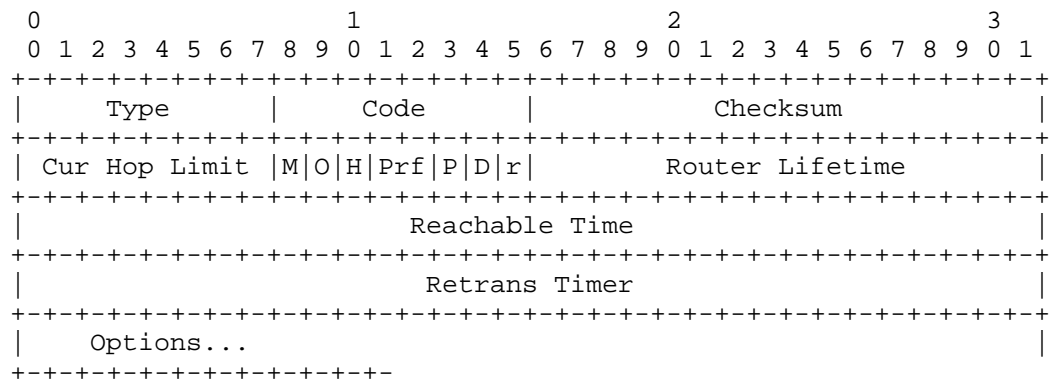


Figure 2: Header of a RA message

M: Managed Address Configuration Flag [NEIGHDISC]  
 O: Other Configuration Flag [NEIGHDISC]  
 H: Mobile IPv6 Home Agent Flag [MIPV6]  
 Prf: Router Selection Preferences [DRPREF]  
 P: Neighbor Discovery Proxy Flag [NDPROXY]  
 D: The prefix Delegation flag that indicates the ND\_PD service provided by the originating router.  
 r: Reserved.

Please refer to section 4.2 of [NEIGHDISC] for more details about the other fields.

## 9. Local operations on requesting and delegating routers

Upon reception of a RS or a RA message, requesting and delegating routers MUST first check the validity of the message as described in section 6.1. "Message Validation" of [NEIGHDISC]. The processing of the message itself along with any option other than the Prefix Delegation option described in this document is out of the scope of this document.

### 9.1. Delegating router behaviour

Upon reception of a REQ message, the delegating router checks the possibility to delegate the requested prefixes to the client. If there are prefixes that are available in the pool to satisfy the whole or part of the request, the delegating router MUST add in its routing table one entry for each delegated prefix via the link-local address of the requesting router, and MUST reply with a REP message that includes all the delegated prefixes and MUST set the status code of the message to either FULL\_SUCCESS (if all requested prefixes are delegated) or PARTIAL\_SUCCESS (if only part of the requested prefixes are delegated). All the prefixes that have been delegated MUST NOT be delegated again before being released either by a REL message coming from the requesting router or by reaching the valid lifetime associated to the prefix. If the request cannot be satisfied, the delegating router MUST reply with a REP message that includes no prefix and MUST set the status code to NO\_PREF\_AVAIL.

Upon reception of a REN message, the delegating router checks the possibility to renew each prefix present in the message. For each prefix that can be renewed, the delegating router MUST update the corresponding entry in its routing table. A REP message MUST then be replied with either the FULL\_SUCCESS or the PARTIAL\_SUCCESS status code depending on the number of prefixes that have been successfully renewed. If none of the prefixes can be renewed, the delegating router MUST remove all corresponding entries in its routing table and MUST reply with a REP message whose status code is set to NO\_PREF\_AVAIL.

Upon reception of a REB message, the delegating router checks the possibility to rebind each prefix included in the message. For each successfully rebinded prefix, the delegating router MUST update its routing table by adding one entry for each delegated prefix via the link-local address of the requesting router. A REP message MUST then be replied with either the FULL\_SUCCESS or the PARTIAL\_SUCCESS status code depending on the number of prefixes that have been successfully rebinded. If none of the prefixes can be rebinded, the delegating router MUST reply with a REP message with a status code set to NO\_PREF\_AVAIL.

Upon reception of a REL message, the delegating router checks the possibility to release each prefix included in the REL message. For each successfully released prefix, the delegating router MUST update its routing table by removing the corresponding entries. A REP message that includes all successfully released prefixes as well as the corresponding FULL\_SUCCESS or PARTIAL\_SUCCESS status code MUST be replied. If, for any reason, none of the prefixes can be released, the delegating router MUST reply with a REP message that do not include any prefix and with the NO\_PREF\_AVAIL status code.

When the Valid Lifetime of a delegated prefix has been reached, the delegating router MUST update its routing table by removing the corresponding entry. The expired prefix is then available for future delegation.

#### 9.2. Requesting router behaviour

Upon reception of a REP message that includes delegated prefixes (i.e. in response of a REQ, REN or REB message), the requesting router is free of advertising the prefixes included in the REP message on any of its interfaces except the one from which the REP message was received. For each prefix that is advertised on an interface, the requesting router MUST add (or update if the entry already exists) an entry for the prefix through the interface where it is advertised.

If a REP message is received in response of a REL message, the requesting router MUST stop advertising any prefix that is included in the REP message. The requesting router MUST also update its routing table by removing each entry that matches the prefixes included in the REP message.

If the Valid Lifetime of a delegated prefix is reached, the requesting router MUST stop advertising the expired prefix and MUST update its routing table by removing the corresponding entry.

## 10. Status codes

Status codes are present in the Prefix Collection headers. They are used to give additionnal information to a requesting router about the result of its request. Therefore, all Prefix Collections included in a REP message MUST include a Status code related to their current status. On the contrary, the Status code has no utility when sending requesting messages (REQ, REN, REB and REL messages). Thus, the Status code of a Prefix Collection MUST be set to FULL\_SUCCESS (default value) by the requesting router and ignored by the recipient. The following Status codes are supported:

FULL_SUCCESS	Indicates that the request has been successfully processed: the requested operation (REQ, REN, REB or REL) has been successfull on all prefixes listed in the Prefix Collection.
PARTIAL_SUCCESS	Informs that only part of the request has been successfully processed: the requested operation has been successfull only on part of the prefixes listed in the requested Prefix Collection. Thus, the replied Prefix Collection includes only the successfull prefixes.
NO_PREF_AVAIL	Informs that no prefixes are available for delegation.

## 11. Security Considerations

TBD



## 12. IANA Considerations

IANA is kindly requested by the authors to allocate the following values:

- o Prefix Delegation option type, which should be added to the Neighbor Discovery option type space defined in section 13 of [NEIGHDISC]
- o Prefix Delegation Message Type:
  - \* REQ
  - \* REP
  - \* REN
  - \* REB
  - \* REL
- o Status Codes:
  - \* FULL\_SUCCESS
  - \* PARTIAL\_SUCCESS
  - \* NO\_PREF\_AVAIL
- o Space allocation for the D flag in the header of RA messages.

### 13. Acknowledgements

The authors would like to acknowledge the useful technical contribution of Sofian Imadali.

This work has been performed in the framework of the ICT project ICT-5-258512 EXALTED, which is partly funded by the European Union. The organisations on the source list [CEA] would like to acknowledge the contributions of their colleagues to the project, although the views expressed in this contribution are those of the authors and do not necessarily represent the project.

## 14. References

## [KEYWORDS]

Bradner, S., "Key words for use in RFCs to indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

## [DHCPV6\_PD]

Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCPv6) version 6", RFC 3633, December 2003.

## [NEIGHDISC]

Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

## [SLAAC]

Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

## [DRAFT\_LUTCHANSKY]

Lutchansky, N., "IPv6 Router Advertisement Prefix Delegation Option",  
draft-lutchann-ipv6-delegate-option-00.txt ,  
February 2002.

## [DRAFT\_HABERMAN]

Haberman, B. and J. Martin, "Automatic Prefix Delegation Protocol for Internet Protocol Version 6 (IPv6)",  
draft-haberman-ipngwg-auto-prefix-02.txt , February 2002.

## [RAFLAGS]

Haberman, B. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 4861, March 2008.

## [IANAWEB]

"<http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xml#icmpv6-parameters-11>".

## [MIPV6]

Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, June 2004.

## [DRPREF]

Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

## [NDPROXY]

Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, April 2006.

Authors' Addresses

Arnaud Kaiser  
Commissariat a l'Energie Atomique  
8 Avenue de la Vauve  
Palaiseau, Ile-de-France 91120  
FR

Phone: +33 1 69 08 07 28  
Email: arnaud.kaiser@cea.fr

Sylvain Decremps  
Commissariat a l'Energie Atomique  
8 Avenue de la Vauve  
Palaiseau, Ile-de-France 91120  
FR

Email: sylvain.decremps@cea.fr

Alexandru Petrescu  
Commissariat a l'Energie Atomique  
8 Avenue de la Vauve  
Palaiseau, Ile-de-France 91120  
FR

Phone: +33 1 69 08 92 23  
Email: alexandru.petrescu@cea.fr



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 2013

H. Kitamura  
NEC Corporation  
S. Ata  
Osaka City University  
October 16, 2012

Corresponding Auto Names for IPv6 Addresses  
<draft-kitamura-ipv6-auto-name-03.txt>

Abstract

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses. With this mechanism, all IPv6 addresses (even if they are link-local scoped addresses) can obtain their own Names, and it will be able to use Names anywhere instead of IPv6 Addresses.

IPv6 address is too long and complicated to remember, and it is very nuisance to type a literal IPv6 address manually as an argument of applications. Also, it is very difficult for human beings to tell which IPv6 address is set to which actual IPv6 node. In this sense, literal IPv6 address information can be called meaningless information for human beings.

In order to solve above problems and to provide annotated meaningful information to IPv6 addresses, mechanisms called Corresponding Auto Names for IPv6 addresses is introduced. They will become human friendly information. By applying a simple naming rule to the Auto Names (e.g., use the same Auto Name Suffix for IPv6 addresses that are set to the same interface (node)), this will contribute to help people to understand which IPv6 address / Name indicates which actual IPv6 node and provide meaningful information.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 2013.

#### Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction	3
2. Goals (What can be achieved)	4
2.1. Assumed typical IPv6 communication environment:	4
2.2. Auto Names examples	4
2.3. Auto Name Suffix for Grouped Addresses	5
2.4. Contribution in Regular Resolving (Name -> Address)	6
2.5. Contribution in Reverse Resolving (Address -> Name)	6
3. Deployed Notions and Functions that are used in Auto Names	8
3.1. Stateless Name	8
3.2. Scoped Name	9
3.3. Target IPv6 Addresses	10
4. Design of Auto Names	11
4.1. Conceptual Design on Naming Rules	11
4.1.1. <P> Value:	11
4.1.2. <I> Value:	12
4.1.3. <NGI> Value:	12
4.2. Address Type Distinction	13
4.2.1. EUI64-based Address Identification	13
4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction	13
5. Name Services	15
6. Security Considerations	15
7. IANA Considerations	15

Appendix A. . . . .	16
A. IPv6 Address Appearance Detection and Auto Name Registration .	16
A.1. IPv6 Address Appearance Detection mechanism . . . . .	16
A.2. Auto Names Generation and Registration mechanism . . . . .	16
A.3. Placement of Detector and Registrar . . . . .	17
A.4. Detection and Registration Procedures . . . . .	18
Appendix B. Implementation . . . . .	19
Acknowledgment . . . . .	19
References . . . . .	19
Authors' Addresses . . . . .	20

## 1. Introduction

This document discusses notion and actual mechanisms of "Corresponding Auto Names" for IPv6 Addresses.

IPv6 address is too long and complicated to remember. For human being, values of EUI64-based or temporary addresses should be felt that they are random number series. So, it is very nuisance to type a literal IPv6 address manually as an argument of applications.

Furthermore, it is very normal and popular cases to set multiple IPv6 addresses to one node. One IPv6 node owns more than two IPv6 addresses (typically: one is link-local scoped address. the other is global scoped address) at least. Some IPv6 addresses (such as link-local scoped stateless auto-configuration addresses and temporary addresses) may become users' conscious-less address, because they are automatically set to the IPv6 node.

It is too difficult for human beings to tell which IPv6 address is set to which IPv6 node. In other words, when an IPv6 address is shown to a person, he almost can not tell that the shown IPv6 address indicates which IPv6 node. In this sense, literal IPv6 address information can be called useless or meaningless information for human beings.

Moreover, when more than two literal IPv6 addresses are shown to human beings, it is almost impossible to distinguish them whether they are same or not at a glance.

So, there are strong desires to use Name information (that is human friendly) instead of literal IPv6 Address information and to use meaningful and distinguishable information that can easily show which IPv6 address / Name indicates which actual IPv6 node.

The Corresponding Auto Names for IPv6 Addresses is introduced to solve above problems and to satisfy the above desires.



## 2. Goals (What can be achieved)

In this section, goals of the mechanisms of the Corresponding Auto Names for IPv6 Addresses and what can be achieved are shown by using examples.

### 2.1. Assumed typical IPv6 communication environment:

Two IPv6 nodes (Node A and Node B) are located on the same link. Their IPv6 Addresses are shown below.

Node A:	Literal Address
-----	
MAC Address:	00:0d:5e:b8:80:7b
-----	
LL-Address:	fe80::20d:5eff:feb8:807b%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b
	fd01:2345:6789::1234
Global Addr:	2001:db8::20d:5eff:feb8:807b
	2001:db8::1234
Node B:	Literal Address
-----	
MAC Address:	00:0c:76:d9:14:e3
-----	
LL-Address:	fe80::20c:76ff:fed9:14e3%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3
	fd01:2345:6789::5678
Global Addr:	2001:db8::20c:76ff:fed9:14e3
	2001:db8::5678

They own altogether 5 IPv6 addresses respectively;

One Link-Local scoped Address

Two Unique Local Addresses (SLLAC and manual set address)

Two Global scoped Addresses (SLLAC and manual set address)

They communicate each other.

### 2.2. Auto Names examples

For all addresses, respective Corresponding Auto Names are prepared and registered to a some name resolving service DB (typically the DNS is used for this) automatically by the mechanism that detects these addresses (that is explained after in this document).

Prepared Auto Names are shown below.

Node A:	Literal Address	Auto Name
	-----	-----
MAC Address:	00:0d:5e:b8:80:7b	
	-----	
LL-Address:	fe80::20d:5eff:feb8:807b%fxp0	-> L0-7bz%fxp0
ULA:	fd01:2345:6789::20d:5eff:feb8:807b	-> U0-7bz
	fd01:2345:6789::1234	-> U1-7bz
Global Addr:	2001:DB8::20d:5eff:feb8:807b	-> G0-7bz
	2001:DB8::1234	-> G1-7bz
Node B:	Literal Address	Auto Name
	-----	-----
MAC Address:	00:0c:76:d9:14:e3	
	-----	
LL-Address:	fe80::20c:76ff:fed9:14e3%em0	-> L0-3ez%em0
ULA:	fd01:2345:6789::20c:76ff:fed9:14e3	-> U0-3ez
	fd01:2345:6789::5678	-> U1-3ez
Global Addr:	2001:DB8::20c:76ff:fed9:14e3	-> G0-3ez
	2001:DB8::5678	-> G1-3ez

### 2.3. Auto Name Suffix for Grouped Addresses

In order to make Auto Names meaningful, IPv6 addresses are grouped and Auto Name Suffix is used to show grouped addresses.

For IPv6 addresses that are set to the same interface (node), the same Auto Name Suffix that stands for the Group ID is used for their Auto Names.

As shown above:

'-7bz' is used for Auto Name Suffix (Group ID) for Node A.  
 '-3ez' is used for Auto Name Suffix (Group ID) for Node B.

In order to make easier to identify and remember the Auto Name Suffixes, their naming rule is based on inheriting the last octet of the node's MAC address in this example.

#### 2.4. Contribution in Regular Resolving (Name -> Address)

In order to communicate with the specific IPv6 address of the destination node, the following procedure to type literal IPv6 address is required in the current environment. They are very stressful and nuisance procedures for human beings.

When 'ping6' or 'telnet' to the specific IPv6 address of Node B from Node A is executed, the following commands are typed.

```
>ping6 fe80::20c:76ff:fed9:14e3%fxp0
>telnet fd01:2345:6789::20c:76ff:fed9:14e3
```

Especially for link-local scoped addresses or temporary addresses, there are no way to type Names instead of literal IPv6 addresses, because they are generally not registered to name resolving services.

By introducing the Corresponding Auto Names, above typed commands are changed and replaced with the following easy and rememberable name typing procedures.

```
>ping6 L0-3ez%fxp0
>telnet U0-3ez
```

#### 2.5. Contribution in Reverse Resolving (Address -> Name)

Communication related status information is shown to human beings in literal IPv6 address format in the current environment.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
fe80::20d:5eff:feb8:807b.8722	fe80:3::20c:76ff:fed9:14e3.23	ESTABLISH
fd01:2345:6789::1234.16258	fd01:2345:6789::5678.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
fe80::20d:5eff:feb8:807b%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
fd01:2345:6789::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::20d:5eff:feb8:807b	0:0d:5e:b8:80:7b	fxp0	permanent	R
2001:DB8::1234	0:0d:5e:b8:80:7b	fxp0	permanent	R
fe80::221:85ff:fea7:82ff%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
fe80::20c:76ff:fed9:14e3%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
fd01:2345:6789::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
fd01:2345:6789::5678	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
2001:DB8::20c:76ff:fed9:14e3	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
2001:DB8::5678	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

People almost can not tell which shown literal IPv6 address indicates which IPv6 node. In this sense, shown information is meaningless and useless.

By introducing the Corresponding Auto Names, above complicated information is converted into simple and meaningful information and shown as followed.

'netstat -a' (on Node A) shows connection status as followed:

Local Address	Foreign Address	(state)
L0-7bz.8722	L0-e3z.23	ESTABLISH
U0-7bz.16258	U0-e3z.23	TIME_WAIT

'ndp -a' (on Node A) shows neighbor cache status as followed:

Neighbor	Linklayer Addr.	Netif	Expire	S
L0-7bz%fxp0	0:0d:5e:b8:80:7b	fxp0	permanent	R
U0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
U1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G0-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
G1-7bz	0:0d:5e:b8:80:7b	fxp0	permanent	R
L0-ffz%fxp0	0:21:85:a7:82:ff	fxp0	23h50m51s	S
L0-3ez%fxp0	0:0c:76:d9:14:e3	fxp0	23h51m56s	S
U0-3ez	0:0c:76:d9:14:e3	fxp0	23h52m50s	S
U1-3ez	0:0c:76:d9:14:e3	fxp0	23h53m51s	S
G0-3ez	0:0c:76:d9:14:e3	fxp0	23h54m53s	S
G1-3ez	0:0c:76:d9:14:e3	fxp0	23h55m54s	S

Other examples where the Auto Name technique can contribute:

In log files of a server application, accesses from clients are recorded into them in literal IPv6 address format. It is almost impossible to read and understand the log files effectively without this Auto Name technique.

Also, in packet dumping applications, address information is shown in literal IPv6 address format. This Auto Name technique can significantly help for human beings to analyze and understand dumped packets.

Shown communication related status information in Auto Name format is simple and easy enough for human beings to understand. As shown above, troublesome IPv6 literal Address information can be converted into meaningful and distinguishable information by using the Corresponding Auto Names technique, and we can achieve our goals.

### 3. Deployed Notions and Functions that are used in Auto Names

#### 3.1. Stateless Name

We know that we can categorize Addresses into two types. One is "stateful" address type, and the other is 'stateless' address type.

On the other, we have not been applied the same categorization to domain Names or host Names clearly. It has been assumed that existing all Names are categorized into stateful type and there is no stateless name type. Authors think that it is a time to change this preconception.

We can grasp that the introduced Corresponding Auto Name is realization of "stateless" name type, and we have deployed a notion Stateless Name clearly here.

Table 1 Stateless Name

	Stateful	Stateless
Address	DHCPv6	SLAAC
Nmae	existing Domain Names	Auto Names

### 3.2. Scoped Name

We also know that a notion called "scope" (such as link-local scope, global-scope) is introduced when we deal with addresses. Every address has its own scope.

In domain names or host names cases, the "scope" notion have NOT clearly introduced now. It is generally assumed that all names are global information and "scope" notion does not exist.

The Corresponding Auto Name is achieved by introducing Scoped Name obviously.

Scope of Auto Name for IPv6 address is the same to the scope of its IPv6 address. For example, scope of the Auto Name for the link-local IPv6 address is link-local. They are only effective within the link-local scope.

Table 1 Scoped Name

	Global	Site-Local (ULA)	Link-Local	Node-Local
Address	2001:db8::/64	fd01:2345:6789::/64	fe80::/64	
Nmae	Domain Names	Domain Names / Auto Names	Auto Names	Auto Names

At some special situation (that it is enough that Auto Name information is shared within a node), Node-Local scoped Auto Name is possible. At such a situation, we can use /etc/hosts file as a name service.

Deployment of Scoped Name:

Scoped Name notion can be easily achieved with current technologies.

As shown above, at a special case when we adopt /etc/hosts file as a name service for Auto Names, scope of Auto Names naturally becomes Node-Local.

At general cases when we adopt the DNS as a name service for Auto Names, scope of Auto Names is easily managed by the DNS query access permission control on DNS servers.

### 3.3. Target IPv6 Addresses

One of the goals of the Auto Name technique is to provide and set Names to all IPv6 addresses (include Link-local addresses).

All IPv6 Addresses are targets of Auto Names.

Some IPv6 Addresses have their own names (let's call them Proper Names) that are assigned manually and are registered into name resolving services (such as the DNS). It may be thought that it is not necessary to assign Auto Name to such IPv6 addresses which have Proper Names. However, we strongly recommend assigning Auto Names to them, too.

In order to provide meaningful information to IPv6 addresses, uniformed Name information for all IPv6 addresses is necessary. This is very natural behavior for 'Stateless' type technology.

Since one-to-multiple mapping is allowed in name resolving services, it will not cause problems to assign both Proper Name and Auto Name for one IPv6 address.

We may need some function that controls name display priority (which name is first Proper Name or Auto Name). This function also achieved easily by using existing current technologies.

If Auto Name and Proper Name are implemented as different name resolving services (e.g., one is /etc/hosts, the other is the DNS), name display priority is can be easily controlled by nsswitch.conf function.

If Auto Name and Proper Name are implemented as same name resolving service, name display priority is can be controlled by their registration order to the name resolving service DB.

## 4. Design of Auto Names

### 4.1. Conceptual Design on Naming Rules

Auto Names are composed of "<P><I>--<NGI>" format:

<P>: stands for Prefix of Address

1 character: (e.g., 'L', 'U', 'G')

<I>: stands for Interface ID of Address

1 character: (e.g., '0', '1')

<NGI>: stands for Node (Interface) Group ID

3 characters:  
(e.g., '7bz', '3ez')

Above discussed Auto Name examples satisfy <P><I>--<NGI> format.

on Node A: L0-7bz, U0-7bz, U1-7bz, G0-7bz, G1-7bz

on Node B: L0-3ez, U0-3ez, U1-3ez, G0-3ez, G1-3ez

#### 4.1.1. <P> Value:

<P> value stands for Prefix (Scope) (upper 64 bit) of Address as 1 character format.

Auto Names of IPv6 addresses whose prefixes are same use the same <P> value.

Typically, following characters are used for <P> value:

"L": used for Link-local scoped addresses.

"U": used for ULA

"G": used for Global scoped address

If multiple prefixes for the same scope are used, other character (such as "H", "I",,,, ) can be used depending on the circumstances.

"Prefix - <P> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "Prefix - <P> value" mapping table) is used to avoid collision and manage mappings of them.



## 4.1.2. &lt;I&gt; Value:

<I> value stands for Interface ID (lower 64 bit) of Address as 1 character format.

Following characters are used for <I> value:

<I> value assignment is based on three address type categorization.

"0": used for EUI64-based address  
"1", "9": used for manually set addresses  
(stateful addresses will be categorized here)  
  
"a", "z": used for automatically generated and set addresses  
except EUI64-based  
(Temporary addresses are categorized here)

## 4.1.3. &lt;NGI&gt; Value:

<NGI> value is also called Auto Name-Suffix.

In order to make IPv6 addresses meaningful, IPv6 addresses are grouped. It is very natural to group IPv6 addresses by which node (interface) they are set. So, IPv6 addresses that are set to the same node (interface) are grouped into the same group.

<NGI> value is shown as 'XYZ' format:

'XY': (1st, 2nd chars) are inherited from  
the last octet (2 characters) of the node's MAC address  
'Z' : (3rd char) suffix char to avoid a collision of 'XY'  
starting from "z"  
if 'XY' is collided, 'Z' is changed into "y", "x" , , ,

By using the birthday paradox theorem, collision probability of 256 states (1 octet) is calculated. If 19 nodes (interfaces) exist on the same link, collision is happened with 50% probability. Collision check procedure of the last octet of MAC addresses is necessary.

"MAC address - <NGI> value" mapping table:

If the scope of Auto Name is wider than link-local and Auto Name information is shared with other nodes, a mapping table (called "MAC address - <NGI> value" mapping table) is needed to avoid collision and manage mappings of them.

Detailed methods how to distinguish and categorize IPv6 addresses are described at the following section.

We can found one remarkable thing with the naming rules:

"L0-XYZ" is a special and very standard Auto Name. "L0-XYZ" is an Auto Name that assigned for Link-local scoped EUI64-based address. Since Almost all the IPv6 nodes have Link-local scoped EUI64-based address, with "L0-XYZ" name information we can reach or communicate the node whose last octet MAC address is known as "XY".

#### 4.2. Address Type Distinction

If we can obtain address type information from Auto Name, convenient environment will be provided. So, there are strong desires to understand type of detected IPv6 addresses. In this section, methods how to distinguish and categorize IPv6 addresses are described.

##### 4.2.1. EUI64-based Address Identification

Only with IPv6 address information, it is impossible to identify that the detected IPv6 address is EUI64-based address or not.

In proposed IPv6 address detection method which is described in the following section, IPv6 address and MAC address that are set to the same node (interface) is detected simultaneously.

So, with this detection method, it is very easy to identify that the detected IPv6 address is EUI64-based address or not.

##### 4.2.2. "Zero Contain Rate" and Manual or Automatic Distinction

It is generally difficult to distinguish whether the detected IPv6 address is manually or automatically set address.

In order to distinguish IPv6 address types, "Zero Contain Rate" technique is introduced.

For a human being, "Zero" is a special value. When a human being omits a part of information, "Zero" is used for the omitted part of information implicitly.

For a machine "Zero" is NOT a special value. "Zero" is treated as almost equal to other values.

We can reach a fact that manually set IPv6 address contains many "Zero", because it is assigned by a human being. 64bit is long for

a human being and there must be too many omitted part filled with "Zero". Especially, a human being is apt to omit and fill with "Zero" upper part of 64bit Interface ID.

In other words, we can see human relation bias from "Zero Contain Rate" of IPv6 address.

Bias Check by using Mathematical Technique:

By using mathematical probability technique, we can distinguish whether value of 64bit Interface ID is biased (manual) or non-biased (automatic).

When we see 64bit value in 8bit (1 octet) unit, it follows the binomial distribution ( $n=8$  and  $p=1/256$ ).

Under this distribution, the probability to meet "Zero" octet two times is 0.042%. It means that to meet "Zero" octet two times is too rare case if the value is non-biased.

So, we can adopt the following method:

If the value of 64bit Interface ID contains two and above "Zero" octets,  
the value is bias and  
the IPv6 address is identified as a manually set address.

Of cause, this method is not perfect because this is based on mathematical probability technique and heuristic human behavior, but this is very effective.  
Even if wrong identification is done, no big problems are found.

## 5. Name Services

It is not clearly defined which Name Services is utilized to achieve Auto Names specifications. In other words, any types of Name Services can be utilized. Which Name Services is utilized is tightly dependent on which Scoped Name is adopted for the Auto Names.

If wide Scoped Name is adopted, it is very natural to utilize wide Name Service (i.e. the DNS) as the Name Services for Auto Names. If narrow scope (e.g., node-local scoped name) is adopted for Auto Names, it becomes possible to utilize node-local scoped Name Service (e.g., /etc/hosts) for Auto Name.

## 6. Security Considerations

Auto Names are generated and registered to the name service in this document. In order to register correct Auto Names information, communication between Detector and Registrar and communication between Registrar and Name Server should be protected and be secured.

In general usage, scope of Auto Names will be local (not global). Auto Names are usually local scoped names. So, we do not have to be too sensitive on the correctness of Auto Names.

## 7. IANA Considerations

This document does not require any resource assignments to IANA.

## Appendix A. IPv6 Address Appearance Detection and Auto Name Registration

In order to generate and register Auto Names automatically, two types of mechanisms are needed. One is a mechanism that detects IPv6 address appearance. The other is a mechanism that checks the detected addresses and generates Auto Names and registers them to name service.

Two functions ("Detector" and "Registrar") are introduced. "Detector" function takes in charge of the former mechanism, and "Registrar" takes in charge the latter mechanism.

### A.1. IPv6 Address Appearance Detection mechanism

In order to detect newly appeared IPv6 address, DAD message (NS for DAD) is effectively used.

DAD message has the following good capabilities:

- issued only when node would like to set new IPv6 address
- issued for All types (link-local, global, temporary,...)
- L2 broadcast and easy to capture (without using mirror port)
- distinguishable from other NS messages, because source address of the message is unspecified ("::") and different from others
- Captured DAD message includes all necessary information (such as, IPv6 address and MAC address)

Detector captures DAD messages and detects newly appeared IPv6 addresses. Detected information is sent to Registrar.

### A.2. Auto Names Generation and Registration mechanism

At first, Registrar checks the Detected address information that is sent from Detector(s). By using the reverse resolving (Address -> Name), it is checked whether the Detected address information is first appearance or not. If an entry for the address does NOT exist, it is confirmed that the address is first appearance and it should be registered to the name server.

After Name for the address is prepared, duplication of the Name can be checked by using the regular resolving (Name -> Address). If an entry for the Name exist, it is confirmed that Name is duplicated (collided). Another Name is prepared and checked again until the Name

is not duplicated.

Finally, Registrar registers both Regular and Reverse resolving entries for the address and prepared Auto Name are registered to the name server.

### A.3. Placement of Detector and Registrar

Placement of Detector and Registrar is designed to make the mechanisms flexible and to make it to be applied to various environments (office networks, home networks, etc.)

Figure 1 and 2 show typical examples that indicate locations where Detector and Registrar functions are placed on the IPv6 network. Figure 1 shows a case for a single link, and Figure 2 shows a case for multiple links.

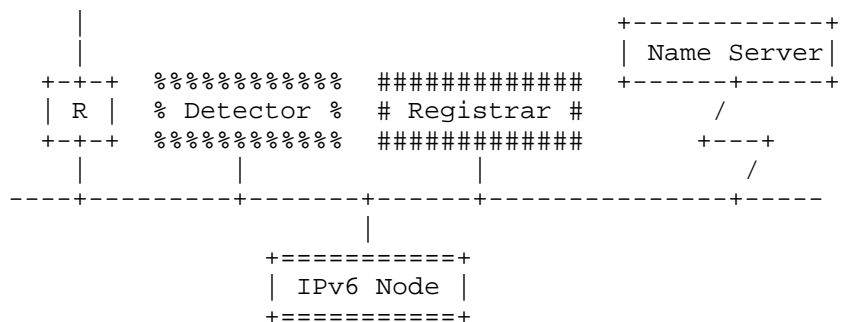


Fig. 1 Single-Link Case Example

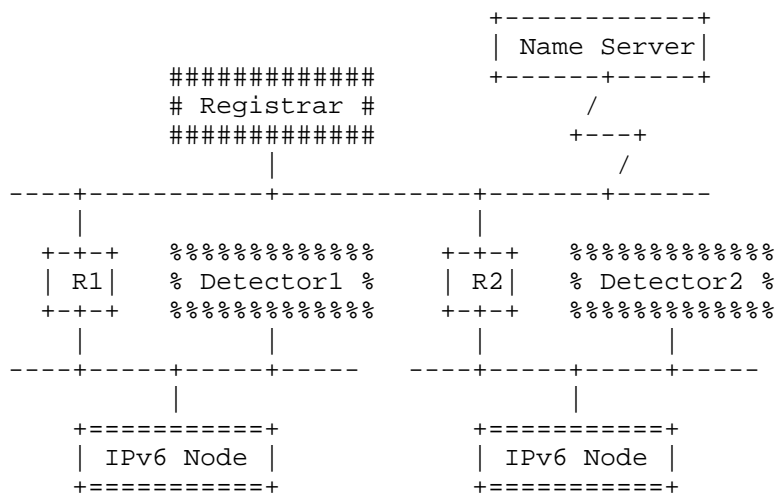
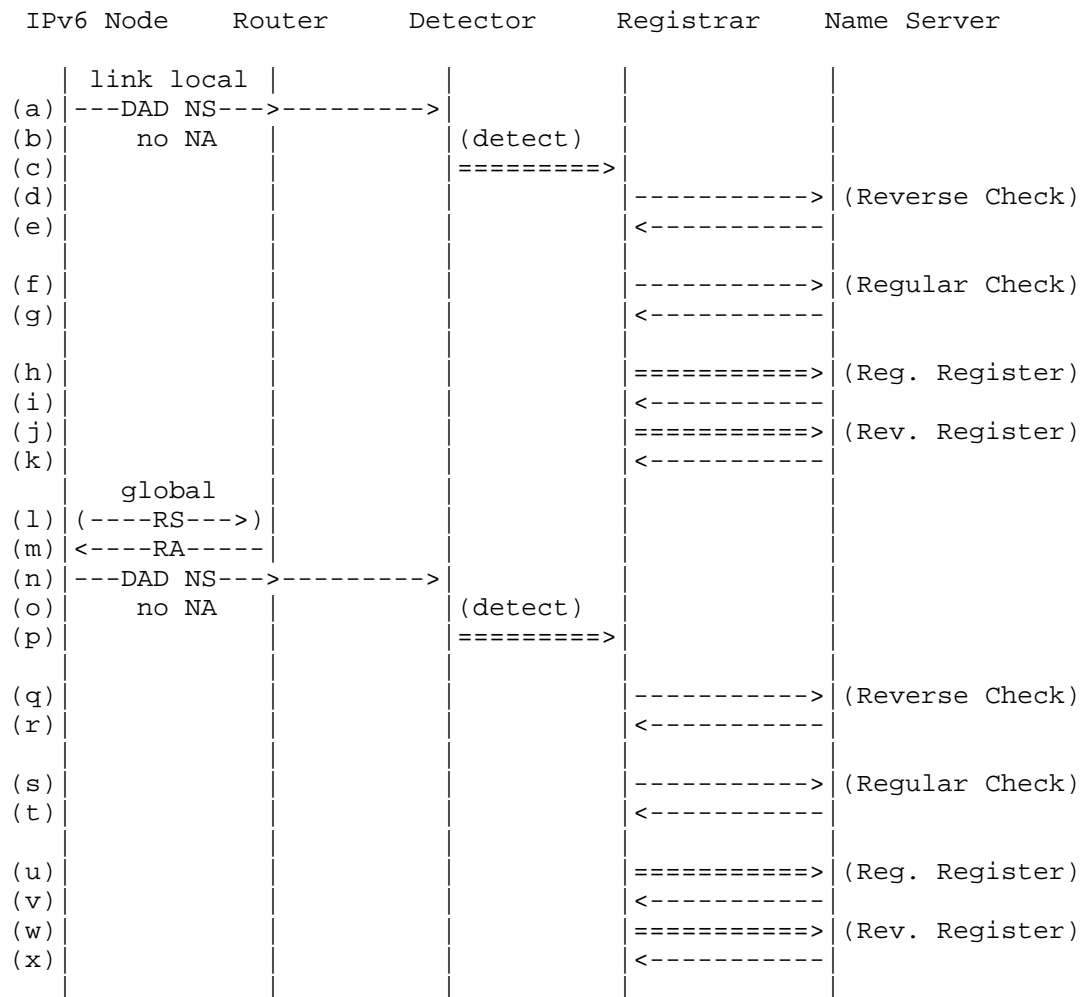


Fig. 2 Multiple-Link Case Example

## A.4. Detection and Registration Procedures

Figure 3 shows an example of typical detection and registration procedures at IPv6 links where DAD packets are issued. DAD message packets are used for the appearance detection.



## Appendix B. Implementation

Auto Name functions have been implemented at the following environments. It has been verified that designed functions work well.

### Used functions:

Packet capture on Detector: libpcap  
Name Server: DNS (BIND9)  
Name Registration: nsupdate (BIND9 bundled)

### OS:

FreeBSD 6.2R  
(Since FreeBSD OS specific functions are not used to implement,  
codes will run on UNIX type OS that has libpcap (such as Linux).)

## Acknowledgment

A part of this work is supported by the program: SCOPE (Strategic Information and Communications R&D Promotion Programme) operated by Ministry of Internal Affairs and Communications of JAPAN.

## References

### Normative References

- [RFC4291] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006
- [RFC4861] T. Narten, E. Nordmark, W. Simpson and H. Soliman, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 4861, September 2007
- [RFC4862] S. Thomson, T. Narten and T. Jinmei "IPv6 Stateless Address Autoconfiguration," RFC4862, September 2007
- [RFC4941] T. Narten, R. Draves and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC4941, September 2007
- [RFC1034] P. Mockapetris, "Domain names - concepts and facilities ", RFC 1034, November 1987
- [RFC1035] P. Mockapetris, "Domain names - implementation and specification", RFC 1035, November 1987
- [RFC2136] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System," RFC 2136, April 1997



[RFC4795] B. Aboba, D. Thaler, and L. Esibov, "Link-Local Multicast Name Resolution (LLMNR)," RFC4795, January 2007

#### Informative References

[RFC4620] M. Crawford and B. Haberman, "IPv6 Node Information Queries," RFC4620, August 2006

[mDNS] S. Cheshire and M. Krochmal, "Multicast DNS" <draft-cheshire-dnsext-multicastdns-14.txt> work in progress, February 2011

[RFC3849] G. Huston, A. Lord and P. Smith, "IPv6 Address Prefix Reserved for Documentation," RFC3849, July 2004

#### Authors' Addresses

Hiroshi Kitamura  
Knowledge Discovery Research Laboratories, NEC Corporation  
(SC building 12F)1753, Shimonumabe, Nakahara-Ku, Kawasaki,  
Kanagawa 211-8666, JAPAN  
Graduate School of Information Systems,  
University of Electro-Communications  
5-1 Chofugaoka 1-Chome, Chofu-shi, Tokyo 182-8585, JAPAN  
Phone: +81 44 431 7686  
Fax: +81 44 431 7680  
Email: kitamura@da.jp.nec.com

Shingo Ata  
Graduate School of Engineering, Osaka City University  
3-3-138, Sugimoto, Sumiyoshi-Ku, Osaka 558-8585, JAPAN  
Phone: +81 6 6605 2191  
Fax: +81 6 6605 2191  
Email: ata@info.eng.osaka-cu.ac.jp

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 11, 2013

B. Sarikaya  
Huawei USA  
July 10, 2012

IPv6 RA Options for Multiple Interface Next Hop Routes  
draft-sarikaya-mif-6man-ra-route-01

Abstract

This draft defines new Router Advertisement options for configuring next hop routes on the mobile or fixed nodes. Using these options, an operator can easily configure nodes with multiple interfaces (or otherwise multi-homed) to enable them to select the routes to a destination. Each option is defined together with definitions of host and router behaviors.

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 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Default Route Configuration . . . . .	4
4. Route Prefix option . . . . .	4
5. Next Hop Address option . . . . .	5
6. Next Hop Address with Route Prefix option . . . . .	6
7. Security Considerations . . . . .	6
8. IANA Considerations . . . . .	6
9. Acknowledgements . . . . .	7
10. References . . . . .	8
10.1. Normative References . . . . .	8
10.2. Informative References . . . . .	8
Author's Address . . . . .	9

## 1. Introduction

IPv6 Neighbor Discovery and IPv6 Stateless Address Autoconfiguration protocols can be used to configure fixed and mobile nodes with various parameters related to addressing and routing [RFC4861], [RFC4862], [RFC4191]. DNS Recursive Server Addresses and Domain Name Search Lists are additional parameters that can be configured using router advertisements [RFC6106].

Router Advertisements can also be used to configure fixed and mobile nodes in multi-homed scenarios with route information and next hop address. Different scenarios exist such as the node is simultaneously connected to multiple access network of e.g. WiFi and 3G. The node may also be connected to more than one gateway. Such connectivity may be realized by means of dedicated physical or logical links that may also be shared with other users nodes such as in residential access networks.

## 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].

### 3. Default Route Configuration

A host, usually a mobile host interested in obtaining routing information usually send a Router Solicitation (RS) message on the link. The router, when configured to do so, provides the route information using zero, one or more Next Hop Address and Route Information options in the router advertisement (RA) messages sent in response.

The route options are extensible, as well as convey detailed information for routes. The router sends one or more Next Hop Address options that specify the IPv6 next hop addresses. Each Next Hop Address option may be associated with zero, one or more Route Prefix options that represent the IPv6 destination prefixes reachable via the given next hop. Router includes Route Prefix option directly in message to indicate that given prefix is available directly on-link. Router MAY send a single Next Hop Address without any Route Prefix options. When router sends Next Hop Address option that is associated with Router Prefix option, the router MUST use Next Hop and Route Prefix option defined in Section 6. The Route Prefix MAY contain `::/0`, i.e. with Prefix Length set to zero to indicate available default route.

RS and RA exchange is for next hop address and route information determination and not for determining the link-layer address of the router. Subsequent Neighbor Solicitation and Neighbor Advertisement exchange can be used to determine link-layer address of the router.

It should be noted that the proposed options in this document will need a central site-wide configuration mechanism. The required values can not automatically be derived from routing tables.

### 4. Route Prefix option

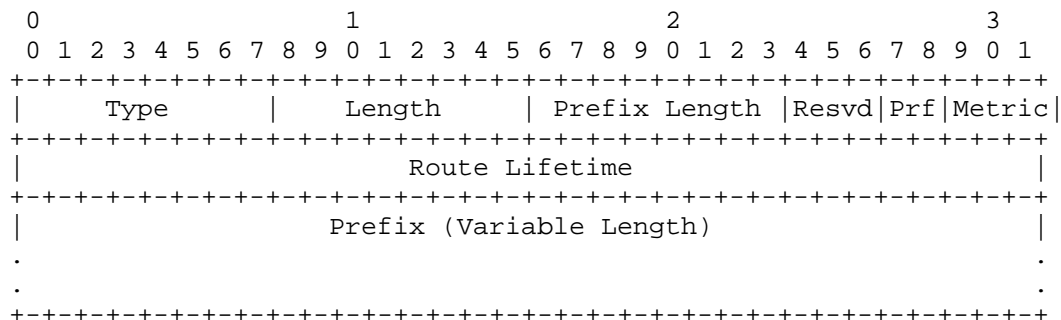


Figure 1: Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the Type and Length fields) in units of 8 octets.

Other fields are as in [RFC4191] except:

Metric Route Metric. 3-bit signed integer. The Route Metric indicates whether to prefer the next hop associated with this prefix over others, when multiple identical prefixes (for different next hops) have been received.

#### 5. Next Hop Address option

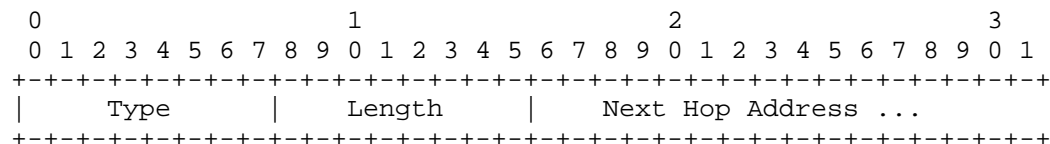


Figure 2: Next Hop Address option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. It's value is 3.

Next Hop Address: An IPv6 address that specifies IPv6 address of the next hop. It is 16 octets in length.

## 6. Next Hop Address with Route Prefix option

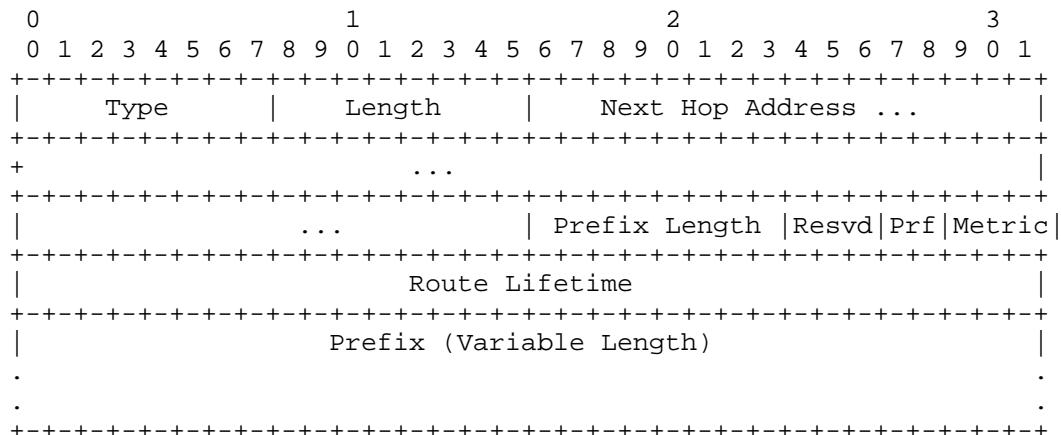


Figure 3: Next Hop Address with Route Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. For example, the length for a prefix of length 16 is 5.

Other fields are as in Section 4 and Section 5.

## 7. Security Considerations

Neighbor Discovery is subject to attacks that cause IP packets to flow to unexpected places. Because of this, neighbor discovery messages MUST be secured, possibly using Secure Neighbor Discovery (SEND) protocol [RFC3971].

## 8. IANA Considerations

Authors of this document request IANA to assign three new RA options:

Option Name	Type
Route Prefix	
Next Hop Address	
Next Hop Address and Route Prefix	

Table 1:

## 9. Acknowledgements

Brian Carpenter provided comments that have led to improvements in the document. We are also grateful to Zhen Cao for his comments.



## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

### 10.2. Informative References

- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.

Author's Address

Behcet Sarikaya  
Huawei USA  
5340 Legacy Dr. Building 175  
Plano, TX 75024

Phone:  
Email: sarikaya@ieee.org



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: February 26, 2013

B. Sarikaya  
Huawei USA  
August 25, 2012

IPv6 RA Options for Translation Multicast Prefixes  
draft-sarikaya-software-6man-raoptions-00

Abstract

This draft defines new Router Advertisement options for configuring multicast prefixes for IGMP to MLD translation and unicast prefixes for multicast source addresses. These prefixes are used in various translation multicast technologies. Each option is defined together with definitions of host and router behaviors.

Status of this Memo

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

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

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

This Internet-Draft will expire on February 26, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Multicast Translation Prefixes Configuration . . . . .	3
4. Architectures of Usage . . . . .	4
4.1. Stateful NAT64 . . . . .	5
4.2. Mapping of Address and Port and 4rd . . . . .	5
4.3. Other Cases . . . . .	6
5. Multicast Translation Prefix Options . . . . .	6
6. Security Considerations . . . . .	8
7. IANA Considerations . . . . .	8
8. Acknowledgements . . . . .	9
9. References . . . . .	10
9.1. Normative References . . . . .	10
9.2. Informative References . . . . .	11
Author's Address . . . . .	12

## 1. Introduction

IPv6 Neighbor Discovery and IPv6 Stateless Address Autoconfiguration protocols can be used to configure fixed and mobile nodes with various parameters related to addressing and routing [RFC4861], [RFC4862], and [RFC4191]. DNS Recursive Server Addresses and Domain Name Search Lists are additional parameters that can be configured using router advertisements [RFC6106].

In some unicast IPv4 to IPv6 transition technologies, IPv4 to IPv6 translation is used such as NAT64 [RFC6145], Mapping of Address and Port Translation mode, or MAP-T [I-D.ietf-softwire-map], and to some extend in IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd) [I-D.ietf-softwire-4rd]. Translation multicast technologies are emerging aimed to complement these unicast solutions for multicast IPv4 to IPv6 transition.

In IPv4 to IPv6 translation multicast, specific IPv6 addresses are used in which IPv4 addresses are embedded [RFC6180]. Some of these addresses are multicast IPv6 addresses and some are unicast. Both multicast and unicast addresses can be generated from the corresponding multicast and unicast prefixes. These prefixes must be provisioned on the hosts that are involved in translation.

## 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].

## 3. Multicast Translation Prefixes Configuration

A host wishing to join a source-specific multicast (SSM) group sends an IGMP Membership Report message and includes the group address in Multicast Address field of Group Record field in IPv4 [RFC3376]. Multicast group address MUST be a multicast group address in SSM range. In the IPv4 to IPv6 transition scenarios that are based on translation multicast, IGMP Membership Report message must be translated into MLD Membership Report message [RFC3810], usually at the IGMP/MLD proxy entity [RFC4605]. SSM group is identified with a group and source address in (S,G) form. The source address is specified in Source Address field of Group Record field in IGMP Membership Report message. There could be more than one source addresses associated with one Multicast Address.

IGMP to MLD translation entity must be configured with an IPv6

multicast group prefix that is in SSM range.

[I-D.ietf-mboned-64-multicast-address-format] defines multicast prefixes (MPREFIX64) for IPv4-embedded IPv6 multicast address based on an IPv4 multicast address of two types: ASM\_MPREFIX64 for any-source multicast (ASM) and SSM\_MPREFIX64 for source-specific multicast (SSM).

IGMP to MLD translation entity can be configured with SSM\_MPREFIX64 by its default router if the default router includes in its Router Advertisements a Multicast SSM Translation Prefix option defined in Figure 1. SSM\_MPREFIX64 is of type ff3x:0:8000/96. Prefix length MAY be set to 96. x stands for the scope bits [RFC3306].

IGMP to MLD translation entity can be configured with ASM\_MPREFIX64 by its default router if the default router includes in its Router Advertisements a Multicast ASM Translation Prefix option defined in Figure 2. ASM\_MPREFIX64 is of type ffx:8000::/20. Prefix length MAY be set to 20. The first x stands for the flags and the second x stands for the scope bits [RFC3306].

SSM groups have one or more sources, S in (S,G). The source address is specified by the joining host in IGMP Membership Report message's Source Address field of Group Record. IPv4 source address in IGMP Membership Report message needs to be translated into an IPv6 source address in the translated MLD message.

IGMP to MLD translation entity can be configured with U\_PREFIX64 by its default router if the default router includes in its Router Advertisements a Multicast Translation Unicast Prefix option defined in Figure 3. U\_PREFIX64 could be assigned to 64:ff9b::/96 which is known as the well-known prefix or to some operator specific value. Prefix length MAY be set to 96. IPv4-Embedded IPv6 Unicast Addresses MUST be formed using the rules defined in [RFC6052].

U\_PREFIX64 MUST also be used in translating IPv4 multicast data packets from SSM sources, S of (S,G) into IPv6 multicast data packets. In this case the translation entity is connected to IPv4 network at the border of an IPv6 network, e.g. Border Router (BR). Border Router MUST also be configured with U\_PREFIX64 by its default router. The default router includes in its Router Advertisements a Multicast Translation Unicast Prefix option defined in Figure 3.

#### 4. Architectures of Usage

#### 4.1. Stateful NAT64

In NAT64 [RFC6145], IGMP to MLD translation entity is the host. The default router of the host MUST be configured to send SSM\_PREFIX64 and ASM\_PREFIX64 in its router advertisements. In some configurations NAT64 box could be the default router then NAT64 box MUST be configured to send these RAs. This includes the case where NAT64 is located in Broadband Network Gateway (BNG) in broadband networks and Packet Data Network Gateway (PDN Gateway) in LTE networks.

The hosts will receive the RAs Multicast ASM Translation Prefix option and Multicast SSM Translation Prefix option and use the prefix (SSM\_PREFIX64 or ASM\_PREFIX64) in translating IGMP messages to MLD messages and vice versa.

In NAT64, Border Router which receives IPv4 multicast data is NAT64 box. The default router of NAT64 box MUST be configured to send Multicast Translation Unicast Prefix option in its router advertisements. This will configure NAT64 box with U\_PREFIX64. NAT64 box can use this to translate IPv4 multicast data coming from SSM sources to IPv6 multicast data.

In case SSM is supported, the default router for the hosts MUST be configured to send Multicast Translation Unicast Prefix option in its router advertisements. The hosts can use U\_PREFIX64 in translating IGMPv3 messages to MLDv2 messages and in translating IPv6 multicast data to IPv4 multicast data.

#### 4.2. Mapping of Address and Port and 4rd

In MAP/4rd [I-D.ietf-softwire-map], [I-D.ietf-softwire-4rd], IGMP to MLD translation entity is the Customer Edge (CE). The default router of CE MUST be configured to send SSM\_PREFIX64 and ASM\_PREFIX64 in its router advertisements.

The CEs will receive the RAs with Multicast ASM Translation Prefix option and Multicast SSM Translation Prefix option and use the prefix (SSM\_PREFIX64 or ASM\_PREFIX64) in translating IGMP messages to MLD messages and vice versa.

In MAP/4rd, the router which receives IPv4 multicast data is the Border Router (BR) box. The default router of BR box MUST be configured to send Multicast Translation Unicast Prefix option in its router advertisements. This will configure BR with U\_PREFIX64. BR can use this to translate IPv4 multicast data coming from SSM sources to IPv6 multicast data.



In case SSM is supported, the default router for the CE MUST be configured to send Multicast Translation Unicast Prefix option in its router advertisements. The CE can use U\_PREFIX64 in translating IGMPv3 messages to MLDv2 messages and in translating IPv6 multicast data to IPv4 multicast data.

#### 4.3. Other Cases

TBD.

### 5. Multicast Translation Prefix Options

In this section we define three prefix options that are related to the translation of IGMP messages into MLD or vice versa.

Figure 1 defines Source Specific Multicast translation prefix option commonly known as SSM\_PREFIX64. It is an IPv6 prefix belonging to source specific multicast (ssm) range with a value similar to ff3x:0:8000/33. It is used in embedding IPv4 SSM range multicast addresses. The values MUST conform to [I-D.ietf-mboned-64-multicast-address-format].

Figure 2 defines Any Source Multicast translation prefix option commonly known as ASM\_PREFIX64. It is an IPv6 prefix belonging to any source multicast (asm) range with a value similar to ffx:x:8000/17. It is used in embedding IPv4 ASM range multicast addresses. The values MUST conform to [I-D.ietf-mboned-64-multicast-address-format]

Figure 3 defines Multicast Translation Unicast prefix option commonly known as U\_PREFIX64. It is an IPv6 unicast prefix. It is used in translating IPv4 source addresses into IPv6 source address or vice versa for source specific multicast sources. [RFC6052] defines two types for U\_PREFIX\_64, well known prefix of 64:ff9b::/96 or the network specific prefix and the mapping rules from IPv4 to IPv6 address and vice versa.

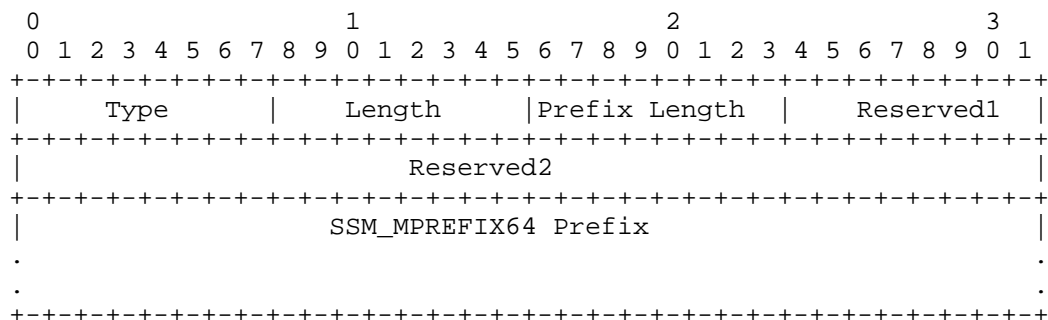


Figure 1: Multicast SSM Translation Prefix option

Fields:

Type: TBD.

Length: The length of the option (including the type and length fields) in units of 8 octets. For example, the length for an IPv6 address is 3.

Prefix Length: 8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.

Reserved1 8-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Reserved2 32-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Prefix: A prefix of type source specific multicast. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and MUST be initialized to zero by the sender and ignored by the receiver.

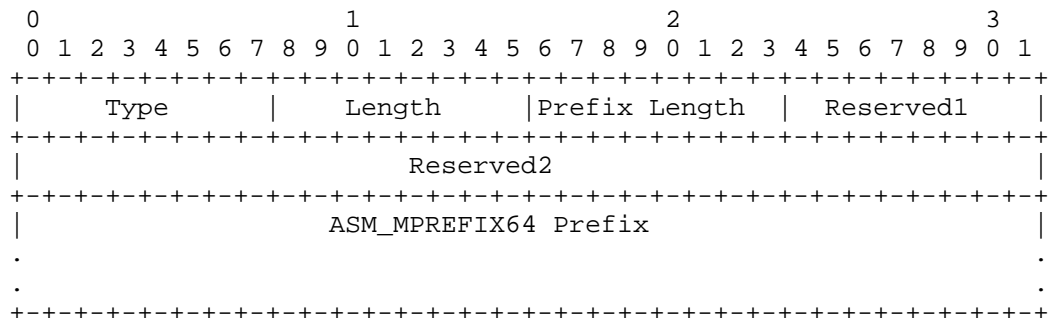


Figure 2: Multicast ASM Translation Prefix option

Fields:

Type: TBD.

Length and other fields are as defined above.

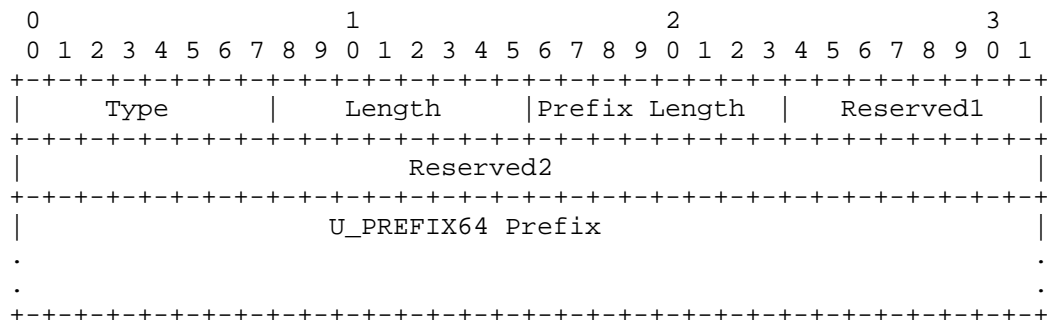


Figure 3: Multicast Translation Unicast Prefix option

Fields:

Type: TBD.

Length and other fields are as defined above.

## 6. Security Considerations

Neighbor Discovery is subject to attacks that cause IP packets to flow to unexpected places. Because of this, neighbor discovery messages MUST be secured, possibly using Secure Neighbor Discovery (SEND) protocol [RFC3971].

## 7. IANA Considerations

Authors of this document request IANA to assign three new RA options:

Option Name	Type
Multicast SSM Translation Prefix option	
Multicast ASM Translation Prefix option	
Multicast Translation Unicast Prefix option	

Table 1:

## 8. Acknowledgements

TBD.

## 9. References

### 9.1. Normative References

- [I-D.ietf-mboned-64-multicast-address-format]  
Boucadair, M., Qin, J., Lee, Y., Venaas, S., Li, X., and M. Xu, "IPv6 Multicast Address With Embedded IPv4 Multicast Address", draft-ietf-mboned-64-multicast-address-format-04 (work in progress), August 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

## 9.2. Informative References

- [I-D.ietf-softwire-4rd]  
Despres, R., Penno, R., Lee, Y., Chen, G., Jiang, S., and M. Chen, "IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd)", draft-ietf-softwire-4rd-03 (work in progress), July 2012.
- [I-D.ietf-softwire-map]  
Troan, O., Dec, W., Li, X., Bao, C., Zhai, Y., Matsushima, S., and T. Murakami, "Mapping of Address and Port (MAP)", draft-ietf-softwire-map-01 (work in progress), June 2012.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.
- [RFC6180] Arkko, J. and F. Baker, "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment", RFC 6180, May 2011.

Author's Address

Behcet Sarikaya  
Huawei USA  
Plano, TX 75075

Phone: +1 972-509-5599  
Email: sarikaya@ieee.org

