

6man WG  
Internet-Draft  
Updates: 4861 (if approved)  
Intended status: Standards Track  
Expires: August 31, 2015

S. Chakrabarti  
Ericsson  
E. Nordmark  
Arista Networks  
P. Thubert  
Cisco Systems  
M. Wasserman  
Painless Security  
February 27, 2015

IPv6 Neighbor Discovery Optimizations for Wired and Wireless Networks  
draft-chakrabarti-nordmark-6man-efficient-nd-07

Abstract

IPv6 Neighbor Discovery (RFC 4861 going back to RFC 1970) was defined at a time when link-local multicast was as reliable and with the same network cost (send a packet on a yellow-coax Ethernet) as unicast and where the hosts were assumed to be always on and connected.

Since 1996 we've seen a significant change with both an introduction of wireless networks and battery operated devices, which poses significant challenges for the old assumptions. We are also seeing datacenter networks where virtual machines are not always on and connected, and scaling of multicast can be challenging.

This specification contains extensions to IPv6 Neighbor Discovery which remove most use of multicast and make sleeping hosts more efficient. The specification includes a default mixed mode where a link can have an arbitrary mix of hosts and/or routers - some implementing legacy Neighbor Discovery and some implementing the optimizations in this specification. The optimizations provide incremental benefits to hosts as soon as the first updated routers are deployed on a 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 August 31, 2015.

#### Copyright Notice

Copyright (c) 2015 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 . . . . .	5
2. Goals and Requirements . . . . .	6
2.1. Mixed-Mode Operations . . . . .	7
3. Changes to ND state management . . . . .	7
4. Definition Of Terms . . . . .	8
5. Protocol Overview . . . . .	9
5.1. Proxying to handle Mixed mode . . . . .	11
6. New Neighbor Discovery Options and Messages . . . . .	11
6.1. Router Advertisement flag for NEARs . . . . .	11
6.2. Address Registration Option (ARO) . . . . .	12
6.3. Registrar Address Option (RAO) . . . . .	14
7. Conceptual Data Structures . . . . .	15
8. Host Behavior . . . . .	16
8.1. Host and/or Interface Initialization . . . . .	16
8.2. Host Receiving Router Advertisements . . . . .	16
8.3. Timing out Registrar List entries . . . . .	17
8.4. Sending AROs . . . . .	17
8.5. Receiving Neighbor Advertisements . . . . .	18
8.6. Host Management of the TID . . . . .	18
8.7. Refreshing a Registration . . . . .	18
8.8. De-registering . . . . .	19
8.9. Refreshing RA information . . . . .	19
8.10. Sleep and Wakeup . . . . .	21
8.11. Receiving Redirects . . . . .	21
8.12. Movement Detection . . . . .	21
9. Router Behavior . . . . .	21
9.1. Router and/or Interface Initialization . . . . .	22
9.2. Receiving Router Solicitations . . . . .	22
9.3. Periodic Multicast RA for legacy hosts . . . . .	23
9.4. Multicast RA when new information . . . . .	23
9.5. Receiving ARO . . . . .	23
9.6. NCE Management in NEARs . . . . .	23
9.7. Sending non-zero status in ARO . . . . .	24
9.8. Timing out Registered NCEs . . . . .	24
9.9. Router Advertisement Consistency . . . . .	25
9.10. Sending Redirects . . . . .	25
9.11. Providing Information to Routing Protocols . . . . .	25
9.12. Creating Legacy NCEs . . . . .	25
9.13. Proxy Address Resolution and DAD for Legacy Hosts . . . . .	25
10. Handling ND DoS Attack . . . . .	26
11. Bootstrapping . . . . .	27
12. Interaction with other protocols . . . . .	28
12.1. Detecting Network Attachment (DNA) . . . . .	28
12.2. DHCPv6 Interaction . . . . .	28
12.3. Other use of Multicast . . . . .	29
12.4. VRRP Interaction . . . . .	29

13. Updated Neighbor Discovery Constants . . . . .	29
14. Security Considerations . . . . .	30
15. IANA Considerations . . . . .	30
16. Changelog . . . . .	30
17. Acknowledgements . . . . .	31
18. Open Issues . . . . .	32
19. References . . . . .	33
19.1. Normative References . . . . .	33
19.2. Informative References . . . . .	33
Authors' Addresses . . . . .	35

## 1. Introduction

IPv6 Neighbor Discovery [RFC4861] was defined at a time when local area networks had different properties than today. A common link was the yellow-coax shared wire Ethernet, where a link-layer multicast and unicast worked the same - send the packet on the wire and the interested receivers will pick it up. Thus the network cost (ignoring any processing cost on the receivers that might not completely filter out Ethernet multicast addresses that they did not want) and the reliability of sending a link-layer unicast and multicast was the same. Furthermore, the hosts at the time was always on and connected. Powering on and off the workstation/PC hosts at the time was slow and disruptive process.

Under the above assumptions it was quite efficient to maintain the shared state of the link such as the prefixes and their lifetimes using periodic multicast Router Advertisement messages. It was also efficient to use multicast Neighbor Solicitations for address resolution as a slight improvement over the broadcast use in ARP. And finally, checking for a potential duplicate IPv6 address using broadcast was efficient and believed to be likely to be robust.

Since then we've seen a tremendous change with the wide-spread deployment of WiFi and other wireless network technologies. WiFi is a case in point in that it provides the same network service abstraction as Ethernet and is often bridged with Ethernets, meaning that the Neighbor Discovery software on hosts and routers might be unaware that WiFi is being used. Yet the performance and reliability of multicast is quite different than for unicast on WiFi (see for instance [I-D.vyncke-6man-mcast-not-efficient]). Similarly 3GPP and M2M networks and devices will benefit from a standard specification for optimized Neighbor discovery. Even wired networks have evolved substantially with modern switch fabrics using explicit packet replication logic to handle multicast packets.

The assumptions about the reliability of a single multicast message for duplicate address detection has also shown to be not correct, due to a set of issues listed in [I-D.yourtchenko-6man-dad-issues].

The hosts and usage patterns has undergone radical changes as well. Hosts go to sleep when not in use to save on battery power [RFC6574] or to be more energy efficient even with mains power. The expectation is that waking up doesn't take much time and power otherwise the benefits of sleeping are greatly reduced. Initially sleeping hosts were esoteric sensor nodes, but this sleeping hosts have become mainstream in smartphones.

Some of the above trends were observed early (e.g., Ohta-san

commented on Neighbor Discovery being inefficient on WiFi a long time back) but the issues were not broadly understood. The issues were raised in the 6LowPAN context where the desire was to run IPv6 over low-power radio networks and battery operated devices. That lead to defining a set of optimizations [RFC6775] for that specific category of links. However, the trends are not limited to such specific link types.

This document applies what we have learned from 6LowPAN to all link types. That includes reusing existing support from base Neighbor Discovery (such as Redirect messages) and reusing from 6LowPAN-ND (Address Registration Option). There are additions above and beyond that to improve the robustness with redundant routers and to support mixed mode.

The optimizations are done in a way to provide incremental benefits. As soon as there is at least one router on a link which supports these optimizations, then the updated hosts on the link can sleep better, while co-existing on the same link with unmodified hosts.

## 2. Goals and Requirements

The goal is to remove as much Neighbor Discovery multicast traffic on the link as possible, and handle Duplicate Address Detection (DAD) without requiring the hosts to always be awake. While not an explicit goal, it turned out that the issues in [I-D.yourtchenko-6man-dad-issues] that are about robustness/correctness are also addressed as a side effect of supporting sleepy hosts.

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

The requirements handle are:

Remove the use of multicast for DAD and Address Resolution (no multicast NS messages), and remove periodic multicast RAs. Some multicast RS and RA are needed to handle the arrival of new hosts and routers on the link since they need to bootstrap to find each other.

Remove the need for hosts to always be awake to defend their addresses by responding to any DAD probes.

Ensure that the protocol is robust against single points of failure and uses soft state which is automatically rebuilt after a state loss.

A router which does not support legacy hosts will always maintain a complete set of Neighbor Cache Entries (NCEs) for all hosts on the link. Hence there is no need for it to send Neighbor Solicitations. Thus it can avoid the problem specified in [RFC6583].

The optimized solution SHOULD be independent of the addresses allocation mechanism. In addition to supporting SLAAC [RFC4862] and DHCPv6 [RFC3315] it SHOULD also work with hosts with 'Privacy Extensions for Stateless Address Autoconfiguration in IPv6' [RFC4941] and with stable IPv6 private addresses [I-D.ietf-6man-stable-privacy-addresses] thus it handles the recommendations in [I-D.ietf-6man-default-iids].

### 2.1. Mixed-Mode Operations

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

The mixed-mode model SHOULD support arbitrary combinations of legacy [RFC4861] hosts and/or routers with new hosts and/or routers on a link. The introduction of one new router SHOULD provide improved services to a new host, allowing the new host to avoid multicast and not requiring the host to be awake to defend its IPv6 addresses using DAD.

This document assumes that an implementation will have configuration knobs to determine whether it is running in legacy IPv6 ND [RFC4861] or Efficiency Aware only mode (no-legacy mode) or both (Mixed mode).

### 3. Changes to ND state management

These optimizations change some fundamental aspects of Neighbor Discovery. Firstly, it moves the distributed address resolution state (each node responding to a multicast NS for its own addresses) to a set of routers which maintain a list of Address Registrations for the hosts. Secondly, the information distributed in Router Advertisements changes from being periodically flooded by the routers to explicit requests from the hosts for refreshed information using unicast Router Solicitations.

#### 4. Definition Of Terms

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

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

A router that 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):**

The EAH is the host which implements the host functionality for optimized Neighbor Discovery mentioned in this document. At least initially implementations are likely to have a fallback to legacy Neighbor Discovery when no NEAR is on the link.

**Legacy IPv6 Host:**

A IPv6 host that implements [RFC4861] without the extensions in this document.

**Legacy IPv6 Router:**

A IPv6 router that implements [RFC4861] without the extensions in this document.

**Mixed mode**

A NEAR supports both legacy hosts and EAH, with a configuration knob to disable the support for legacy hosts. Some routers on the link can be legacy and some can be NEAR.

**No-legacy mode**

A mode configured on a NEAR to not support any legacy [RFC4861] hosts or routers. Opposite of mixed mode.

**IPv6 Address Registrar**

Normally the default router(s) on the link will act as IPv6 Address Registrars tracking all the EAHs. But in some cases it is more efficient to use a different set of routers as Address Registrars. The hosts are informed of the address registrars using router advertisement messages, and register with the available registrars.

**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. The protocol supports EUI-64 for compatibility with [RFC6775].

#### DUID

It is a DHCP Unique ID of a device [RFC3315]. The DUID is assumed to be stable in a given IPv6 subnet. A device which does not have an EUI-64 can form and use a DUID in its address registrations.

#### NCE

Neighbor Cache Entry. It is a conceptual data structure introduced in section 5.1 of [RFC4861] and further elaborated in [RFC6775].

#### TID

The Transaction ID is a device generated sequence number used for registration. This number is used to allow a host to have concurrent registrations with different routers, while also being able to robustly replace a registration with a new one. It facilitates interoperability with protocols like RPL [RFC6550] which use a TID internally to handle host movement.

## 5. Protocol Overview

In a nutshell, the following basic optimizations are made from the original IPv6 Neighbor Discovery protocol [RFC4861]:

- o Adds Node Registration with the default router(s).
- o Address Resolution and DAD uses the registered addresses instead of multicast Neighbor Solicitation messages for non-link-local IPv6 addresses.
- o Does not require unsolicited periodic Router Advertisements.
- o Supports mixed-mode operation where legacy IPv6 hosts and routers and NEARs and EAHs can co-exist on the same link. This support can be configured off.

When a host powers on it behaves conforms to legacy ND [RFC4861] by multicasting up to MAX\_RTR\_SOLICITATIONS Router Solicitations and receives Router Advertisements. The additional information in the Router Advertisements by the NEARs is used by the EAH to build a list of IPv6 Address Registrars. As the host is forming its IPv6 addresses (using any of the many stateless and stateful IPv6 address allocation mechanism) then, instead of using a multicast DAD message, it unicasts an Neighbor Solicitation with the new Address Registration Option (ARO) to the Registrars. Assuming an IPv6

addresses are not duplicate the EAH receives a Neighbor Advertisement with the ARO option from the NEARs. The EAH refreshes the registered addresses before they expire, thereby removing the need for the EAH to be awake to defend its addresses using DAD as specified in [RFC4862] as the NEARs will handle DAD.

The routers on the link advertise the prefixes without setting the L flag. Thus only the IPv6 link-local addresses are considered on-link. Thus the hosts will send packets to a default router, and the default routers maintain all the registrations. Hence a router will know the link-layer addresses of all the registered hosts. This enables the router to forward the packet (without needing any Address Resolution with the multicast Neighbor Solicitation), and also to send a Redirect to the originating host informing the host of the link-layer address.

Instead of relying on periodic multicast RAs to refresh the lifetimes of prefixes etc., the hosts ask for refreshed information by unicasting a Router Solicitation before the information expires. Note that [I-D.nordmark-6man-rs-refresh] make that behavior more explicit by having the routers advertise a timeout.

The periodic multicast RAs may be used to provide new information such as additional prefixes, radical reduction in the preferred and/or valid lifetime for a prefix. A host does not know to ask for such information. Thus a router that wishes to quickly disseminate such change can resort to a few multicast RAs, or wait for the hosts to request a refresh using a Router Solicitation.

The routers can crash and loose all their state, including the Address Registrations. On router initialization the router will multicast a few initial RAs. The protocol has a Router Epoch mechanism which is used by the hosts to detect that the router has lost state. In that case the hosts will immediately re-register allowing the router to quickly rebuild its Address Registration state.

Normally it is sufficient for the hosts to register with all the default routers on the link. However, in some cases such as simplistic VRRP deployment the hosts should register with all the VRRP routers even though they only know of one virtual router IPv6 address. And in other cases it would be more efficient to only register with one router even though there are multiple default routers. The RAs can contain a Registrar Address Option to explicitly tell the hosts where to register.

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 before the registration lifetime expiration. The recommended procedure on wakeup is to unicast a Neighbor Solicitation to the default router(s), which serves as DNA check [RFC6059] that the host is on the same link, performs NUD against the router, and includes the Address Registration Option to refresh the registration.

#### 5.1. Proxying to handle Mixed mode

When there are one or more legacy routers on the link then the recommendation is to configure those to advertise the prefixes with L=0 just as the NEARs. That results in the hosts sending all packets to a default router unless/until they receive a Redirect. However, the legacy routers do not maintain the address registrations. Thus even though all the hosts send the packets to the routers, the legacy routers might in turn need to perform Address Resolution by multicasting a Neighbor Solicitation per [RFC4861]. In addition, legacy hosts and legacy routers will perform DAD as specified in [RFC4862] that is, by sending a multicast NS and waiting for a NS or NA which indicates a conflict. A EAH will not receive and respond to such messages.

If the NEARs have been configured to operate in mixed-mode, then they will respond to multicast NS messages from legacy nodes for both DAD and Address Resolution. They will respond with the Target Link Layer Address being that of the registered host, so that subsequent communication will not go via the routers. (Implementations of "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] might proxy using their own MAC address as TLLA, but that is outside of the scope of this document.)

### 6. New Neighbor Discovery Options and Messages

This specification introduces a new flag in the RAs, reuses and extends the ARO option from [RFC6775] and introduces a new Registrar Address option.

#### 6.1. Router Advertisement flag for NEARs

A new Router Advertisement flag is needed in order to distinguish a router advertisement sent by a NEAR, which will trigger an EAH to register with the NEAR. This flag is ignored by the legacy IPv6 hosts.

The current flags field in RA is reproduced here with the added 'E' bit.

```

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

```

The 'E' bit is set to 1 in a RA sent by a NEAR. In all other cases the E bit MUST be 0.

## 6.2. Address Registration Option (ARO)

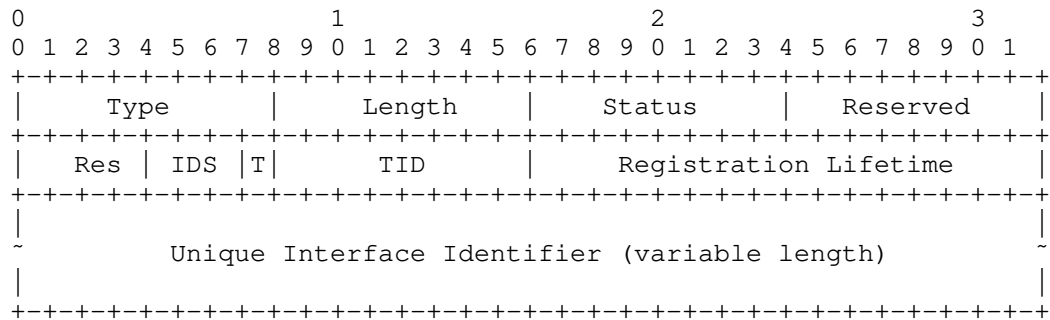
The Address Registration Option was defined in [RFC6775] for the purposes of 6LoWPAN and this document extends it in a backwards compatible way by using some of the reserved fields. The extensions are to handle different unique identifiers than EUI-64 (this document specifies how to use DHCP Unique Identifiers with the ability to use other identifier namespaces in the future) and a Transaction Id.

The Unique Interface Identifier is used by the NEARs to distinguish between a refresh of an existing registration and a different host trying to register an IPv6 address which is already registered by some other host. Thus the requirement is that the unique id is unique per link, but due to the potential for host mobility across links and subnets it should be globally unique. Both an EUI-64 and a DUID satisfies that requirement.

The TID is used by the NEARs to handle the case when due to packet loss one NEAR might have a old registration and another NEAR has a newer registration. The TID allows them to determine which is more recent. The TID also facilitates the interaction with RPL [RFC6550].

An Address Registration Option 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 the default router(s). 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.

When the ARO is sent by a host then, for links which have link-layer addresses, a SLLA option MUST be included. The address that is registered is the IPv6 source address of the Neighbor Solicitation message. Typically a host would have several addresses to register, with each one being registered using a separate NS containing an ARO. (This approach facilitates applying SeND [RFC3971].)



## Fields:

Type: 33 [RFC6775]

Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. The value 0 is invalid.

Status: 8-bit unsigned integer. Indicates the status of a registration in the NA response. MUST be set to 0 in NS messages. See [RFC6775].

Reserved: 8 bits. This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Res: 4 bits. This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

IDS: 3 bits. Identifier name Space. Indicates whether the Unique Interface Identifier is a DUID or or a IEEE assigned EUI-64 with room to define additional name spaces.

T bit: One bit flag. Set if the TID octet is valid.

TID: 8-bit integer. It is a transaction id maintained by the host and used by the NEARs to determine the most recent registration.

Registration Lifetime: 16-bit unsigned integer. The amount of time in a unit of 60 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option. A value of zero means to remove

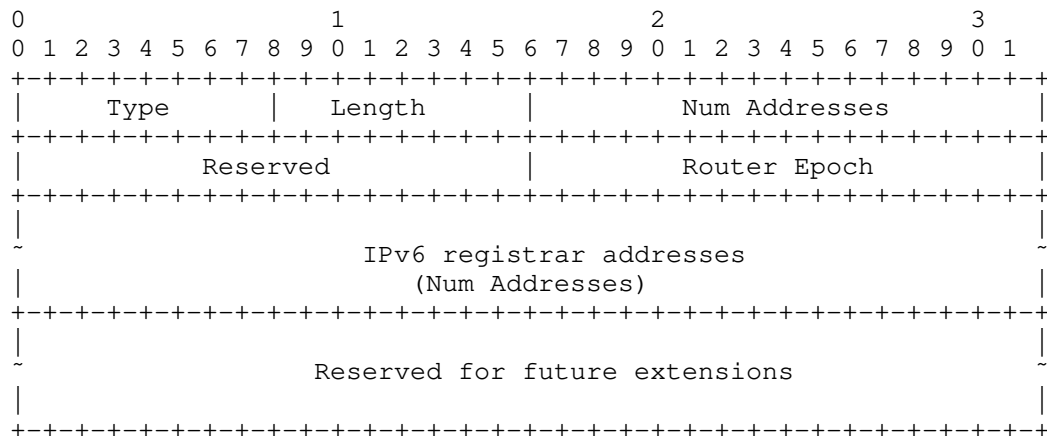
the registration.

Unique Interface Identifier: Variable length in multiples of 8 bytes. If the IDS=000, then it is an 8 byte (64 bit) unmodified EUI-64. If IDS=0011 then it is a variable length DUID. A DUID MUST be zero padded to a multiple of 8 bytes.

When a node includes ARO option in a Neighbor Solicitation it MUST, on links that have link-layer addresses, also include a SLLA option. That option is needed so that the registrar can record the link-layer address on success and send back an error if the address is a duplicate.

### 6.3. Registrar Address Option (RAO)

Normally the Registrars are the Default Routers. However, there are cases (like some approaches to handle VRRP, or coordinated separate routers) where there is a need to have different (and either more or less) Registrars than Default Routers. Furthermore, to robustly handle NEAR state state loss this option carries a Router Epoch which triggers the EAHs to re-register on a router epoch change. The RAO contains the information for both of those.



Fields:

Type: TBD (IANA)

Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. The value 0 is invalid.

Num Addresses 16-bit unsigned integer. Set to zero if there are no addresses i.e., when the option is used to only carry the router epoch. NumAddresses\*2 + 1 must not exceed the Length.

Reserved 16-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Router epoch 16-bit integer. A router MUST pick a new epoch after a state loss, either by keeping the epoch in stable storage and incrementing it, or picking a good random number.

IPv6 registrar addresses Zero or more IPv6 addresses, typically of link-local scope.

The receiver MUST silently ignore any data after the IPv6 registrar addresses field (such data is present when the Length is greater than NumAddresses\*2 + 1).

The Registrar Addresses are subject to the same lifetime as the Default Router Lifetime (thus there is no explicit lifetime field in the RAO).

## 7. Conceptual Data Structures

In addition to the Conceptual Data structures in [RFC4861] a EAH needs to maintain the new Registrar List for each interface. The Registrar List contains the set of IP addresses to which the host needs to send Address Registrations. Each IP address has a Router Epoch (used to determine when a router might have lost state). Also, the host MAY use this data structure to track when it needs to refresh its registrations with the registrar.

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 [RFC4861]. 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 [RFC4861] that allow for garbage collection when low on memory. Legacy entries are created only when there is no duplicate NCE. 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 [RFC6775].

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 [RFC4861]. There can only be one type of NCE for an IP address at a time.

## 8. Host Behavior

A EAH follows [RFC4861] and applicable parts of [RFC6775] as specified in this section./

A EAH implementation MAY be able to fall back to [RFC4861] host behavior if there is no NEAR on the link.

### 8.1. Host and/or Interface Initialization

A host multicasts Router Solicitation at system startup or interface initialization as specified in [RFC4861] and its updates such as [I-D.ietf-6man-resilient-rs]. If the interface initialization is due to potential host movement or a wakeup from sleep then the host initially sends a unicast Neighbor Solicitation to the default router(s).

Unlike RFC4861 the RS MUST (on link layers which have addresses) include a SLLA option, which is used by the router to unicast the RA.

The host is not required to join the solicited-node multicast address(es) but it MUST join the all-nodes multicast address.

### 8.2. Host Receiving Router Advertisements

The RA is validated and processed as specified in [RFC4861] with additional behavior for RAO and the Registrar List as follows.

When a RA is received without a RAO (but with the E flag set), or the RAO contains no Registrar Addresses, then the IPv6 source address is added/updated in the Registrar List. When a RA is received with a RAO then the IPv6 Registrar Addresses in that option are added/updated in the data structure.

If those Registrar List (or entries) already exist and the Router Epoch in the RAO differs from the Router Epoch in the Registrar List



entry, or if the entry does not exist, then the host will initiate sending NS messages with ARO options to the new/updated Registration List entries. Note that if the RA contains no RAO (but the E flag is set) then for the purposes of the epoch comparison one should use a zero Router Epoch.

However, if the Default Router Lifetime in the RA is zero, then any matching Registration List entry (or entries) are instead deleted from the Registration List. It is OPTIONAL for the host to de-register when an entry is deleted from the Registration List.

The host can form its IPv6 address using any available mechanism - SLAAC, DHCPv6, temporary addresses, etc - as the registration mechanism is orthogonal and independent of the address allocation. The Address Registration procedure replaces the DAD procedure in [RFC4862].

### 8.3. Timing out Registrar List entries

The lifetime for the Registrar List entries are taken from the Default Router Lifetime in the RA. When an entry is removed the host MAY send AROs with a zero Registration Lifetime to the removed Registrar Addresses.

### 8.4. Sending AROs

When a host has formed a new IPv6 address, or when the host learns of a new NEAR and has existing IPv6 addresses, then it would register the new things (could be new addresses to all the existing Registrars, or the all the IPv6 addresses with the new Registrar. IPv6 link-local addresses are registered as well as the global addresses and ULAs.

If the EAH has a TID then it sets the T-bit and includes the TID in the ARO. When the host registers its addresses with multiple Registrars it uses the same TID. However, if the host has moved (lost its network attachment and determines it is attached to a different link using e.g., DNA [RFC6059]), then it will increment the TID value and use the new value for subsequent registrations.

The host places its Unique Interface Identifier (whether it is a DUID or EUI-64) in the ARO. This identifier is typically kept in stable storage so that the host can use the same identifier over time. It MUST use the same identifier when it re-registers its address, since otherwise all those will be returned as duplicates.

The NS which includes the ARO option MUST include a SLLA option on link layers that have link-layer addresses.

The EAH retransmits NS messages with ARO as specified in [RFC6775] until it receives a NA message from the Registrar containing an ARO. The number of such retransmissions SHOULD be configurable.

#### 8.5. Receiving Neighbor Advertisements

The Neighbor Advertisement are validated and processed as specified in [RFC4861] for example to handle Neighbor Unreachability Detection (NUD). In addition, the host processes any received ARO as follows.

If the ARO has status code 0 (Success), then the host updates the information in the Registrar List to know when it next needs to refresh the registered address with this Registrar. No further processing is needed of the ARO.

If the ARO has status code 1 (Duplicate), then the host can not use the IPv6 address. The host follows the address allocation protocol it used to pick a new address - be that DHCPv6, temporary addresses, etc.

If the ARO has a status code 2 (Neighbor Cache Full) in response to its registration request from a Registrar, then the node SHOULD continue to register the address with other Registrars (when available).

TBD: What about other not yet defined status code values?

#### 8.6. Host Management of the TID

It is RECOMMENDED that the EAH MAY maintain a sequence counter (TID) in stable storage according to section 7 of [RFC6550]. The TID is used to resolve conflicts between different registrations issues by the same host for the same IPv6 address. Conflicts can be due to different link-layer addresses, but it can also be due to registering with different NEARs/Registrars and those routers connect use protocols like RPL for routing, and RPL uses a TID to handle movement vs. multi-homing.

Thus the same TID should be used if the host is registering its addresses with multiple Registrars at the same time. But if the host might have moved to a different attachment point, then it should increment its TID for subsequent registrations.

#### 8.7. Refreshing a 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.

This specification does not constrain the implementation. One possible implementation strategy is to attempt re-register at 1/3rd of the registration lifetime, and if no response try again at 2/3rd of the lifetime, etc. Another possible strategy is to wait until the end of the registration lifetime and then do the same relatively rapid retransmissions as for the initial registration [RFC6775]. In all cases the host SHOULD apply a random factor to its re-registration timeout to avoid self-synchronizing behavior across lots of hosts. Sleeping hosts would re-register when they are waking up to do other work.

#### 8.8. De-registering

If anytime, the node decides that it does not need a particular default router's service anymore, then it SHOULD send a de-registration message to that NEAR/Registrar. Similarly if the host stops using a particular IPv6 address, then it SHOULD de-register that address with all the Registrars it had registered with. This applies even if the host was using the IPv6 address, then went to sleep, and then picked a different set of IPv6 addresses. In such a case it is preferred if the host de-registers before going to sleep. A mobile host SHOULD first de-register its addresses before it moves away from the subnet (if the mobile host can know that in advance of moving.)

De-registration is performed by unicasting a Neighbor Solicitation with an ARO where the Registration Lifetime is set to zero. Such an ARO should have an incremented TID. De-registration AROs are retransmitted just like other AROs as specified above.

#### 8.9. Refreshing RA information

The EAH is responsible for asking the routers for updates to the information contained in the Router Advertisements, since the NEARs will not multicast such updates. That is done by sending unicast RSs to the router(s) which will result in unicast RAs. However, significant care is required in determining when the RSs should be transmitted.

As part of normal operation the Default Routers, Prefixes, and other RA information have lifetimes, and there are a few common cases:

1. The advertised lifetimes are constant i.e., the routers keep on advancing the time when the information will expire.
2. The routers decrement the advertised lifetimes in real time i.e., the information is set to expire at a determined time and subsequent RAs have lower and lower lifetimes.

3. The routers forcibly expire some information by advertising it with a zero lifetime for a while, and then stop advertising it.
4. A router crashes or is silently removed from the network and as a result there are no more updates. For example, that default router will expire and there is little benefit in trying to refresh it by sending lots of RSs.

The host's logic for refreshing the information needs to be careful to not send a large number of RSs, in particular if there is information that is supposed to expire at a fixed time i.e., the lifetime decrements in real time.

A host MUST NOT try to refresh information because its lifetime is near zero, since that would cause unnecessary RSs. Instead the refresh needs to be based on when the information was most recently received from the router. A lifetime of 10 minutes that was heard from the router 10 minutes ago might be normal as part of expiring some information. But a remaining lifetime of 10 minutes for a prefix that was last heard 24 hours ago with a lifetime of 24 hours means that a refresh is in order.

It is RECOMMENDED that the host track the expiry time (the wall clock time when some information will expire) and when it receives an RA with that information it SHOULD check whether the expiry time is moving forward, or appears to be frozen in time. That can tell the difference between the first two cases above, and avoid unnecessary RSs as some information naturally expires. Furthermore it is RECOMMENDED that the host track which information was received from which router, so that it can see when a router used to provide the information no longer provides it. That helps to see if the third case above might be in play. Finally, if a router has not responded to a few (e.g., MAX\_RTR\_SOLICITATIONS) attempts to get a refresh, then the router might be unreachable or dead, and there is little benefit in sending further RSs to that router. When the router comes back it will multicast a few RAs.

When the hosts determines that case 1 above is likely, then it should pick a reasonable time to ask for refreshes. The exact refresh behavior is not mandated by this specification, but the same implementation strategies as for refreshing address registrations in Section 8.7 can be considered.

A example simple implementation approach is to only base the refreshing on the default router lifetime (thus ignore prefix and other lifetime), and pick a refresh time which is 1/3 of the default router lifetime. If no RA is received, a subsequent refresh can be done at 2/3 of the default router lifetime. If that does not result

in a RA, then MAX\_INITIAL\_RTR\_ADVERTISEMENTS can be sent as the router lifetime is about to expire. Note that a default router lifetime of zero MUST NOT result in sending a RS refresh based on a timeout of zero.

If the host is unable to refresh the information and as a result ends up with an empty default router list, or all the default routers are marked as UNREACHABLE by NUD, then the host MAY switch to sending initial multicast Router Solicitations as in Section 8.1.

Note that [I-D.nordmark-6man-rs-refresh] make that behavior more explicit by having the routers advertise a timeout.

#### 8.10. Sleep and Wakeup

The protocol 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 related 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.

When a host wakes up it can combine movement detecting (DNA), NUD, and refreshing its Address Registration by sending a unicast NS with an ARO to its existing default router(s).

#### 8.11. Receiving Redirects

An EAH handles Redirect messages as specified in [RFC4861].

#### 8.12. Movement Detection

When a host moves from one subnet to another its IPv6 prefix changes and the movement detection is determined according to the existing IPv6 movement detection described in [RFC6059].

### 9. Router Behavior

A NEAR follows [RFC4861] and applicable parts of [RFC6775] as follows in this section.

A NEAR SHOULD support legacy hosts and mixed mode as specified in this section by being able to proxy Address Resolution and DAD. The NEAR SHOULD implement a knob to be able to disable this behavior. That knob can either be set to "mixed-mode" or to "no-legacy-mode".

The RECOMMENDED default mode of operation for the NEAR is Mixed-mode.

A NEAR should be configured to advertise prefixes without the on-link (L-bit) unset. Furthermore, any legacy routers attached to the same link as a NEAR should be configured the same way. That reduces the cases in mixed mode when multicast NS messages are needed between legacy hosts and routers.

#### 9.1. Router and/or Interface Initialization

A NEAR multicasts some initial Router Advertisements (MAX\_INITIAL\_RTR\_ADVERTISEMENTS) at system startup or interface initialization as specified in [RFC4861] and its updates.

The NEAR MUST join the all-nodes and all-routers multicast addresses. In mixed mode it MUST also join the solicited-node multicast address(es) for its addresses and also for all the Registered NCEs.

A NEAR picks a new Router Epoch if it has lost the Registered NCEs, which is typically the case for router initialization. Either the Router Epoch can be stored in stable storage and incremented on each router initialization, or the NEAR can pick a good random number on router initialization. The effect of occasionally picking the same Router Epoch as before the state loss is that it will take longer for the router to build up its state of Registered NCEs.

#### 9.2. Receiving Router Solicitations

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 [RFC4861]). An RA MUST carry any 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 which the hosts use to distinguish a NEAR from a legacy router.

Unlike [RFC4861] which suggests multicast Router Advertisements, this specification optimizes the exchange by always unicasting RAs in response to RSs. This is possible since the RS always includes a SLLA option, which is used by the router to unicast the RA.

If the NEAR has been configured to send an explicit set of IPv6 Registrar Addresses, or implements a Router Epoch, then the NEAR includes a RAO in all its RAs.

### 9.3. Periodic Multicast RA for legacy hosts

The NEAR MUST NOT send periodic RA in no-legacy mode. In mixed mode the NEAR needs to send periodic multicast RAs as specified in [RFC4861] to support legacy hosts.

### 9.4. Multicast RA when new information

When a router has new information to share (new prefixes, prefixes that should be immediately deprecated, etc) it MAY multicast up to MAX\_INITIAL\_RTR\_ADVERTISEMENTS number of Router Advertisements. Note that such new information is not likely to reach sleeping hosts until those hosts refresh by sending a RS.

### 9.5. Receiving ARO

The NEAR follows the logic in [RFC6775] for managing the NCEs and responding to NS messages with the ARO option. The slight modification is that instead of using an EUI-64 as the key to check for duplicates, the NEAR uses the IDS value plus the variable length Unique Interface Identifier value as the key. In addition the NEAR checks the new TID field as follows.

The TID field is used together with age of a registration for arbitration between two routers to ensure freshness of the registration of a given target address. Same value of TID indicates that both states of registration are valid. In case of a mismatch between comparable TIDs, the most recent TID wins. The TIDs are compared as specified in section 7 of [RFC6550].

### 9.6. NCE Management in NEARs

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.

A Router Solicitation might be received from a host that has not yet registered its address with the router or from a legacy [RFC4861] host in the Mixed-mode operation.

The router MUST NOT modify an existing Registered 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 from a EAH. 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 or 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 requester 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 Lifetime and IDS plus Unique Interface Identifier are recorded in the Neighbor Cache Entry along with the existing information described in [RFC4861]. The registered NCE are meant to be ready and reachable for communication and no address resolution is required in the link. An EAH will renew its registration to Registered NCE at the router. However the router may perform NUD towards the EAH hosts as per [RFC4861]. Should NUD fail the NEAR MUST NOT remove the Registered NCE. Instead it marks it as UNREACHABLE.

#### 9.7. Sending non-zero status in ARO

If the Registration fails (due to it being a duplicate or the Neighbor Cache being full), then the NEAR will send an NA with ARO having a non-zero status. However, it needs to send that back to the originator of the failing ARO, and that host and link-layer address will not be present in the Neighbor Cache.

The NEAR forms a NA with ARO, and then forms the link-layer address by using the content of the SLLA option in the NS, bypassing the Neighbor Cache to send this error.

#### 9.8. Timing out Registered NCEs

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.



Similarly, if Neighbor Unreachability Detection on the router determines that the host is UNREACHABLE (based on the logic in [RFC4861]), 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 UNREACHABLE state, that NCE should be marked as STALE.

The NEAR router SHOULD deny registration to a new registration request with the status code 2 when it reaches the maximum capacity for handling the neighbor cache.

#### 9.9. Router Advertisement Consistency

The NEAR follows section 6.2.7 in [RFC4861] by receiving RAs from other routers (NEAR and legacy) on the link. In addition to the checks in that section it verifies that the prefixes do not have the L flag set, and that the Registrar Address options are consistent. Two RAOs are inconsistent if they contain the have a different Router Epoch and have some IPv6 Registration Addresses in common.

#### 9.10. Sending Redirects

A NEAR sends redirects (with target=destination) to inform the hosts of the link-layer address of the nodes on the link.

This can be disabled on specific link types for instance, radio link technologies with hidden terminal issues.

#### 9.11. Providing Information to Routing Protocols

If there is a routing protocols like RPL which wants visibility into the location of each IPv6 address, then this can be retrieved from the Registered NCEs on the NEAR.

#### 9.12. Creating Legacy NCEs

In mixed-mode a NEAR will create Legacy NCEs when receiving RA, RS, and NS messages based on the source of those packets. When not in mixed-mode it needs to create Legacy NCEs for other routers by looking at those packets.

#### 9.13. Proxy Address Resolution and DAD for Legacy Hosts

This section applies in mixed mode. It does not apply in no-legacy mode.

A NEAR in mixed mode MUST join all solicited-node for all Registered NCEs.

The NEAR SHOULD 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. This form of proxying is to respond with a NA that has a TLLAO taken from the Registered NCE for the target. Thus it is unlike ND Proxy as specified in [RFC4389]. (Implementations of "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] might proxy using their own MAC address as TLLA, but that is outside of the scope of this document.)

In the context of this specification, proxy means:

- o Responding to DAD probes for a registered NCE. A DAD probe from a legacy host would not contain any ARO, hence the NEAR will assume it is always a duplicate if the IPv6 target address has a Registered NCE.
- o Defending a registered address using NA messages with and ARO option and the Override bit set if the ARO option in the NS indicates either a different node (different IDS+Unique Interface Id) or a older registration (when comparing the TID).
- o Advertising a registered address using NA messages, asynchronously or as a response to a Neighbor Solicitation messages.
- o Looking up a destination on the link using Neighbor Solicitation messages in order to deliver packets arriving for the EAH.

The NEAR SHOULD also support DAD from a EAH for IPv6 address that might be in use by a legacy node. Thus when a NEAR in mixed-mode received an ARO for a new address it SHOULD perform DAD as specified in [RFC4862] by sending a multicast DAD message. Once that times out the NEAR can respond to the ARO. If a legacy host responds to the DAD probe, then the NEAR will respond to the ARO with Status=1 (Duplicate Address).

#### 10. 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 an 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(s).
- o Having the hosts send their packets via the default router(s).
- 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 On-link IPv6-destinations on a particular link must be registered else these packets are not resolved and extra NCEs are not created

In order to get maximal benefits from the ND-DoS protection from Address Registrations, the hosts and routers on the link need to be upgraded to NEARs and EAHs, respectively. With some legacy hosts the routers will still need to create INCOMPLETE NCEs and send NSs, which keeps the DoS opportunity open. However, with fewer legacy hosts the lower rate limits can be applied on creation of INCOMPLETE NCEs.

## 11. Bootstrapping

The bootstrapping mechanism described here is applicable for the efficiency-aware hosts and routers. At the start, the host uses its link-local address to send Router Solicitation and then it sends the Address Registration Option as described in this document in order to verify the IPv6 address. Note that on wakeup from sleep and after potential movement to a different link the host initially sends a unicast Neighbor Solicitation to the default router(s).

The following step 3 and 4 SHOULD be repeated for all the IPv6 addresses that are used for communications.

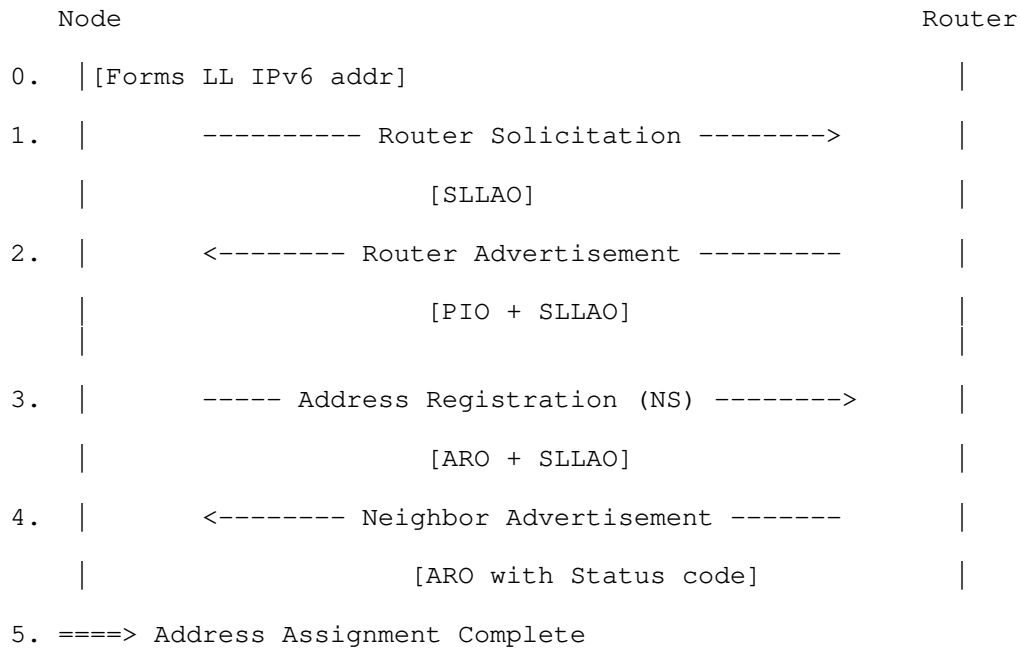


Figure 1: Neighbor Discovery Address Registration and bootstrapping

## 12. Interaction with other protocols

### 12.1. Detecting Network Attachment (DNA)

IPv6 DNA [RFC6059] uses unicast NS probes and link-layer indications to detect movement of its network attachments. That is consistent with the mechanism in this specification to unicast a NS when a host wakes up - this document recommends adding the ARO to that NS message.

Thus the ND optimization solution will work seamlessly with DNA implementations and no change is required in DNA solution because of Neighbor Discovery updates. It is a deployment and configuration consideration as to run the network in mixed mode or efficient-mode.

### 12.2. DHCPv6 Interaction

The protocol mechanisms in this document are orthogonal to the address assignment mechanism in use. If DHCPv6 is used for address assignment by an EAH then, if there are one or more NEARs on the subnet, the EAH will replace the DAD check specified in [RFC3315]

with Address Registration as specified in this document.

### 12.3. Other use of Multicast

Although the solution described in this document prevents unnecessary multicast messages in the IPv6 ND procedure, it does not affect normal IPv6 multicast packets nor the ability of nodes to join and leave the multicast group or forwarding multicast traffic or responding to multicast queries.

### 12.4. VRRP Interaction

A VRRP set of routers can operate with efficient-nd in two different ways:

- o Provide the illusion to hosts that they are a single router for the purposes of registrations. No RAO is needed in that case. But the pair needs some mechanism to synchronize their neighbor caches. Such a mechanism is out of scope of this document.
- o Have the hosts register with each router independently. In that case the VRRP router includes the RAO with the individual IP addresses of the routers in the pair. No synchronization of the neighbor caches are needed in that case.

## 13. Updated Neighbor Discovery Constants

This section discusses the updated default values of ND constants based on [RFC4861] section 10. New and changed constants are listed only for efficiency-aware-nd implementation. These values SHOULD be configurable and tunable to fit implementations and deployment.

#### 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
-------------------------------------	------------

Also refer to [RFC6583] , [RFC7048] and [RFC6775] for further tuning of ND constants.

#### 14. Security Considerations

These optimizations are not known to introduce any new threats against Neighbor Discovery beyond what is already documented for IPv6 [RFC3756].

Section 11.2 of [RFC4861] applies to this document as well.

This mechanism minimizes the possibility of ND /64 DoS attacks in efficiency-aware mode. See Section 10.

The mechanisms in this document work with SeND [RFC3971] in the no-legacy mode. In the mixed mode operation when a NEAR needs to respond to a legacy host sending a NS for a EAH, then SeND would not automatically fit. Potentially proxy SeND [RFC6496] could be used, but that would require explicit awareness and setup between the proxy and the proxied EAHs which seems impractical.

The mechanisms in this specification are orthogonal to the address allocation thus works as well with SLAAC and DHCPv6 as the various privacy-enhanced address allocation specifications. In particular, using an EUI-64 for the Unique Interface Identifier in this protocol does not require or assume that the IPv6 addresses will be formed using the modified EUI-64 format.

The mechanism uses a Unique Interface Identifier for the purposes of telling apart a re-registration from the same host and a duplicate/conflicting registration from a different host. That unique ID is not globally visible. Currently the protocol supports DHCPv6 DUID and EUI-64 format for this I-D, but other formats which facilitate non-linkability (such as strong random numbers large enough to be unlikely to cause collisions) can be defined.

#### 15. 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.

This document needs a new Neighbor Discovery option type for the RAO.

#### 16. Changelog

Changes from draft-chakrabarti-nordmark-energy-aware-nd-06:

- o Added references to dad-issues and rs-refresh.

Changes from draft-chakrabarti-nordmark-energy-aware-nd-05:

- o Fixed typos.
- o Clarified that on interface initialization after sleep or potential movement the host unicasts a NS to the default router(s).
- o Simplified the example timer handling for refreshing RA information.
- o Added handling of DAD from EAH to legacy node that was included in -04 and lost in the -05 edits.

Changes from draft-chakrabarti-nordmark-energy-aware-nd-04:

- o Significantly simplified the description of the protocol.
- o Added clarification on problem statement
- o Clarified that privacy and temporary addresses will be supported
- o Added an IDS field in the ARO to allow a DHCP Unique ID (DUID) as an alternative to EUI-64, with room to define other (pseudo) unique identifiers.
- o Allowed router redirects for NEAR.
- o Addressed some of comments made in the 6man list.
- o Added RAO to handle VRRP and similar cases when the default router list and registrar list needs to be different.
- o Added Router Epoch to cause re-registration on NEAR state loss.
- o Specified considerations for when to refresh address registrations.
- o Specified considerations for when to refresh RA information.

## 17. Acknowledgements

The primary idea of this document are from 6LoWPAN Neighbor Discovery document [RFC6775] 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 [RFC6775].

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 discussion in this document.

The authors like to thank Dave Thaler, Stuart Cheshire, Jari Arkko, Ylva Jading, Niklas J. Johnsson, Reda Nedjar, Purvi Shah, Jaume Rius Riu, Fredrik Garneij, Andrew Yourtchenko, Jouni Korhonen, Suresh Krishnan, Brian Haberman, Anders Brandt, Mark Smith, Lorenzo Colitti, David Miles, Eric Vyncke, Mark ZZZ Smith, Mikael Abrahamsson, Eric Levy-Abignoli, and Carsten Bormann for their useful comments and suggestions on this work.

## 18. Open Issues

The known open issues are:

- o IPv6 link-local addresses are always on-link and in this version of the document that results in multicast NS messages. The technique used in 6LowPAN-ND is too restrictive (extract the link-layer address from the IID). Should we send link-locals to routers and depend on Redirect?
- o If the Router Epoch is critical then we will see a RAO in all the RAs sent by NEARs. In such a case we don't need the E-bit in the RA.
- o Editorial: Add Comparison with 6lowpan-nd and 4861?
- o Editorial: Verify and update the description in this document to make it complete removing the need to read 6LowPAN-ND.
- o When a router has new information for the RA, currently it takes a while to disseminate that to sleeping nodes as this depends on when the hosts send a RS. We could potentially improve this is we could have an "information epoch number" in the ARO sent in the NA. But that only helps if the registrations are refreshed more frequently than the RA information.
- o Future? Currently if a router changes its information, a sleeping host would not find out when it wakes up and sends the NS with ARO. That could be improved if we fit the Router Epoch in NA/ARO.



But there is no room for 16 bits.

- o A separate but related problem is with unused NCEs due to frequent IPv6 address change e.g., hosts which pick a different set of addresses each time they wake up. This document recommends that they be de-registered by the host. That could be made simpler by introducing some Host Epoch counter in the NS/ARO.

## 19. References

### 19.1. Normative References

- [I-D.ietf-6man-resilient-rs]  
Krishnan, S., Anipko, D., and D. Thaler, "Packet loss resiliency for Router Solicitations", draft-ietf-6man-resilient-rs-04 (work in progress), October 2014.
- [I-D.nordmark-6man-rs-refresh]  
Nordmark, E., Yourtchenko, A., and S. Krishnan, "IPv6 Neighbor Discovery Optional Unicast RS/RA Refresh", draft-nordmark-6man-rs-refresh-01 (work in progress), October 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

### 19.2. Informative References

- [I-D.ietf-6man-default-iids]  
Gont, F., Cooper, A., Thaler, D., and W. Will, "Recommendation on Stable IPv6 Interface Identifiers", draft-ietf-6man-default-iids-02 (work in progress), January 2015.
- [I-D.ietf-6man-stable-privacy-addresses]  
Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address

Autoconfiguration (SLAAC)",  
draft-ietf-6man-stable-privacy-addresses-17 (work in  
progress), January 2014.

[I-D.vyncke-6man-mcast-not-efficient]

Vyncke, E., Thubert, P., Levy-Abegnoli, E., and A.  
Yourtchenko, "Why Network-Layer Multicast is Not Always  
Efficient At Datalink Layer",  
draft-vyncke-6man-mcast-not-efficient-01 (work in  
progress), February 2014.

[I-D.yourtchenko-6man-dad-issues]

Yourtchenko, A., "A survey of issues related to IPv6  
Duplicate Address Detection",  
draft-yourtchenko-6man-dad-issues-00 (work in progress),  
October 2014.

[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.

[RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor  
Discovery (ND) Trust Models and Threats", RFC 3756,  
May 2004.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure  
Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery  
Proxies (ND Proxy)", RFC 4389, April 2006.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless  
Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy  
Extensions for Stateless Address Autoconfiguration in  
IPv6", RFC 4941, September 2007.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for  
Detecting Network Attachment in IPv6", RFC 6059,  
November 2010.

[RFC6496] Krishnan, S., Laganier, J., Bonola, M., and A. Garcia-  
Martinez, "Secure Proxy ND Support for SEcure Neighbor  
Discovery (SEND)", RFC 6496, February 2012.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R.,  
Levis, P., Pister, K., Struik, R., Vasseur, JP., and R.

Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

[RFC6574] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, April 2012.

[RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, March 2012.

[RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, January 2014.

#### Authors' Addresses

Samita Chakrabarti  
Ericsson  
San Jose, CA  
USA

Email: samita.chakrabarti@ericsson.com

Erik Nordmark  
Arista Networks  
Santa Clara, CA  
USA

Email: nordmark@arista.com

Pascal Thubert  
Cisco Systems

Email: pthubert@cisco.com

Margaret Wasserman  
Painless Security

Email: mrw@painless-security.com



IPv6 maintenance Working Group (6man)  
Internet-Draft  
Updates: 2464, 2467, 2470, 4291 (if approved)  
Intended status: Standards Track  
Expires: April 25, 2014

F. Gont  
SI6 Networks / UTN-FRH  
A. Cooper  
CDT  
D. Thaler  
Microsoft  
W. Liu  
Huawei Technologies  
October 22, 2013

Deprecating EUI-64 Based IPv6 Addresses  
draft-gont-6man-deprecate-eui64-based-addresses-00

Abstract

Stateless Address Autoconfiguration (SLAAC) for IPv6 typically results in hosts configuring one or more stable addresses composed of a network prefix advertised by a local router, and an Interface Identifier that typically embeds a hardware address (e.g., an IEEE LAN MAC address). The security and privacy implications of embedding hardware addresses in the Interface Identifier have been known and understood for some time now, and some popular IPv6 implementations have already deviated from such scheme to mitigate these issues. This document deprecates the use of hardware addresses in IPv6 Interface Identifiers, and recommends the use of an alternative scheme ([I-D.ietf-6man-stable-privacy-addresses]) for the generation of IPv6 stable addresses.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 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.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Generation of IPv6 Interface Identifiers . . . . .	3
4. IANA Considerations . . . . .	3
5. Security Considerations . . . . .	4
6. Acknowledgements . . . . .	4
7. References . . . . .	4
7.1. Normative References . . . . .	4
7.2. Informative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

[RFC4862] specifies Stateless Address Autoconfiguration (SLAAC) for IPv6 [RFC2460], which typically results in hosts configuring one or more "stable" addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) [RFC4291] that typically embeds a hardware address (e.g., an IEEE LAN MAC address).

The security and privacy implications of embedding a hardware address in an IPv6 Interface ID have been known for some time now, and are discussed in great detail in

[I-D.ietf-6man-ipv6-address-generation-privacy]; they include:

- o Network activity correlation
- o Location tracking
- o Address scanning
- o Device-specific vulnerability exploitation

Some popular IPv6 implementations have already deviated from the traditional IID generation scheme to mitigate the aforementioned security and privacy implications [Microsoft].

As a result of the aforementioned issues, this document deprecates the use of hardware addresses in Interface Identifiers, and recommends the implementation of an alternative scheme ([I-D.ietf-6man-stable-privacy-addresses]) that mitigates most of the aforementioned issues.

NOTE: [RFC4291] defines the "Modified EUI-64 format" (which this document does not deprecate) for Interface identifiers. Appendix A of [RFC4291] then describes how to transform an IEEE EUI-64 identifier, or an IEEE 802 48-bit MAC address from which an EUI-64 identifier is derived, into an interface identifier in the Modified EUI-64 format. Deriving an IPv6 interface identifier based on an IEEE EUI-64 identifier is what is deprecated in this document. Other ways of generating an interface identifier in the Modified EUI-64 format are unaffected.

## 2. Terminology

### Stable address:

An address that does not vary over time within the same network (as defined in [I-D.ietf-6man-ipv6-address-generation-privacy]).

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. Generation of IPv6 Interface Identifiers

Nodes MUST NOT employ IPv6 address generation schemes that embed the underlying hardware address in the Interface Identifier. Namely, nodes MUST NOT generate Interface Identifiers with the schemes specified in [RFC2464], [RFC2467], and [RFC2470].

Nodes SHOULD implement and employ [I-D.ietf-6man-stable-privacy-addresses] as the default scheme for generating stable IPv6 addresses with SLAAC.

## 4. IANA Considerations

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

## 5. Security Considerations

This document deprecates the use of hardware addresses in IPv6 Interface Identifiers, and recommends an alternative scheme for generating IPv6 addresses with SLAAC such that a number of security and privacy issues are mitigated.

## 6. Acknowledgements

The authors would like to thank [TBD] for providing valuable comments on earlier versions of this document.

Fernando Gont would like to thank Ray Hunter for providing valuable input on this topic.

## 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.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC2467] Crawford, M., "Transmission of IPv6 Packets over FDDI Networks", RFC 2467, December 1998.
- [RFC2470] Crawford, M., Narten, T., and S. Thomas, "Transmission of IPv6 Packets over Token Ring Networks", RFC 2470, December 1998.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [I-D.ietf-6man-stable-privacy-addresses]  
Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", draft-ietf-6man-stable-privacy-addresses-14 (work in progress), October 2013.



## 7.2. Informative References

[I-D.ietf-6man-ipv6-address-generation-privacy]

Cooper, A., Gont, F., and D. Thaler, "Privacy Considerations for IPv6 Address Generation Mechanisms", draft-ietf-6man-ipv6-address-generation-privacy-00 (work in progress), October 2013.

[Microsoft]

Davies, J., "Understanding IPv6, 3rd. ed", page 83, Microsoft Press, 2012, <<http://it-ebooks.info/book/1022/>>.

## Authors' Addresses

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

Phone: +54 11 4650 8472  
Email: [fgont@si6networks.com](mailto:fgont@si6networks.com)  
URI: <http://www.si6networks.com>

Alissa Cooper  
CDT  
1634 Eye St. NW, Suite 1100  
Washington, DC 20006  
US

Phone: +1-202-637-9800  
Email: [acooper@cdt.org](mailto:acooper@cdt.org)  
URI: <http://www.cdt.org/>

Dave Thaler  
Microsoft  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

Phone: +1 425 703 8835  
Email: [dthaler@microsoft.com](mailto:dthaler@microsoft.com)

Will Liu  
Huawei Technologies  
Bantian, Longgang District  
Shenzhen 518129  
P.R. China  
  
Email: liushucheng@huawei.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 26, 2016

A. Cooper  
Cisco  
F. Gont  
Huawei Technologies  
D. Thaler  
Microsoft  
September 23, 2015

Privacy Considerations for IPv6 Address Generation Mechanisms  
draft-ietf-6man-ipv6-address-generation-privacy-08.txt

## Abstract

This document discusses privacy and security considerations for several IPv6 address generation mechanisms, both standardized and non-standardized. It evaluates how different mechanisms mitigate different threats and the trade-offs that implementors, developers, and users face in choosing different addresses or address generation mechanisms.

## 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 26, 2016.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	4
3. Weaknesses in IEEE-identifier-based IIDs . . . . .	4
3.1. Correlation of activities over time . . . . .	5
3.2. Location tracking . . . . .	6
3.3. Address scanning . . . . .	6
3.4. Device-specific vulnerability exploitation . . . . .	7
4. Privacy and security properties of address generation mechanisms . . . . .	7
4.1. IEEE-identifier-based IIDs . . . . .	9
4.2. Static, manually configured IIDs . . . . .	10
4.3. Constant, semantically opaque IIDs . . . . .	10
4.4. Cryptographically generated IIDs . . . . .	10
4.5. Stable, semantically opaque IIDs . . . . .	10
4.6. Temporary IIDs . . . . .	11
4.7. DHCPv6 generation of IIDs . . . . .	12
4.8. Transition/co-existence technologies . . . . .	12
5. Miscellaneous Issues with IPv6 addressing . . . . .	13
5.1. Network Operation . . . . .	13
5.2. Compliance . . . . .	13
5.3. Intellectual Property Rights (IPRs) . . . . .	13
6. Security Considerations . . . . .	13
7. IANA Considerations . . . . .	13
8. Acknowledgements . . . . .	14
9. References . . . . .	14
9.1. Normative References . . . . .	14
9.2. Informative References . . . . .	15
Authors' Addresses . . . . .	17

## 1. Introduction

IPv6 was designed to improve upon IPv4 in many respects, and mechanisms for address assignment were one such area for improvement. In addition to static address assignment and DHCP, stateless autoconfiguration was developed as a less intensive, fate-shared means of performing address assignment. With stateless autoconfiguration, routers advertise on-link prefixes and hosts generate their own interface identifiers (IIDs) to complete their addresses. [RFC7136] clarifies that the IID should be treated as an opaque value, while [RFC7421] provides an analysis of the 64-bit boundary in IPv6 addressing (e.g. the implications of the IID length

on security and privacy). Over the years, many interface identifier generation techniques have been defined, both standardized and non-standardized:

- o Manual configuration
  - \* IPv4 address
  - \* Service port
  - \* Wordy
  - \* Low-byte
- o Stateless Address Auto-Configuration (SLAAC)
  - \* IEEE 802 48-bit MAC or IEEE EUI-64 identifier [RFC2464]
  - \* Cryptographically generated [RFC3972]
  - \* Temporary (also known as "privacy addresses") [RFC4941]
  - \* Constant, semantically opaque (also known as random) [Microsoft]
  - \* Stable, semantically opaque [RFC7217]
- o DHCPv6-based [RFC3315]
- o Specified by transition/co-existence technologies
  - \* Derived from an IPv4 address (e.g., [RFC5214], [RFC6052])
  - \* Derived from an IPv4 address and port set ID (e.g., [RFC7596], [RFC7597], [RFC7599])
  - \* Derived from an IPv4 address and port (e.g., [RFC4380])

Deriving the IID from a globally unique IEEE identifier [RFC2464] [RFC4862] was one of the earliest mechanisms developed (and originally specified in [RFC1971] and [RFC1972]). A number of privacy and security issues related to the IIDs derived from IEEE identifiers were discovered after their standardization, and many of the mechanisms developed later aimed to mitigate some or all of these weaknesses. This document identifies four types of threats against IEEE-identifier-based IIDs, and discusses how other existing techniques for generating IIDs do or do not mitigate those threats.

## 2. Terminology

This section clarifies the terminology used throughout this document.

### Public address:

An address that has been published in a directory or other public location, such as the DNS, a SIP proxy [RFC3261], an application-specific DHT, or a publicly available URI. A host's public addresses are intended to be discoverable by third parties.

### Stable address:

An address that does not vary over time within the same IPv6 link. Note that [RFC4941] refers to these as "public" addresses, but "stable" is used here for reasons explained in Section 4.

### Temporary address:

An address that varies over time within the same IPv6 link.

### Constant IID:

An IPv6 interface identifier that is globally stable. That is, the Interface ID will remain constant even if the node moves from one IPv6 link to another.

### Stable IID:

An IPv6 interface identifier that is stable within some specified context. For example, an Interface ID can be globally stable (constant), or could be stable per IPv6 link (meaning that the Interface ID will remain unchanged as long as the node stays on the same IPv6 link, but may change when the node moves from one IPv6 link to another).

### Temporary IID:

An IPv6 interface identifier that varies over time.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words take their normative meanings only when they are presented in ALL UPPERCASE.

## 3. Weaknesses in IEEE-identifier-based IIDs

There are a number of privacy and security implications that exist for hosts that use IEEE-identifier-based IIDs. This section discusses four generic attack types: correlation of activities over time, location tracking, address scanning, and device-specific vulnerability exploitation. The first three of these rely on the attacker first gaining knowledge of the IID of the target host. This

could be achieved by a number of different entities: the operator of a server to which the host connects, such as a web server or a peer-to-peer server; an entity that connects to the same IPv6 link as the target (such as a conference network or any public network); a passive observer of traffic that the host broadcasts; or an entity that is on-path to the destinations with which the host communicates, such as a network operator.

### 3.1. Correlation of activities over time

As with other identifiers, an IPv6 address can be used to correlate the activities of a host for at least as long as the lifetime of the address. The correlation made possible by IEEE-identifier-based IIDs is of particular concern since they last roughly for the lifetime of a device's network interface, allowing correlation on the order of years.

As [RFC4941] explains,

"[t]he use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Anytime the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. ... The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear."

IP addresses are just one example of information that can be used to correlate activities over time. DNS names, cookies [RFC6265], browser fingerprints [Panopticlick], and application-layer usernames can all be used to link a host's activities together. Although IEEE-identifier-based IIDs are likely to last at least as long or longer than these other identifiers, IIDs generated in other ways may have shorter or longer lifetimes than these identifiers depending on how they are generated. Therefore, the extent to which a host's activities can be correlated depends on whether the host uses multiple identifiers together and the lifetimes of all of those identifiers. Frequently refreshing an IPv6 address may not mitigate correlation if an attacker has access to other longer lived identifiers for a particular host. This is an important caveat to keep in mind throughout the discussion of correlation in this document. For further discussion of correlation, see Section 5.2.1 of [RFC6973].

As noted in [RFC4941], in some cases correlation is just as feasible for a host using an IPv4 address as for a host using an IEEE identifier to generate its IID in its IPv6 address. Hosts that use static IPv4 addressing or who are consistently allocated the same address via DHCPv4 can be tracked as described above. However, the widespread use of both NAT and DHCPv4 implementations that assign the same host a different address upon lease expiration mitigates this threat in the IPv4 case as compared to the IEEE identifier case in IPv6.

### 3.2. Location tracking

Because the IPv6 address structure is divided between a topological portion and an interface identifier portion, an interface identifier that remains constant when a host connects to different IPv6 links (as an IEEE-identifier-based IID does) provides a way for observers to track the movements of that host. In a passive attack on a mobile host, a server that receives connections from the same host over time would be able to determine the host's movements as its prefix changes.

Active attacks are also possible. An attacker that first learns the host's interface identifier by being connected to the same IPv6 link, running a server that the host connects to, or being on-path to the host's communications could subsequently probe other networks for the presence of the same interface identifier by sending a probe packet (ICMPv6 Echo Request, or any other probe packet). Even if the host does not respond, the first hop router will usually respond with an ICMP Destination Unreachable/Address Unreachable (type 1, code 3) when the host is not present, and be silent when the host is present.

Location tracking based on IP address is generally not possible in IPv4 since hosts get assigned wholly new addresses when they change networks.

### 3.3. Address scanning

The structure of IEEE-based identifiers used for address generation can be leveraged by an attacker to reduce the target search space [I-D.ietf-opsec-ipv6-host-scanning]. The 24-bit Organizationally Unique Identifier (OUI) of MAC addresses, together with the fixed value (0xff, 0xfe) used to form a Modified EUI-64 interface identifier, greatly help to reduce the search space, making it easier for an attacker to scan for individual addresses using widely-known popular OUIs. This erases much of the protection against address scanning that the larger IPv6 address space could provide as compared to IPv4.



### 3.4. Device-specific vulnerability exploitation

IPv6 addresses that embed IEEE identifiers leak information about the device (Network Interface Card vendor, or even Operating System and/or software type), which could be leveraged by an attacker with knowledge of device/software-specific vulnerabilities to quickly find possible targets. Attackers can exploit vulnerabilities in hosts whose IIDs they have previously obtained, or scan an address space to find potential targets.

## 4. Privacy and security properties of address generation mechanisms

Analysis of the extent to which a particular host is protected against the threats described in Section 3 depends on how each of a host's addresses is generated and used. In some scenarios, a host configures a single global address and uses it for all communications. In other scenarios, a host configures multiple addresses using different mechanisms and may use any or all of them.

[RFC3041] (later obsoleted by [RFC4941]) sought to address some of the problems described in Section 3 by defining "temporary addresses" for outbound connections. Temporary addresses are meant to supplement the other addresses that a device might use, not to replace them. They use IIDs that are randomly generated and change daily by default. The idea was for temporary addresses to be used for outgoing connections (e.g., web browsing) while maintaining the ability to use a stable address when more address stability is desired (e.g., for IPv6 addresses published in the DNS).

[RFC3484] originally specified that stable addresses be used for outbound connections unless an application explicitly prefers temporary addresses. The default preference for stable addresses was established to avoid applications potentially failing due to the short lifetime of temporary addresses or the possibility of a reverse look-up failure or error. However, [RFC3484] allowed that "implementations for which privacy considerations outweigh these application compatibility concerns MAY reverse the sense of this rule" and instead prefer by default temporary addresses rather than stable addresses. Indeed most implementations (notably including Windows) chose to default to temporary addresses for outbound connections since privacy was considered more important (and few applications supported IPv6 at the time, so application compatibility concerns were minimal). [RFC6724] then obsoleted [RFC3484] and changed the default to match what implementations actually did.

The envisioned relationship in [RFC3484] between stability of an address and its use in "public" can be misleading when conducting privacy analysis. The stability of an address and the extent to

which it is linkable to some other public identifier are independent of one another. For example, there is nothing that prevents a host from publishing a temporary address in a public place, such as the DNS. Publishing both a stable address and a temporary address in the DNS or elsewhere where they can be linked together by a public identifier allows the host's activities when using either address to be correlated together.

Moreover, because temporary addresses were designed to supplement other addresses generated by a host, the host may still configure a more stable address even if it only ever intentionally uses temporary addresses (as source addresses) for communication to off-link destinations. An attacker can probe for the stable address even if it is never used as such a source address or advertised (e.g., in DNS or SIP) outside the link.

This section compares the privacy and security properties of a variety of IID generation mechanisms and their possible usage scenarios, including scenarios in which a single mechanism is used to generate all of a host's IIDs and those in which temporary addresses are used together with addresses generated using a different IID generation mechanism. The analysis of the exposure of each IID type to correlation assumes that IPv6 prefixes are shared by a reasonably large number of nodes. As [RFC4941] notes, if a very small number of nodes (say, only one) use a particular prefix for an extended period of time, the prefix itself can be used to correlate the host's activities regardless of how the IID is generated. For example, [RFC3314] recommends that prefixes be uniquely assigned to mobile handsets where IPv6 is used within GPRS. In cases where this advice is followed and prefixes persist for extended periods of time (or get reassigned to the same handsets whenever those hand sets reconnect to the same network router), hosts' activities could be correlatable for longer periods than the analysis below would suggest.

The table below provides a summary of the whole analysis. A "No" entry indicates that the attack is prevented from being carried out on the basis of the IID, but the host may still be vulnerable depending on how it employs other protocols.

Mechanism(s)	Correlation	Location tracking	Address scanning	Device exploits
IEEE identifier	For device lifetime	For device lifetime	Possible	Possible
Static manual	For address lifetime	For address lifetime	Depends on generation mechanism	Depends on generation mechanism
Constant, semantically opaque	For address lifetime	For address lifetime	No	No
CGA	For lifetime of (modifier block + public key)	No	No	No
Stable, semantically opaque	Within single IPv6 link	No	No	No
Temporary	For temp address lifetime	No	No	No
DHCPv6	For lease lifetime	No	Depends on generation mechanism	No

Table 1: Privacy and security properties of IID generation mechanisms

#### 4.1. IEEE-identifier-based IIDs

As discussed in Section 3, addresses that use IIDs based on IEEE identifiers are vulnerable to all four threats. They allow correlation and location tracking for the lifetime of the device since IEEE identifiers last that long and their structure makes address scanning and device exploits possible.

#### 4.2. Static, manually configured IIDs

Because static, manually configured IIDs are stable, both correlation and location tracking are possible for the life of the address.

The extent to which location tracking can be successfully performed depends, to a some extent, on the uniqueness of the employed IID. For example, one would expect "low byte" IIDs to be more widely reused than, for example, IIDs where the whole 64-bits follow some pattern that is unique to a specific organization. Widely reused IIDs will typically lead to false positives when performing location tracking.

Whether manually configured addresses are vulnerable to address scanning and device exploits depends on the specifics of how the IIDs are generated.

#### 4.3. Constant, semantically opaque IIDs

Although a mechanism to generate a constant, semantically opaque IID has not been standardized, it has been in wide use for many years on at least one platform (Windows). Windows uses the [RFC4941] random generation mechanism in lieu of generating an IEEE-identifier-based IID. This mitigates the device-specific exploitation and address scanning attacks, but still allows correlation and location tracking because the IID is constant across IPv6 links and time.

#### 4.4. Cryptographically generated IIDs

Cryptographically generated addresses (CGAs) [RFC3972] bind a hash of the host's public key to an IPv6 address in the SEcure Neighbor Discovery (SEND) [RFC3971] protocol. CGAs may be regenerated for each subnet prefix, but this is not required given that they are computationally expensive to generate. A host using a CGA can be correlated for as long as the lifetime of the combination of the public key and the chosen modifier block, since it is possible to rotate modifier blocks without generating new public keys. Because the cryptographic hash of the host's public key uses the subnet prefix as an input, even if the host does not generate a new public key or modifier block when it moves to a different IPv6 link, its location cannot be tracked via the IID. CGAs do not allow device-specific exploitation or address scanning attacks.

#### 4.5. Stable, semantically opaque IIDs

[RFC7217] specifies an algorithm that generates, for each network interface, a unique random IID per IPv6 link. The aforementioned algorithm is employed not only for global unicast addresses, but also

for unique local unicast addresses and link-local unicast addresses, since these addresses may leak out via application protocols (e.g., IPv6 addresses embedded in email headers).

A host that stays connected to the same IPv6 link could therefore be tracked at length, whereas a mobile host's activities could only be correlated for the duration of each network connection. Location tracking is not possible with these addresses. They also do not allow device-specific exploitation or address scanning attacks.

#### 4.6. Temporary IIDs

A host that uses only a temporary address mitigates all four threats. Its activities may only be correlated for the lifetime a single temporary address.

A host that configures both an IEEE-identifier-based IID and temporary addresses makes the host vulnerable to the same attacks as if temporary addresses were not in use, although the viability of some of them depends on how the host uses each address. An attacker can correlate all of the host's activities for which it uses its IEEE-identifier-based IID. Once an attacker has obtained the IEEE-identifier-based IID, location tracking becomes possible on other IPv6 links even if the host only makes use of temporary addresses on those other IPv6 links; the attacker can actively probe the other IPv6 links for the presence of the IEEE-identifier-based IID. Device-specific vulnerabilities can still be exploited. Address scanning is also still possible because the IEEE-identifier-based address can be probed.

If the host instead generates a constant, semantically opaque IID to use in a stable address for server-like connections together with temporary addresses for outbound connections (as is the default in Windows), it sees some improvements over the previous scenario. The address scanning and device-specific exploitation attacks are no longer possible because the OUI is no longer embedded in any of the host's addresses. However, correlation of some activities across time and location tracking are both still possible because the semantically opaque IID is constant. And once an attacker has obtained the host's semantically opaque IID, location tracking is possible on any network by probing for that IID, even if the host only uses temporary addresses on those networks. However, if the host generates but never uses a constant, semantically opaque IID, it mitigates all four threats.

When used together with temporary addresses, the stable, semantically opaque IID generation mechanism [RFC7217] improves upon the previous scenario by limiting the potential for correlation to the lifetime of

the stable address (which may still be lengthy for hosts that are not mobile) and by eliminating the possibility for location tracking (since a different IID is generated for each subnet prefix). As in the previous scenario, a host that configures but does not use a stable, semantically opaque address mitigates all four threats.

#### 4.7. DHCPv6 generation of IIDs

The security/privacy implications of DHCPv6-based addresses will typically depend on whether the client requests an IA\_NA (Identity Association for Non-temporary Addresses) or an IA\_TA (Identity Association for Temporary Addresses) [RFC3315] and the specific DHCPv6 server software being employed.

DHCPv6 temporary addresses have the same properties as SLAAC temporary addresses Section 4.6 [RFC4941]. On the other hand, the properties of DHCPv6 non-temporary addresses typically depend on the specific DHCPv6 server software being employed. Recent releases of most popular DHCPv6 server software typically lease random addresses with a similar lease time as that of IPv4. Thus, these addresses can be considered to be "stable, semantically opaque". [I-D.ietf-dhc-stable-privacy-addresses] specifies an algorithm that can be employed by DHCPv6 servers to generate "stable, semantically opaque" addresses.

On the other hand, some DHCPv6 software leases sequential addresses (typically low-byte addresses). These addresses can be considered to be stable addresses. The drawback of this address generation scheme compared to "stable, semantically opaque" addresses is that, since they follow specific patterns, they enable IPv6 address scans.

#### 4.8. Transition/co-existence technologies

Addresses specified based on transition/co-existence technologies that embed an IPv4 address within an IPv6 address are not included in Table 1 because their privacy and security properties are inherited from the embedded address. For example, Teredo [RFC4380] specifies a means to generate an IPv6 address from the underlying IPv4 address and port, leaving many other bits set to zero. This makes it relatively easy for an attacker to scan for IPv6 addresses by guessing the Teredo client's IPv4 address and port (which for many NATs is not randomized). For this reason, popular implementations (e.g., Windows), began deviating from the standard by including 12 random bits in place of zero bits. This modification was later standardized in [RFC5991].

Some other transition technologies (e.g., [RFC5214], [RFC6052]) specify means to generate an IPv6 address from an underlying IPv4

address without a port. Such mechanisms thus make it much easier for an attacker to conduct an address scan than for mechanisms that require finding a port number as well.

Finally, still other mechanisms (e.g., [RFC7596], [RFC7597], [RFC7599]) are somewhere in between, using an IPv4 address and a port set ID (which for many NATs is not randomized). In general, such mechanisms are thus typically as easy to scan as in the Teredo example above without the 12-bit mitigation.

## 5. Miscellaneous Issues with IPv6 addressing

### 5.1. Network Operation

It is generally agreed that IPv6 addresses that vary over time in a specific IPv6 link tend to increase the complexity of event logging, trouble-shooting, enforcement of access controls and quality of service, etc. As a result, some organizations disable the use of temporary addresses [RFC4941] even at the expense of reduced privacy [Broersma].

### 5.2. Compliance

Some IPv6 compliance testing suites required (and might still require) implementations to support IEEE-identifier-based IIDS in order to be approved as compliant. This document recommends that compliance testing suites be relaxed to allow other forms of address generation that are more amenable to privacy.

### 5.3. Intellectual Property Rights (IPRs)

Some IPv6 addressing techniques might be covered by Intellectual Property rights, which might limit their implementation in different Operating Systems. [CGA-IPR] and [KAME-CGA] discuss the IPRs on CGAs.

## 6. Security Considerations

This whole document concerns the privacy and security properties of different IPv6 address generation mechanisms.

## 7. IANA Considerations

This document does not require actions by IANA.

## 8. Acknowledgements

The authors would like to thank Bernard Aboba, Brian Carpenter, Tim Chown, Lorenzo Colitti, Rich Draves, Robert Hinden, Robert Moskowitz, Erik Nordmark, Mark Smith, Ole Troan, and James Woodyatt for providing valuable comments on earlier versions of this document.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<http://www.rfc-editor.org/info/rfc2464>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<http://www.rfc-editor.org/info/rfc3971>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<http://www.rfc-editor.org/info/rfc3972>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<http://www.rfc-editor.org/info/rfc4380>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.



- [RFC5991] Thaler, D., Krishnan, S., and J. Hoagland, "Teredo Security Updates", RFC 5991, DOI 10.17487/RFC5991, September 2010, <<http://www.rfc-editor.org/info/rfc5991>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <<http://www.rfc-editor.org/info/rfc7136>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.

## 9.2. Informative References

- [Broersma] Broersma, R., "IPv6 Everywhere: Living with a Fully IPv6-enabled environment", Australian IPv6 Summit 2010, Melbourne, VIC Australia, October 2010, October 2010, <[http://www.ipv6.org.au/10ipv6summit/talks/Ron\\_Broersma.pdf](http://www.ipv6.org.au/10ipv6summit/talks/Ron_Broersma.pdf)>.
- [CGA-IPR] IETF, "Intellectual Property Rights on RFC 3972", 2005, <<https://datatracker.ietf.org/ipr/676/>>.
- [I-D.ietf-dhc-stable-privacy-addresses] Gont, F. and S. LIU, "A Method for Generating Semantically Opaque Interface Identifiers with Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", draft-ietf-dhc-stable-privacy-addresses-02 (work in progress), April 2015.
- [I-D.ietf-opsec-ipv6-host-scanning] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning-08 (work in progress), August 2015.
- [KAME-CGA] KAME, "The KAME IPR policy and concerns of some technologies which have IPR claims", 2005, <<http://www.kame.net/newsletter/20040525/>>.

- [Microsoft] Microsoft, "IPv6 interface identifiers", 2013, <target='http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag\_ip\_v6\_imp\_addr7.mspx?mfr=true>.
- [Panopticlick] Electronic Frontier Foundation, "Panopticlick", 2011, <http://panopticlick.eff.org>.
- [RFC1971] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 1971, DOI 10.17487/RFC1971, August 1996, <http://www.rfc-editor.org/info/rfc1971>.
- [RFC1972] Crawford, M., "A Method for the Transmission of IPv6 Packets over Ethernet Networks", RFC 1972, DOI 10.17487/RFC1972, August 1996, <http://www.rfc-editor.org/info/rfc1972>.
- [RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, DOI 10.17487/RFC3041, January 2001, <http://www.rfc-editor.org/info/rfc3041>.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.
- [RFC3314] Wasserman, M., Ed., "Recommendations for IPv6 in Third Generation Partnership Project (3GPP) Standards", RFC 3314, DOI 10.17487/RFC3314, September 2002, <http://www.rfc-editor.org/info/rfc3314>.
- [RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <http://www.rfc-editor.org/info/rfc3484>.
- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <http://www.rfc-editor.org/info/rfc5214>.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <http://www.rfc-editor.org/info/rfc6052>.

- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<http://www.rfc-editor.org/info/rfc6265>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<http://www.rfc-editor.org/info/rfc6973>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<http://www.rfc-editor.org/info/rfc7421>>.
- [RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <<http://www.rfc-editor.org/info/rfc7596>>.
- [RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <<http://www.rfc-editor.org/info/rfc7597>>.
- [RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <<http://www.rfc-editor.org/info/rfc7599>>.

#### Authors' Addresses

Alissa Cooper  
Cisco  
707 Tasman Drive  
Milpitas, CA 95035  
US  
  
Phone: +1-408-902-3950  
Email: [alcoop@cisco.com](mailto:alcoop@cisco.com)  
URI: <https://www.cisco.com/>

Fernando Gont  
Huawei Technologies  
Evaristo Carriego 2644  
Haedo, Provincia de Buenos Aires 1706  
Argentina

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

Dave Thaler  
Microsoft  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

Phone: +1 425 703 8835  
Email: dthaler@microsoft.com

6man Working Group  
Internet-Draft  
Updates: 3306,3956,4291 (if approved)  
Intended status: Standards Track  
Expires: February 12, 2015

M. Boucadair  
France Telecom  
S. Venaas  
Cisco  
August 11, 2014

Updates to the IPv6 Multicast Addressing Architecture  
draft-ietf-6man-multicast-addr-arch-update-08

Abstract

This document updates the IPv6 multicast addressing architecture by re-defining the reserved bits as generic flag bits. The document provides also some clarifications related to the use of these flag bits.

This document updates RFC 3956, RFC 3306 and RFC 4291.

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].

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 12, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

#### Table of Contents

1. Introduction . . . . .	3
2. Addressing Architecture Update . . . . .	3
3. Flag Bits: New Processing Rules . . . . .	3
4. RFC Updates . . . . .	4
4.1. RFC 3306 . . . . .	4
4.1.1. Update #1 . . . . .	4
4.1.2. Update #2 . . . . .	5
4.2. RFC 3956 . . . . .	6
4.2.1. Update #1 . . . . .	6
4.2.2. Update #2 . . . . .	7
4.2.3. Update #3 . . . . .	8
4.2.4. Update #4 . . . . .	8
5. IANA Considerations . . . . .	9
6. Security Considerations . . . . .	9
7. Acknowledgements . . . . .	9
8. References . . . . .	9
8.1. Normative References . . . . .	9
8.2. Informative References . . . . .	9
Authors' Addresses . . . . .	10

## 1. Introduction

This document updates the IPv6 addressing architecture [RFC4291] by re-defining reserved bits as generic flag bits (Section 2). The document provides also some clarifications related to the use of these flag bits (Section 3).

This document updates [RFC3956], [RFC3306], and [RFC4291]. These updates are logical consequences of the new processing rules in Section 3.

Textual representation of IPv6 addresses included in the RFC updates follows the recommendation in [RFC5952].

## 2. Addressing Architecture Update

Bits 17-20 of a multicast address, where bit 1 is the most significant bit, are defined in [RFC3956] and [RFC3306] as reserved bits. This document defines these bits as generic flag bits so that they apply to any multicast address. These bits are referred to as ff2 (flag field 2) while the flgs bits in [RFC4291][RFC3956] are renamed to ff1 (flag field 1).

Within this document, flag bits denote both ff1 and ff2.

Defining the bits 17-20 as flags for all IPv6 multicast addresses allows addresses to be treated in a more uniform and generic way, and allows for these bits to be defined in the future for different purposes, irrespective of the specific type of multicast address. For the record, this design choice was initially triggered by the specification in [I-D.ietf-mboned-64-multicast-address-format] which proposed for associating a meaning with one of the reserved bits. Moreover, [I-D.ietf-mboned-64-multicast-address-format] considered also the use of the last remaining flag in ff1 but that approach was abandoned because it is not clear at this stage whether there is other usage scenarios of the flag.

Section 4 specifies the updated structure of the addressing architecture.

Further specification documents may define a meaning for these flag bits.

## 3. Flag Bits: New Processing Rules

Some implementations and specification documents do not treat the flag bits as separate bits but tend to use their combined value as a 4-bit integer. This practice is a hurdle for assigning a meaning to

the remaining flag bits. Below are listed some examples for illustration purposes:

- o the reading of [RFC3306] may lead to conclude that ff3x::/32 is the only allowed Source Specific Multicast (SSM) IPv6 prefix block.
- o [RFC3956] states only ff70::/12 applies to Embedded-RP. Particularly, implementations should not treat the fff0::/12 range as Embedded-RP.

To avoid such confusion and to unambiguously associate a meaning with the remaining flags, the following requirement is made:

Implementations MUST treat flag bits as separate bits.

#### 4. RFC Updates

##### 4.1. RFC 3306

##### 4.1.1. Update #1

This document changes Section 4 of [RFC3306] as follows:

OLD:

8	4	4	8	8	64	32
11111111	flgs	scop	reserved	plen	network prefix	group ID
+	+	+	+	+	+	+

flgs is a set of 4 flags:

0	0	P	T
---	---	---	---

- o P = 0 indicates a multicast address that is not assigned based on the network prefix. This indicates a multicast address as defined in [ADDRARCH].
- o P = 1 indicates a multicast address that is assigned based on the network prefix.
- o If P = 1, T MUST be set to 1, otherwise the setting of the T bit is defined in Section 2.7 of [ADDRARCH].

The reserved field MUST be zero.



Note: [ADDRARCH] is a reference listed in [RFC3306]. [ADDRARCH] has been since obsoleted by [RFC4291].

NEW:

8	4	4	4	4	8	64	32
11111111	ff1	scop	ff2	rsvd	plen	network prefix	group ID

```

ffl (flag field 1) is a set of 4 flags:
                                     +-+--+--+
                                     |X|Y|P|T|
                                     +-+--+--+

```

X and Y may each be set to 0 or 1. Note, X is for future assignment while a meaning is associated with Y in RFC3956.

- o P = 0 indicates a multicast address that is not assigned based on the network prefix. This indicates a multicast address as defined in [RFC4291].
- o P = 1 indicates a multicast address that is assigned based on the network prefix.
- o If P = 1, T MUST be set to 1, otherwise the setting of the T bit is defined in Section 2.7 of [RFC4291].

```

ff2 (flag field 2) is a set of 4 flags:      +-+--+--+
|r|r|r|r|                                     +-+--+--+

```

where "rrrr" are for future assignment as additional flag bits.  
r bits MUST each be sent as zero and MUST be ignored on receipt.

Flag bits denote both ff1 and ff2.

#### 4.1.2. Update #2

This document changes Section 6 of [RFC3306] as follows:

OLD :

These settings create an SSM range of FF3x::/32 (where 'x' is any valid scope value). The source address field in the IPv6 header identifies the owner of the multicast address.

NEW :

If the flag bits in ff1 are set to 0011, these settings create an SSM range of ff3x::/32 (where 'x' is any valid scope value). The source address field in the IPv6 header identifies the owner of the multicast address. ff3x::/32 is not the only allowed SSM prefix range. For example if the most significant flag bit in ff1 is set, then we would get the SSM range ffbx::/32.

## 4.2. RFC 3956

### 4.2.1. Update #1

This document changes Section 2 of [RFC3956] as follows:

OLD:

As described in [RFC3306], the multicast address format is as follows:

	8		4		4		8		8		64		32	
+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
	11111111		flgs		scop		reserved		plen		network prefix		group ID	
+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+

Where flgs are "0011". (The first two bits are as yet undefined, sent as zero and ignored on receipt.)

NEW:

The multicast address format is as follows:

	8		4		4		4		4		8		64		32	
+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
	11111111		ff1		scop		ff2		rsvd		plen		network prefix		group ID	
+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+

ff1 (flag field 1) is a set of four flags:      +--+--+--+  
    |X|R|P|T|  
    +--+--+--+

where X is for future assignment as additional flag bit. X may be set to 0 or 1.

ff2 (flag field 2) is a set of 4 flags:      +--+--+--+  
    |r|r|r|r|  
    +--+--+--+

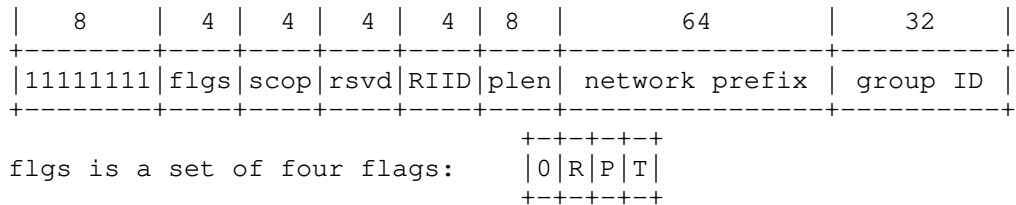
where "rrrr" are for future assignment as additional flag bits. r bits MUST each be sent as zero and MUST be ignored on receipt.

Flag bits denote both ff1 and ff2.

## 4.2.2. Update #2

This document changes Section 3 of [RFC3956] as follows:

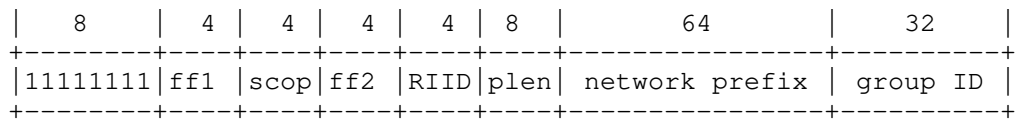
OLD:



When the highest-order bit is 0, R = 1 indicates a multicast address that embeds the address on the RP. Then P MUST be set to 1, and consequently T MUST be set to 1, as specified in [RFC3306]. In effect, this implies the prefix FF70::/12. In this case, the last 4 bits of the previously reserved field are interpreted as embedding the RP interface ID, as specified in this memo.

The behavior is unspecified if P or T is not set to 1, as then the prefix would not be FF70::/12. Likewise, the encoding and the protocol mode used when the two high-order bits in "flgs" are set to 11 ("FFF0::/12") is intentionally unspecified until such time that the highest-order bit is defined. Without further IETF specification, implementations SHOULD NOT treat the FFF0::/12 range as Embedded-RP.

NEW:



ff1 is a set of four flags:

X	R	P	T
---	---	---	---

where X is for future assignment as additional flag bit. X may be set to 0 or 1.

R = 1 indicates a multicast address that embeds the address of the RP. Then P MUST be set to 1, and consequently T MUST be set to 1, according to [RFC3306], as this is a special case of unicast-prefix based addresses. This implies that, for instance, prefixes ff70::/12 and fff0::/12 are embedded RP prefixes. When the R-bit is set, the last 4 bits of the field that were reserved in [RFC3306] are interpreted as embedding the RP interface ID, as specified in this memo.

#### 4.2.3. Update #3

This document changes Section 4 of [RFC3956] as follows:

OLD:

It MUST be a multicast address with "flgs" set to 0111, that is, to be of the prefix FF70::/12,

NEW:

It MUST be a multicast address with R-bit set to 1.

It MUST have P-bit and T-bit both set to 1 when using the embedding in this document as it is a prefix-based address.

#### 4.2.4. Update #4

This document changes Section 7.1 of [RFC3956] as follows:

OLD:

To avoid loops and inconsistencies, for addresses in the range FF70::/12, the Embedded-RP mapping MUST be considered the longest possible match and higher priority than any other mechanism.

NEW:

To avoid loops and inconsistencies, for addresses with R-bit set to 1, the Embedded-RP mapping MUST be considered the longest possible match and higher priority than any other mechanism.

## 5. IANA Considerations

This document does not require any action from IANA.

## 6. Security Considerations

The same security considerations as those discussed in [RFC3956], [RFC3306] and [RFC4291] are to be taken into account.

## 7. Acknowledgements

Special thanks to Brian Haberman for the discussions prior to the publication of this document.

Many thanks to Jouni Korhonen, Tatuya Jinmei, Charlie Kaufman, and Ben Campbell for their review.

## 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.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, November 2004.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

### 8.2. Informative 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-05 (work in progress), April 2013.

Authors' Addresses

Mohamed Boucadair  
France Telecom  
Rennes 35000  
France

Email: mohamed.boucadair@orange.com

Stig Venaas  
Cisco  
USA

Email: stig@cisco.com

6man Working Group  
Internet-Draft  
Updates: 4861 (if approved)  
Intended status: Standards Track  
Expires: October 11, 2015

S. Krishnan  
Ericsson  
D. Anipko  
Unaffiliated  
D. Thaler  
Microsoft  
April 9, 2015

Packet loss resiliency for Router Solicitations  
draft-ietf-6man-resilient-rs-06

Abstract

When an interface on a host is initialized, the host transmits Router Solicitations in order to minimize the amount of time it needs to wait until the next unsolicited multicast Router Advertisement is received. In certain scenarios, these router solicitations transmitted by the host might be lost. This document specifies a mechanism for hosts to cope with the loss of the initial Router Solicitations.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 11, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Conventions used in this document . . . . .	2
2. Proposed algorithm . . . . .	4
2.1. Stopping the retransmissions . . . . .	4
3. Configuring the use of retransmissions . . . . .	5
4. Known Limitations . . . . .	5
5. IANA Considerations . . . . .	5
6. Security Considerations . . . . .	5
7. Acknowledgements . . . . .	5
8. References . . . . .	5
8.1. Normative References . . . . .	6
8.2. Informative References . . . . .	6
Authors' Addresses . . . . .	6

## 1. Introduction

As specified in [RFC4861], when an interface on a host is initialized, in order to obtain Router Advertisements quickly, a host transmits up to MAX\_RTR\_SOLICITATIONS (3) Router Solicitation messages, each separated by at least RTR\_SOLICITATION\_INTERVAL (4) seconds. In certain scenarios, these router solicitations transmitted by the host might be lost. e.g. The host is connected to a bridged residential gateway over Ethernet or WiFi. LAN connectivity is achieved at interface initialization, but the upstream WAN connectivity is not active yet. In this case, the host just gives up after the initial RS retransmits.

Once the initial RSs are lost, the host gives up and assumes that there are no routers on the link as specified in Section 6.3.7 of [RFC4861]. The host will not have any form of Internet connectivity until the next unsolicited multicast Router Advertisement is received. These Router Advertisements are transmitted at most MaxRtrAdvInterval seconds apart (maximum value 1800 seconds). Thus in the worst case scenario a host would be without any connectivity for 30 minutes. This delay may be unacceptable in some scenarios.

### 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].

## 2. Proposed algorithm

To achieve resiliency to packet loss, the host needs to continue retransmitting the Router Solicitations until it receives a Router Advertisement, or until it is willing to accept that no router exists. If the host continues retransmitting the RSs at RTR\_SOLICITATION\_INTERVAL second intervals, it may cause excessive network traffic if a large number of such hosts exists. To achieve resiliency while keeping the aggregate network traffic low, the host can use some form of exponential backoff algorithm to retransmit the RSs.

Hosts complying to this specification MUST use the exponential backoff algorithm for retransmits that is described in Section 14 of [RFC3315] in order to continuously retransmit the Router Solicitations until a Router Advertisement is received. The hosts SHOULD use the following variables as input to the retransmission algorithm:

IRT 4 seconds

MRT 3600 seconds

MRC 0

MRD 0

The initial value IRT was chosen to be in line with the current retransmission interval (RTR\_SOLICITATION\_INTERVAL) that is specified by [RFC4861] and the maximum retransmission time MRT was chosen to be in line with the new value of SOL\_MAX\_RT as specified by [RFC7083]. This is to ensure that the short term behavior of the RSs is similar to what is experienced in current networks, and longer term persistent retransmission behavior trends towards being similar to that of DHCPv6 [RFC3315] [RFC7083].

### 2.1. Stopping the retransmissions

On multicast-capable links, the hosts following this specification SHOULD stop retransmitting the RSs when Router Discovery is successful (i.e. an RA with a non-zero Router Lifetime that results in a default route is received). If an RA is received from a router and it does not result in a default route (i.e. Router Lifetime is zero) the host MUST continue retransmitting the RSs.

On non-multicast links, the hosts following this specification MUST continue retransmitting the RSs even after an RA that results in a default route is received. This is required because, in such links,

sending an RA can only be triggered by an RS. Please note that such links have special mechanisms for sending RSes as well. e.g. The mechanism specified in Section 8.3.4. of ISATAP [RFC5214] unicasts the RSes to specific routers.

### 3. Configuring the use of retransmissions

Implementations of this specification are encouraged to provide a configuration option to enable or disable potentially infinite RS retransmissions. If a configuration option is provided, it **MUST** enable RS retransmissions by default. Providing an option to enable/disable retransmissions on a per-interface basis allows network operators to configure RS behavior most applicable to each connected link.

### 4. Known Limitations

When an IPv6-capable host attaches to a network that does not have IPv6 enabled, it transmits 3 (MAX\_RTR\_SOLICITATIONS) Router Solicitations as specified in [RFC4861]. If it receives no Router Advertisements, it assumes that there are no routers present on the link and it ceases to send further RSs. With the mechanism specified in this document, the host will continue to retransmit RSs indefinitely at the rate of approximately 1 RS per hour. It is unclear how to differentiate between such a network with no IPv6 routers and a link where an IPv6 router is temporarily unreachable but could become reachable in the future.

### 5. IANA Considerations

This document does not require any IANA actions.

### 6. Security Considerations

This document does not present any additional security issues beyond those discussed in [RFC4861] and those RFCs that update [RFC4861].

### 7. Acknowledgements

The authors would like to thank Steve Baillargeon, Erik Kline, Andrew Yourtchenko, Ole Troan, Erik Nordmark, Lorenzo Colitti, Thomas Narten, Ran Atkinson, Allison Mankin, Les Ginsberg, Brian Carpenter, Barry Leiba, Brian Haberman, Spencer Dawkins, Alia Atlas, Stephen Farrell and Mehmet Ersue for their reviews and suggestions that made this document better.

### 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.
- [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.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC7083] Droms, R., "Modification to Default Values of SOL\_MAX\_RT and INF\_MAX\_RT", RFC 7083, November 2013.

## 8.2. Informative References

- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, March 2008.

## Authors' Addresses

Suresh Krishnan  
Ericsson  
8400 Decarie Blvd.  
Town of Mount Royal, QC  
Canada

Phone: +1 514 345 7900 x42871  
Email: suresh.krishnan@ericsson.com

Dmitry Anipko  
Unaffiliated

Phone: +1 425 442 6356  
Email: dmitry.anipko@gmail.com

Dave Thaler  
Microsoft  
One Microsoft Way  
Redmond, WA  
USA

Email: dthaler@microsoft.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 24, 2014

J. Wu  
C. Liu  
Y. Cui  
Tsinghua University  
October 21, 2013

Identifying Addresses of IPv6 Tunnel Packets at Tunnel Exit-point  
draft-liu-6man-ident-tunnel-packet-addr-00

Abstract

In the networks where IPv6 tunneling is used, it is not specific about how a tunnel end-node identifies the received tunnel packets by checking the destination and source addresses. when the tunnel end-node is configured with multiple IPv6 addresses or multiple IPv6 tunnel instances, such identification is necessary. This document describes the problem and defines the behavior of IPv6 tunnel end-nodes about identifying tunnel packets.

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 24, 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.

## Table of Contents

1. Introduction . . . . .	2
2. Requirements Language . . . . .	2
3. Terminology . . . . .	2
4. Problem Statement . . . . .	3
5. Tunnel End-node Behavior . . . . .	4
5.1. Acceptable Local/Remote Address Set Maintenance . . . . .	4
5.2. Inbound Tunnel Packet Identification . . . . .	4
6. Security Considerations . . . . .	4
7. IANA Considerations . . . . .	5
8. References . . . . .	5
8.1. Normative References . . . . .	5
8.2. Informative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

IPv6 tunneling mechanism [RFC2473] provides support for various protocols to work in IPv6-only network. But when a tunnel end-node receives a tunnel packet, it is not specific about whether or not the tunnel end-node should identify the tunnel packet by checking the destination and source addresses in the packet. When the tunnel end-node is configured with multiple IPv6 tunnel instance, it is also undefined how to dispatch the received tunnel packet to each tunnel instance. This document provides a solution to this problem by defining the behavior of tunnel end-node on how to identify received tunnel packets.

## 2. Requirements Language

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

## 3. Terminology

This document makes use of the following terms:

Acceptable Local Address Set: A set of one or more IPv6 addresses or prefixes maintained by a tunnel instance. It represents the set of acceptable IPv6 destination addresses in the received IPv6 tunnel packets.

Usually it consists of one or more IPv6 addresses on local interfaces of the tunnel end-node.

Acceptable Remote Address Set: A set of one or more IPv6 addresses or prefixes maintained by a tunnel instance. It represents the set of acceptable IPv6 source addresses in the received IPv6 tunnel packets.

Terminology defined in [RFC2473] is used extensively in this document.

#### 4. Problem Statement

Consider an IPv6 tunnel end-node with multiple IPv6 addresses configured on its interface. One of the IPv6 addresses is chosen as the tunnel entry-point node address [RFC2473]. When the tunnel end-node receives an IPv6 tunnel packet with its destination address other than the tunnel entry-point node address, the tunnel end-node either discards it or accepts it. Section 3.3 of [RFC2473] states that:

"Upon receiving an IPv6 packet destined to an IPv6 address of a tunnel exit-point node, its IPv6 protocol layer processes the tunnel headers."

According to this statement, the tunnel end-node ought to accept the tunnel packet. However, when the destination IPv6 address is used to distinguish among multiple tunnel instances running in the same tunnel end-node, one tunnel packet may be passed to multiple tunnel instances, and it may not be the expected result.

When there are multiple tunnel instances in a tunnel end-node and each instance with a separated process engine, it must be decided about which tunnel instance(s) to be chosen to process a received IPv6 tunnel packet. As the payload of a IPv6 tunnel packet may be various protocols, only 3 items may be used to identify a tunnel packet: IPv6 destination address, IPv6 source address, and the payload protocol type (the Next Header field in IPv6 tunnel packet). The payload protocol type can be used to distinguish between an IPv4-in-IPv6 tunnel and an IPv6-in-IPv6 tunnel. The IPv6 source and destination address can be used to distinguish among tunnels of the same protocol type.

There are several IPv6 transition mechanisms relies on point-to-multipoint IPv6 tunnel, such as DS-Lite [RFC6333], Lightweight 4over6 [I-D.ietf-softwire-lw4over6], MAP-E [I-D.ietf-softwire-map], etc. In

these mechanisms, one tunnel instance may have multiple remote tunnel-ends, each with different IPv6 unicast addresses. Thus, the acceptable destination / source address of a inbound tunnel packet could be multiple address.

## 5. Tunnel End-node Behavior

### 5.1. Acceptable Local/Remote Address Set Maintenance

An IPv6 tunnel end-node maintains an Acceptable Local Address Set in each of its tunnel instance. An Acceptable Local Address Set contains one or more IPv6 addresses or prefixes. The destination address of an inbound IPv6 tunnel packet to be passed to a tunnel instance MUST match a record in the Acceptable Local Address Set of the tunnel instance.

An IPv6 tunnel end-node maintains an Acceptable Remote Address Set in each of its tunnel instance. An Acceptable Remote Address Set contains one or more IPv6 addresses or prefixes. The source address of an inbound IPv6 tunnel packet to be passed to a tunnel instance MUST match a record in the Acceptable Remote Address Set of the tunnel instance.

For example, in a point-to-point IPv6 tunnel, the Acceptable Local Address Set contains one IPv6 address(tunnel entry-point node address [RFC2473]), and the Acceptable Remote Address Set contains one IPv6 address(tunnel exit-point node address [RFC2473]). In the tunnel model of DS-Lite AFTR [RFC6333], the Acceptable Remote Address Set may contains a IPv6 prefix ::/0, to represent that the tunnel accepts tunnel packets from any B4s.

### 5.2. Inbound Tunnel Packet Identification

When an IPv6 tunnel end-node receives an IPv6 tunnel packet, the tunnel end-node identifies the packet by comparing its IPv6 source address, IPv6 destination address and protocol type (Next Header) with the Acceptable Remote Address Set, Acceptable Local Address Set, and protocol type of each tunnel instance running on the node. If all the 3 fields match, the IPv6 tunnel packet is passed to the tunnel instance.

If a tunnel packet matches more than one tunnel instance, it is passed to each of the tunnel instances. If a tunnel packet matches no tunnel instance, the tunnel end-node MUST discard the packet, and SHOULD send a ICMPv6 error message to the source address of the tunnel packet. ICMPv6 type is TBD.

## 6. Security Considerations



TBD

## 7. IANA Considerations

This document does not include an IANA request.

## 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.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

### 8.2. Informative References

- [I-D.ietf-softwire-lw4over6]  
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-01 (work in progress), July 2013.
- [I-D.ietf-softwire-map]  
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-08 (work in progress), August 2013.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

## Authors' Addresses

Jianping Wu  
Tsinghua University  
Department of Computer Science, Tsinghua University  
Beijing 100084  
P.R.China

Phone: +86-10-6278-5983  
Email: jianping@cernet.edu.cn

Cong Liu  
Tsinghua University  
Department of Computer Science, Tsinghua University  
Beijing 100084  
P.R.China

Phone: +86-10-6278-5822  
Email: gnocuil@gmail.com

Yong Cui  
Tsinghua University  
Department of Computer Science, Tsinghua University  
Beijing 100084  
P.R.China

Phone: +86-10-6260-3059  
Email: yong@csnet1.cs.tsinghua.edu.cn

Network Working Group  
Internet Draft  
Intended status: Stand Track  
Expires: April 30, 2015

B. Liu  
Huawei Technologies  
October 27, 2014

IPv6 ND Option for Network Management Server Discovery  
draft-liu-6man-nd-nms-discovery-01.txt

#### 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 30, 2015.

#### 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.

## Abstract

This document introduces a mechanism for devices to actively learn the NMS server address from the neighbors through IPv6 ND protocol extension. It is a good leverage of IPv6 automatic features.

This document only discusses problem/solution within the IPv6-only networks/plane.

## Table of Contents

1. Introduction .....	3
2. Basic Approach .....	3
3. Scenario Description.....	3
3.1. New Devices Getting Online .....	3
3.2. Regarding Connectivity .....	4
4. Neighbor Discovery Extension for Supporting NMS Discovery ....	4
4.1. Option Definition .....	5
4.2. Sub-Options Definition .....	5
4.3. Option Carried in Router Advertisement Messages .....	6
4.4. Option Carried in Neighbor Solicit/Advertisement Messages	7
5. Security Considerations .....	7
6. IANA Considerations .....	7
7. Acknowledgments .....	7
8. References .....	7
8.1. Normative References .....	7

## 1. Introduction

NMS (Network Management System) has become a must-have component in modern networks. It could be utilized to benefit various aspects of a network. For example, the emerging router auto-configuration solutions are mostly based on NMS. If the devices could successfully connect to the NMS server(s), then auto-configuration won't be a problem.

So there is a key problem of how to discover the NMS server for the devices when they get online. Currently there are mainly two methods to solve the problem. One is to set the NMS server's IP/URL into the devices before shipping to the customer premises; the other one is the NMS actively discovering the devices through some polling mechanisms. The former one is easy to be implemented and deployed, but it lacks flexibility due to the static pre-configuration and might be error-prone for configuration when the different networks have different NMS servers; the latter one lacks the instantaneity due to the polling mechanisms need the intermediate nodes to integrate supporting features which introduce complex functions and protocols.

This document introduces a mechanism for devices to actively learn the NMS server from the neighbors through IPv6 ND protocol extension. It is a good leverage of IPv6 automatic features.

This document only discusses problem/solution within the IPv6-only scope.

## 2. Basic Approach

When a device gets online, we could assume that its neighbors who have already got online have learnt the NMS server's address. So it is quite easy for the new device to learn the information from its neighbor.

This document is based on the above Neighbor-Learning approach.

## 3. Scenario Description

### 3.1. New Devices Getting Online

- Adding a New Device into an Existing Network

For adding a new device into an existing network, it is very reasonable to assume that the neighbors have already connected to the NMS server. So it is obvious that the new device could easily learn the NMS server's address from neighbors.

#### - Deploying a New Network

In the case of deploying a new network, the NMS server address needs to be propagated to the whole network, then some kind of flooding mechanism is needed if the propagation also relies on above mentioned neighbor-learning approach. This is applicable through careful plan which might need proper order for the devices to get online successively.

The detail of the flooding mechanism is out of the scope of this document. We treat it as an assumption for the application of neighbor-learning NMS discovery.

### 3.2. Regarding Connectivity

#### - Connecting NMS after Getting Global Connectivity

Normally, address assignment is not coupled with NMS processing. Before connected to the NMS server, the devices could obtain global connectivity either through SLAAC or DHCPv6.

In this case, once the devices have learnt the NMS server address, they could directly connect to get more configurations.

#### - Connecting NMS before Getting Global Connectivity

In contrast, address assignment might be done through NMS in some situations. For example, the device is a backbone router, and the address has been carefully planned and pre-configured in the NMS server, when the device connect to the server, it will be assigned global address through network management processing.

In this case, after learning the NMS server address, the device might need a proxy to communicate with the server or configuring itself a ULA address and utilizing the NPTv6 processing on its neighbor or uplink router. The details are out of the scope of this document.

### 4. Neighbor Discovery Extension for Supporting NMS Discovery

Since ND is a basic protocol in IPv6, every router supports IPv6 would support ND, we utilize ND extension to achieve the above mentioned neighbor-learning NMS server discovery.



- o Length: 3
- o IPv6 Address: 128bit IPv6 address with zero padding behind

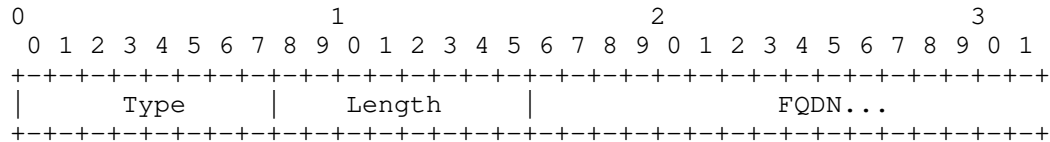


Figure 3: FQDN Sub-option of NMS Server Location

- o Type: TBD (to be assigned by IANA)
- o Length: The length of the option (including the type and length fields) in units of 8 octets.
- o FQDN: FQDN of the NMS server, variable length

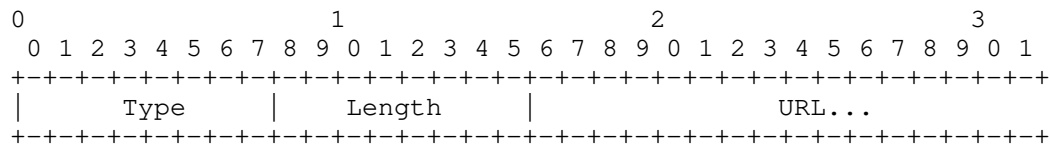


Figure 3: FQDN Sub-option of NMS Server Location

- o Type: TBD (to be assigned by IANA)
- o Length: The length of the option (including the type and length fields) in units of 8 octets.
- o URL: URL of the NMS server, variable length

#### 4.3. Option Carried in Router Advertisement Messages

- RA-only Mode

A device discovers NMS server's address through received Router Advertisement messages which include a new option defined for carrying NMS server's address.



Since RA messages are usually generated by the gateway on a link, this approach is suitable for a hub-and-spoke subnet in which a new device joins in.

After having learnt the NMS server's address, then the device could directly connect to the server

#### 4.4. Option Carried in Neighbor Solicit/Advertisement Messages

A device discovers NMS server's address through actively initiating Neighbor Solicit message and receiving Neighbor Advertisement messages which include the new option carrying the NMS server's address.

This approach is suitable for point-to-point or non-broad circuits.

### 5. Security Considerations

#### - Device authentication for NMS Servers

With applying the mechanism described in this document, the devices would actively connect to the NMS servers. So there might be stronger desire for the NMS servers to authenticate the devices.

#### - ND security

This document doesn't introduce more threats than original Neighbor Discovery protocol, so generally it aligns with the security considerations described in [RFC4861].

### 6. IANA Considerations

The newly defined options need IANA to assign type codes.

### 7. Acknowledgments

Many useful comments and contributions were made by Sheng Jiang.

### 8. References

#### 8.1. Normative References

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

Authors' Addresses

Bing Liu  
Huawei Technologies Co., Ltd  
Q14, Huawei Campus  
No.156 Beiqing Rd.  
Hai-Dian District, Beijing 100095  
P.R. China

Email: [leo.liubing@huawei.com](mailto:leo.liubing@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 24, 2014

Y. Cui  
C. Liu  
Tsinghua University  
October 21, 2013

IPv6 Tunnel MTU Configuration  
draft-liu-6man-tunnel-mtu-config-00

Abstract

It is not specific about how to decide IPv6 tunnel MTU in IPv6 tunneling mechanisms in some situations. This document describes the problem and provides a general solution to decide tunnel MTU value.

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 24, 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.

## Table of Contents

1. Introduction . . . . .	2
2. Requirements Language . . . . .	2
3. Terminology . . . . .	2
4. Problem Statement . . . . .	2
5. Tunnel MTU Configuration . . . . .	3
6. Security Considerations . . . . .	4
7. IANA Considerations . . . . .	4
8. References . . . . .	4
8.1. Normative References . . . . .	4
8.2. Informative References . . . . .	4
Authors' Addresses . . . . .	5

## 1. Introduction

IPv6 tunneling mechanism defined in [RFC2473] provides support for various protocols to work in IPv6-only network. Because IPv6 intermediate routers do not support fragmentation, a IPv6 tunnel packet may be discarded by an intermediate router if the packet size exceeds the next-hop MTU. Thus, the tunnel MTU of IPv6 tunnel nodes should be well decided and managed in order to eliminate data loss.

But [RFC2473] is not specific about how to decide tunnel MTU, when a tunnel entry-point connects to multiple tunnel exit-points. It is also not clear about how to decide tunnel MTU without Path MTU Discovery [RFC1981]. This document describes the problems and proposes a solution to specify the behavior of tunnel entry-point to configure its tunnel MTU.

## 2. Requirements Language

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

## 3. Terminology

Terminology defined in [RFC2473] is used extensively in this document.

## 4. Problem Statement

Section 6.7 of [RFC2473] defines the behavior of the IPv6 tunnel entry-point to set its tunnel MTU that:

"The tunnel MTU is set dynamically to the Path MTU between the tunnel entry-point and the tunnel exit-point nodes, minus the size

of the tunnel headers"

"The tunnel entry-point node performs Path MTU discovery on the path between the tunnel entry-point and exit-point nodes"

However, it is unspecific about how a tunnel entry-point sets its tunnel MTU without performing Path MTU discovery. As IPv6 tunneling is the foundation of several IPv6 transition mechanisms, but some of these mechanisms require tunnel end-nodes not to perform Path MTU discovery. DS-Lite [RFC6333] handles this by increasing the MTU size of all the links in the path by at least 40 bytes. MAP-E [I-D.ietf-software-map] strongly recommends to well manage the MTU of links in the whole MAP domain, which is similar to what DS-Lite does, and specifies that "A MAP BR SHOULD NOT by default use Path MTU discovery across the MAP domain" in section 8.3.1 of [I-D.ietf-software-map].

In [RFC2473], tunnel MTU is defined as the Path MTU between the tunnel entry-point and the tunnel exit-point nodes minus the size of the tunnel header. In some cases, a single tunnel entry-point may connect to multiple tunnel exit-points, e.g. the IPv6 address of the tunnel exit-point is a multicast or an anycast address, or the tunnel entry-point works as an AFTR element [RFC6333].

Figure 1 shows an example that a tunnel entry-point is connecting to 2 tunnel exit-points. When the tunnel entry-point sends a tunnel packet of length 1500 to tunnel exit-point1, the packet is discarded by the intermediate router and the router sends an ICMPv6 "Packet Too Big"(PTB) message to tunnel entry-point with the MTU field equals 1280. After received the ICMPv6 PTB message, the tunnel entry-point sets its tunnel MTU to 1280-40=1240 according to section 6.7 of [RFC2473]. After that, when the tunnel entry-point sends tunnel packets to tunnel exit-point2, the tunnel packet size will be restricted to 1280. This is inefficient.

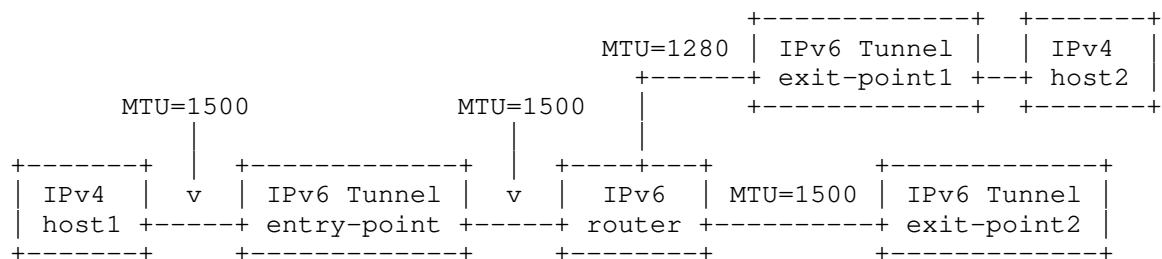


Figure 1: An Example of IPv6 Tunneling Scenario

## 5. Tunnel MTU Configuration

An IPv6 tunnel entry-point node SHOULD perform Path MTU discovery dynamically on the paths between the tunnel entry-point itself and each exit-point node it connects to. The PMTU information SHOULD be stored in a table indexed by the destination IPv6 address as is described in section 5.2 of [RFC1981]. When a packet enters the tunnel, the tunnel entry-point looks up the PMTU table for the corresponding PMTU value. If not found, it uses the MTU of IPv6 next-hop link as the default value. This PMTU value minus the size of the tunnel headers is set as the tunnel MTU value.

If the IPv6 tunnel entry-point node does not perform Path MTU discovery to decide its tunnel MTU, the network operator MUST estimate a safe MTU value and configure this value as the tunnel MTU. This safe MTU value should be no larger than the minimum Path MTU between the tunnel node and every potential tunnel exit-point, minus the size of tunnel headers. If the operator can not decide this value, the tunnel MTU SHOULD be set to 1280 minus the size of tunnel headers.

## 6. Security Considerations

TBD

## 7. IANA Considerations

This document does not include an IANA request.

## 8. References

### 8.1. Normative References

- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.
- [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.

### 8.2. Informative References

- [I-D.ietf-software-map]  
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S.,

Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-08 (work in progress), August 2013.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

#### Authors' Addresses

Yong Cui  
Tsinghua University  
Department of Computer Science, Tsinghua University  
Beijing 100084  
P.R.China

Phone: +86-10-6260-3059  
Email: yong@csnet1.cs.tsinghua.edu.cn

Cong Liu  
Tsinghua University  
Department of Computer Science, Tsinghua University  
Beijing 100084  
P.R.China

Phone: +86-10-6278-5822  
Email: gnocuil@gmail.com

Network Group  
INTERNET-DRAFT  
Intended status: Experimental

H. Rafiee  
Huawei Technologies Duesseldorf GmbH  
C. Meinel

Hasso Plattner Institute

Expires: March 19, 2015

September 19, 2014

A Simple Secure Addressing Scheme for IPv6 AutoConfiguration (SSAS)  
<draft-rafiee-6man-ssas-11.txt>

## Abstract

Since performance and security are, both, two important criteria for a mechanism to be widely used by different nodes with various resources, the purpose of this document is to propose a mechanism for local security and to prevent IP spoofing. This mechanism also consider user's privacy.

## 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 19, 2015.

## 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. Conventions used in this document . . . . .	4
3. Algorithms Overview . . . . .	4
3.1. Interface ID (IID) Generation . . . . .	4
3.1.1. Signature Generation . . . . .	5
3.1.2. Generation of NDP/SeND Messages . . . . .	6
3.1.2.1. SSAS signature data field . . . . .	6
3.1.3. SSAS verification process . . . . .	8
3.2. Resource Public key Infrastructure (RPKI) . . . . .	9
4. SSAS Applications . . . . .	9
4.1. A solution for all nodes . . . . .	9
4.2. Authentication in Network layer . . . . .	9
4.3. Authentication in Application Layer . . . . .	10
4.4. Other Applications . . . . .	10
5. Security Considerations . . . . .	10
6. IANA Considerations . . . . .	11
7. Privacy Consideration . . . . .	11
8. Appendix A . . . . .	11
8.1. Comparison of CGA and SSAS generation time . . . . .	11
9. Appendix B . . . . .	12
9.1. Network-based protection vs. Node-based protection . . . . .	12
10. Acknowledgements . . . . .	13
11. References . . . . .	13
11.1. Normative . . . . .	13
11.2. Informative . . . . .	14
Authors' Addresses . . . . .	16

## 1. Introduction

In IPv6 networks, nodes can use two different mechanisms to configure their IP addresses -- Neighbor Discovery Protocol (NDP) [RFC4861, RFC4862] and Dynamic Host Configuration Protocol (DHCPv6) [RFC3315]. Unfortunately none of these mechanisms are natively secure. So, they open the nodes with so many local security problems. There are several attacks possible in local network [RFC3756]. One example is IP spoofing that enable an attacker to forge the identity of a victim node, the other example is preventing the node from configuring its IP address.

The reasons that local security is important are as follows [localSecurity]:

- Not all the nodes on the local link are trusted: viruses or other malware can infect a legitimate node in the local link and turn it to an attacker.

- Attacker might be inside the network: The networks of big enterprises might be harmed by one of the staff that was recently fired.

There is currently a mechanism available to secure the NDP, i.e., Secure Neighbor Discovery (SeND) [RFC3971]. SeND does this protection by adding 4 options to NDP packets. Among these options, Cryptographically Generated Addresses (CGA) [RFC3972] is a very important option that provides the node with the proof of IP address ownership by finding a binding between the node's public key and its IP address. Unfortunately CGA has some problems that are listed as follows:

- CGA sec value problem: This problem is explained in [cgaattack] and addressed in [cgabis].

- CGA increases complexity and decreases performance: CGA uses sec value (the value between 0 to 7) and claims to complicate the brute force attacks. (However it is not true based on [cgaattack]) If CGA sec value higher than 0 is in use, then this will reduce the performance because CGA algorithm needs to repeat some steps and it needs the high attention of the CPU and makes the CPU busy. So, CGA sec value higher than 0, consumes more energy than other nodes that do not use CGA. Today, the demands on multi-functioning smaller devices are increasing but unfortunately the battery technology is not as advanced as expected. So, the use of CGA algorithm that needs to use higher level of energy is not ideal for these types of nodes and the use of CGA sec value zero does not protect the node as expected. (Please refer to appendix A for more information)

- CGA might cause privacy issue: Since the generation of CGA higher sec values might take time. The nodes might not be willing to change its IP address and keep this address as long as the subnet prefix is

valid. If the node is a fixed node in the network, then it will be vulnerable to node tracking. The node might also not change the CGA address when it visits a new network or it might not generate any new key pairs. In other word, it might use the same CGA parameters (excluding prefix) as used in the old network and thus it will be vulnerable to node tracking.

- Packet size

CGA uses RSA as a default key pair generation algorithm. This is why, if SeND with CGA option is in use to secure NDP messages, the minimum packet size needs to carry this public key for CGA nodes is 460 bytes. Packet size also reduces the performance and causes delays in the network.

Since privacy and security are, both, very important issues in everyday life, the purpose of this document is to offer an alternative and simple addressing mechanism to generate an interface ID (IID) which provides the node with both security and privacy while does not sacrifice the performance, and tries to decrease the packet size as much as possible.

## 2. 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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document the use of || indicates the concatenation of the values on either side of the sign.

## 3. Algorithms Overview

As explained earlier, one of the problems with using the current IID generation approach is the intensive computer processing that is needed for the IID algorithm generation. Another concern is for the lack of security (if CGA is not in use). This is what this document intends to address.

### 3.1. Interface ID (IID) Generation

To generate the IID a node will need to execute the following steps.

1. Generate key pairs (public/private keys) using one of the latest version of ECC algorithm [RFC6090] or other fastest short key size algorithms. . The implementations SHOULD be updated with any new

version of ECC algorithm when ECC current version is no longer secure. ECC is the default algorithm, but any algorithm capable of generating a small key size in a short amount of time is viable. The node then uses this new value for the generation of the IP address and signature. Comparing the use of ECC to that of RSA shows that an ECC with a 192 bit key is equivalent to a RSA with a 7680 bit key (according to US National Security Agency) In this case the packet size would be decreased by a factor 11 times smaller than that when using RSA.

Note 1: The node MUST not generate the weak key. For ECC, the node MUST not use ECC key size lower than 192 bits. If any nodes used a weak key size, then the other nodes MUST discard receiving the message from that node. If in future, key size 192 bits is considered as a weak key size, the default key size value MUST be changed to the next strong key size.

2. Execute a hash function on the public key. The default hash function is SHA256. If in future, this hash function is no longer secure, the node MUST use the next strong hash function.

3. Take the first 64bits of the digest and call it IID. In case collision count is higher than 1, then depends on the number, takes second 64 bits or third 64 bits of this hash value.

It is not RECOMMENDED to use this algorithm in case IID is less than 64 bits [variableprefix]. A node MUST obtain the prefix length information from router advertisement messages.

4. Concatenate the IID with the local subnet prefix to set the link local IP address.

5. Concatenate the IID with the router subnet prefix (Global subnet prefix), obtained from the Router Advertisement (RA) message, and set it as a tentative public IP address. This IP address will become permanent after Duplicate Address Detection (DAD) processing. (For more information about DAD refer to section 3.1.2.)

Note 1: In this document bits u and g does not have any particular meaning and is used as a part of public key. This assumption is by the clarification of using these bits in [RFC7136].

### 3.1.1. Signature Generation

SSAS is not dependent to SeND but it can be used as a new option of SeND. When SSAS is used as an option of SeND, SSAS signature can be placed as a RSA signature in SeND. If SSAS is used alone, this section MUST be included in SSAS data structure. This proves that SSAS is compatible to use with SeND.

The SSAS signature is added to NDP messages in order to protect them from IP spoofing and spoofing types of attack. SSAS will provide

proof of IP address ownership. To generate the SSAS signature, the node needs to execute the following steps:

1. Concatenate the timestamp with the MAC address, collision count, algorithm type and the global (public) IP address. (see figure 1)

timestamp 8 bytes	Mac address 6 bytes	Collision Count 3 bits	Algorithm type 1 byte
Global IP address 16 bytes		Other Options variable	

Figure 1 SSAS Signature format

2. Sign the resulting value from step 1, using the ECC private key (or any other short key size algorithm), and call the resulting output the SSAS signature.

If NDP messages contain other data that must be protected, such as important routing information, then this data SHOULD also be included in the signature. The signature is designed for the inclusion of any data needing protection. If there is no data that needs protection, then the signature will only contain the timestamp, MAC address, Collision count and Global IP address (Router subnet prefix plus IID).

### 3.1.2. Generation of NDP/SeND Messages

After a node generates its IP address, it should then process Duplicate Address Detection in order to avoid address collisions in the network. In order to do this the node needs to generate a Neighbor Solicitation (NS) message. The SSAS signature is added to the ICMPv6 options of NS messages. The SSAS signature data field is an extended version of the standard format of the RSA signature option of SeND [RFC3971]. The timestamp option is the same as that used with SeND. In the SSAS signature, the data field contains the following items: type, length, reserved, Other Len, algorithm type, collision count, subnet prefix, other option and padding.

#### 3.1.2.1. SSAS signature data field

Type 1 byte	Length 1 byte	Reserved 2 bytes	Other len 1 byte
Algorithm type 1 byte	Collision count 3 bits	Subnet prefix 8bytes	Other Options
Hash Function	Response No.	SSAS Signature	

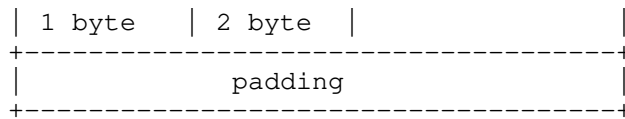


Figure 2 NDP Message Format with SSAS Signature Data Field

- Type: This option is set to 15. This is the sequential number used in SeND to indicate a SSAS data field.

- Length: The length of the Signature Data field, including the Type, Length, Reserved, Algorithm type, Signature and padding, must be a multiple of eight.

- Reserved: A 2 byte field reserved for future use. The value must be initialized to zero by the sender and should be ignored by the receiver.

- Other Len: The length of other options in multiples of eight. The length of this field is 1 byte.

- Algorithm type: The algorithm used to generate key pairs and sign the message. The length of this field is 1 byte. For ECC, this value is 0. Future algorithms will start at one and increase from there.

- Collision count: When a collision occurs during the DAD, the node will increment this value and store it in a file to be included in the sent packets for as long as the current IP address is valid. This value indicates to the node where it needs to start its check from, i.e., the first or second or third 64 bytes from the start of the hash value (digest) array of the public key.

- Subnet Prefix: This is the router subnet prefix.

- Hash Function: A hash function used to generate IID. The length of this field is 1 byte. For SHA256, this value is 0. Future algorithms will start at one and increase from there.

- Response No: This is similar to nonce but by the use of different mechanism. This value is not random and it is a copy of timestamp. In sender's message, this value MUST be set to zero and in response message (sent from a receiver node), this value MUST be set to the timestamp of the sender's message. The length of this field is 2 bytes. The sender node should cache this value in order to compare it with all responses sent by other nodes. This informs the sender node that the message is the response to his message and protects the node against replay attack.

- Other Options. This variable-length field contains important data that needs to be protected in the packet. The padding is used to insure that the field is a multiple of eight in length.

- Padding. A variable-length field containing padding to insure that the entire signature field is a multiple of eight in length. It thus contains the number of blanks needed to make the entire signature

field end on a multiple of eight.

All NDP messages (except RS messages) SHOULD contain the SSAS signature data field which allows receivers to verify senders. If a node receives a solicited NA message in response to its NS message showing that another node claims to own this address, then, after a successful verification process, this node increments the collision count by one and this value is used as explained in the "Collision count" item above. It will start from that section of the public key for the generation of a new IP address. The node repeats this 3 times and after 3 times generates a new public/private keys. Since the likelihood of two nodes having the same value is  $1/(2^{63})$ . This is really a small value while we also considered the order of magnitude relative to roughly 2 power 64 against sloppy implementations.

### 3.1.3. SSAS verification process

A node's verification process should start when it receives NDP messages. Following are the steps used in the verification process:

1. Obtain Response No from the sender's packet. Compare this value with its own timestamp that used in its previous message. If it is the same go to the next step, otherwise discard the message. (If SSAS is a part of SeND, this step should be skipped.)
2. Obtain prefix information from its own cache or from a router advertisement to make sure about the prefix sizes and number of bits used for IID.
3. Obtain the timestamp from the NDP message and call this value t1.
4. Obtain the timestamp from the node's system, convert it to UTC, and call this value t2.
5. If  $(t2 - x) \leq t1 \leq (t2 + x)$  go to step 6. Otherwise, the message SHOULD be discarded without further processing. The value of x is dependent on network delays and network policy. The default value would be the value of Round Trip Time (RTT). The implementations SHOULD allow to set different values.
6. Obtain the public key from its own neighboring cache. If no matches are found in the node cache and if there is a centralized RPKI model available in the local network, then the node MIGHT obtain this public key from that node. Otherwise go to the next step.
7. Compare this to its own public key. If it is not the same, go to the next step. Otherwise, the message should be discarded without further processing. (This step should be skipped when the node uses the RPKI to obtain the other nodes' public key.)
8. Obtain the hash algorithm from the packet. By default it is SHA256.

9. Execute hash function on the public key. Takes 64bits, depends on collision count, from the hash function. Compare this value with the node's IID source IP. If it is the same, go to the next step. Otherwise, discard the message without further processing.

10. Concatenate the timestamp with the MAC address, algorithm type, collision count, sender's Global IP address (subnet prefix and IID), and other options (if any) and call this entity the plain message.

11. Obtain the SSAS signature from the SSAS signature data field. Obtain the Algorithm type from the message.

12. Verify the Signature using the public key and then enter the plain message and the SSAS signature as an input to the verification function. If the verification process is successful, process the message. Otherwise, the message should be discarded without further processing.

After a successful verification, the node SHOULD store the public key and MAC address of the sender node in its neighboring cache. By default, the cache is valid for two days but the implementation SHOULD consider a way to let the end users change this default value.

### 3.2. Resource Public key Infrastructure (RPKI)

To Authorize the Routers in the network and increase the security of the nodes in this network, it is recommended to use an RPKI explained in RFC 6494 and 6495. It is explained in more detail in [SSASAnalysis] and local security deployment [localSecurity].

## 4. SSAS Applications

### 4.1. A solution for all nodes

SSAS is capable to be used in standard nodes (standard computers) and nodes where limited computational resources are available. One example is the use of SSAS in sensor networks. Sensor networks are a prime example of nodes with limited resources (such as battery, CPU, and etc); see RFC 4919 [RFC4919] for use in IPv6 networks. Because currently, as explained in section 4. RFC 6775, the generation of the IID is based on EUI-64 which makes these nodes vulnerable to privacy and security attacks. One of these types of attack can occur during the Duplicate Address Detection (DAD) process.

### 4.2. Authentication in Network layer

Another example for the use of SSAS would be in mobile networks during the generation of IP addresses, as explained in section 4.4



RFC 6275 [RFC6275]. The current problem with the addressing mechanism in a mobile node is that no privacy is observed when a node moves to another network while usually keeping its Home Address. If there were a fast and secure mechanism available, then it would be possible to set this Home Address and change it and re-register it to the Home network. Another possible use for SSAS in mobile nodes could be as a security mechanism during the configuration of Care of Address (CoA); see section 3. RFC 5213 [RFC5213]. In that RFC, home proxy plays the role of a home agent for mobile nodes and mobile nodes set their CoA by the use of either stateful or stateless autoconfiguration. Currently they MUST use IPsec in order to secure this process. Section 4 of that RFC discusses the possibility of using another algorithm in order to secure mobile nodes.

#### 4.3. Authentication in Application Layer

SSAS can be used as a means of authentication for the nodes in application layer. It is really important that the nodes know who they are talking to. This is because a user uses an application to connect to another node on the internet. This application either uses a domain name of the destination node (that later translates to the IP addresses) or directly uses the IP address of this node. This is where the attacker can play a role and spoof this IP address and play a MITM attack or other types of attacks. If the node uses this approach, the attacker does not have a possibility to spoof the IP address of the communicating node. So, this approach can mitigate IP spoofing during the authentication of two nodes in application layer.

#### 4.4. Other Applications

With the wide usage of IP addresses in different types of devices and by the use of autoconfiguration mechanisms to configure these IP addresses, the need for the use of a security algorithm is increased. One type of application would be for use in vehicular networks or in the car-to-car networks. There is currently some work in progress that makes use of Neighbor Discovery. SSAS could also be a solution for enabling fast protection against ND attacks.

#### 5. Security Considerations

There are two security considerations:

Since SSAS cannot prevent the layer 2 attacks but can mitigate it after the first verification, therefore one would need to use a monitoring device to prevent MAC spoofing. The other possibility is to have a dynamic MAC address. This means the SSAS node can use the 48 rightmost bits of the its public key as a MAC address. In this case there is a binding between the IP address, MAC address and public key. Since the verification process would have failed, it cannot be spoofed. However, this approach might be problematic from an operational view and might need to have some consideration before

being used.

Another security consideration is how to attack SSAS. One might ask oneself that what are the odds of an attacker being able to generate a public key having two four sequential bytes (from two different halves of public key) that are the same as 64 bits of that in Interface ID? If he could, he could then generate the signature using his own private key and thus break SSAS. Mathematically it has been shown that the likelihood of matching 64 bits in the public key against 64bits in the IID is  $1/(2^{64})$ . in [SSASAnalysis] the analysis of SSAS is explained and compared to CGA. Since the nodes in the network need to keep the public key and the MAC address of other nodes in the cache, the attacker only has a few seconds to perform this attack and then the attacker needs to perform this attack against the whole public key. For CGA, this value is less. in [cgaattack], the attack in CGA was explained. So, in general, SSAS is faster and in a good security level. In other word, SSAS tried to address the security and performance problem exists in CGA and offer a fastest algorithm.

## 6. IANA Considerations

There is no IANA consideration

## 7. Privacy Consideration

When an attacker is inside a local link, he is enable to identify a node. although, this target node changes its IP address. The reason is because the target node does not change its MAC address. However, if the public key needs to be used for verification in other mechanisms and not in local link, then it is RECOMMENDED that the public/private keys to be valid for a short period of time. The default value would be a week. The implementations need to consider the automatic key generation to avoid administrative requirements for this process.

## 8. Appendix A

### 8.1. Comparison of CGA and SSAS generation time

The following information was retrieved from [cgatime]. It shows the time required to generate CGA in different sec value. This is why, in practice, only sec value 0 and 1 can be used.

sec value 1 => ~ 1 second

sec value 2 => ~ 3 hours

sec value 3 => ~ 24 years

sec value 4 => ~  $1.16 \times 10^6$  years

sec value 5 => ~  $1 \times 10^{11}$  years

sec value 6 => ~  $6.8 \times 10^{15}$  years

The above information is based on the fact that one uses RSA key sizes less than 1280 bits. If one needs to use the higher security, then it needs more time for the generation of CGA value. Using RSA higher key sizes also increases the packet size needs to carry the public key. Here is our evaluation of ECC and RSA key generation time in a standard computer with 2.6 GHz CPU.

SSAS generation time is about the time needed to generate key pairs. Since, by default, SSAS uses ECC 192 bits, the following values compares ECC with RSA. RSA is the algorithm uses in CGA. As explained earlier, the security of ECC 192 bits is equivalent to the security of RSA 7680 bits.

ECC 192 bits: Average key generation time = 195011 microseconds

RSA 1280 bits: Average key generation time = 681039 microseconds

RSA 7680 bits: Average key generation time = 163473350 microseconds

## 9. Appendix B

### 9.1. Network-based protection vs. Node-based protection

Node-based protection is the ability of the node to protect against some types of attacks such as IP spoofing, MITM attack. On the other hand, network-based protection is the use of some devices in the network edges to protect the nodes inside this network against router advertisement spoofing attacks or other types of attack. Both of these protection is required and both can complement each other. This is because the attacker might be inside the network and play a role of MITM, spoof the other nodes' IP address, prevent other nodes from configuring their IP address and cause many delays and problems in the local network (Not all the nodes in the network is ever trustee). One important consideration about node-based protection is that, it should support any node and apply to any nodes (Including nodes with limited energy resources or limited memory resources). This is why there is a need for a good mechanism to provide this protection with less cost. The proposed mechanism in this document, i.e., SSAS can provide the node with node-based protection. With only node-based protection, the malicious node inside this network can claim to be a router and the node does not have any means to authorize him. This is

why, the network-based protection is also the complement solution to a node-based protection. There are some approaches to provide the node with network-based protection. One such approach might be RA-gaurd [RAgaurd] which limits subnet prefixes. Unfortunately with this approach, still the node inside this network can maliciously claim to be a router and play the MITM attack inside the network by sending unicast router advertisement messages. So, the attack is still possible. The other approach is the use of RPKI as explained in RFC 6494 and RFC 6495. Unfortunately these RFCs only explain the possibility of using them but not the detail of implementation. The detail implementation is explained in [SSASAnalysis]. The local RPKI node also can play a role of monitoring device in the network.

## 10. Acknowledgements

The Authors would like to acknowledge Erik Nordmard and Joel M. Halpern for their supports and assistance to improve this document. The authors also would like to acknowledge Michael Richardson, Dan Wing, Tim Chown, Christian Huitema, Joel M. Halpern for their comments to improve this document

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4291] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture," RFC 4291, February 2006.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4941] Narten, T., Draves, R., Krishnan, S., "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and Nikander, P., "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3756] Nikander, P., Kempf, J., Nordmark, E., "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3972, May 2004.
- [RFC4919] Kushalnagar, N., Montenegro, G., Schumacher, C., "IPv6 over Low-Power Wireless Personal Area Networks

(6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., Bormann, C. , " Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[RFC6275] Perkins, C., Johnson, D., Arkko, J., "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6543] Gundavell, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, May 2012.

[RFC6090] McGrew, D., Igoe, K., Salter, M., "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2012.

[RFC3756] Nikander, F., Kempf, J., Nordmark, E., "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[RFC7136] Carpenter, B., Jiang, S., "Significance of IPv6 Interface Identifiers", RFC 7136, 2013

## 11.2. Informative References

[SSASAnalysis] Rafiee, H., Meinel, C., "'SSAS: a Simple Secure Addressing Scheme for IPv6 AutoConfiguration". In Proceedings of the 11th IEEE International conference on Privacy, Security and Trust (PST), IEEE Catalog number: CFP1304F-ART, ISBN: 978-1-4673-5839-2.

[cgaattack] Rafiee, H., Meinel, C., "Possible Attack on Cryptographically Generated Addresses (CGA)" ,<http://tools.ietf.org/html/draft-rafiee-6man-cga-attack>, 2014

[RAgaurd] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", <http://tools.ietf.org/html/draft-ietf-v6ops-ra-guard-implementation>, 2012

[localSecurity] Rafiee, H., Meinel, C., "Recommendations for Local Security Deployments" ,<http://tools.ietf.org/html/draft-rafiee-6man-local-security>, 2013

[cgatime] Bos, J., Oezen, O., Hubaux, J., "Analysis and Optimization of Cryptographically Generated Addresses", In Proceedings of the 12th International conference on Information Security (2009), ACM, pp. 17 ? 32.

[variableprefix] Carpenter, B., Chown, T, Gont, F.,  
Jiang, S., Petrescu, A., Yourtchenko, A., " Analysis  
of the 64-bit Boundary in IPv6 Addressing",  
<http://tools.ietf.org/html/draft-ietf-6man-why64> ,  
April 2014

[cgabis] Rafiee,H., Zhang, D., "CGA Security Improvement"  
,<http://tools.ietf.org/html/draft-rafiiee-rfc3972-bis>, 2014

Authors' Addresses

Hosnieh Rafiee  
HUAWEI TECHNOLOGIES Duesseldorf GmbH  
Riesstrasse 25, 80992,  
Munich, Germany  
Phone: +49 (0)162 204 74 58  
Email: hosnieh.rafiiee@huawei.com

Christoph Meinel  
Hasso-Plattner-Institute  
Prof.-Dr.-Helmert-Str. 2-3  
Potsdam, Germany  
Email: meinel@hpi.uni-potsdam.de





Network Working Group  
Internet-Draft  
Obsoletes: rfc5320 (if approved)  
Updates: rfc2460 (if approved)  
Intended status: Standards Track  
Expires: April 24, 2014

F. Templin, Ed.  
Boeing Research & Technology  
October 21, 2013

The Subnetwork Encapsulation and Adaptation Layer (SEAL)  
draft-templin-intarea-seal-65.txt

## Abstract

This document specifies a Subnetwork Encapsulation and Adaptation Layer (SEAL). SEAL operates over virtual topologies configured over connected IP network routing regions bounded by encapsulating border nodes. These virtual topologies are manifested by tunnels that may span multiple IP and/or sub-IP layer forwarding hops, where they may incur packet duplication, packet reordering, source address spoofing and traversal of links with diverse Maximum Transmission Units (MTUs). SEAL addresses these issues through the encapsulation and messaging mechanisms specified in this document.

## 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 24, 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.

## Table of Contents

1. Introduction . . . . .	4
1.1. Motivation . . . . .	4
1.2. Approach . . . . .	6
1.3. Differences with RFC5320 . . . . .	7
2. Terminology . . . . .	8
3. Requirements . . . . .	10
4. Applicability Statement . . . . .	10
5. SEAL Specification . . . . .	11
5.1. SEAL Tunnel Model . . . . .	11
5.2. SEAL Model of Operation . . . . .	12
5.3. SEAL Encapsulation Format . . . . .	14
5.4. ITE Specification . . . . .	16
5.4.1. Tunnel MTU . . . . .	16
5.4.2. Tunnel Neighbor Soft State . . . . .	17
5.4.3. SEAL Layer Pre-Processing . . . . .	18
5.4.4. SEAL Encapsulation and Segmentation . . . . .	19
5.4.5. Outer Encapsulation . . . . .	21
5.4.6. Path Probing and ETE Reachability Verification . . . . .	21
5.4.7. Processing ICMP Messages . . . . .	22
5.4.8. IPv4 Middlebox Reassembly Testing . . . . .	24
5.4.9. Stateful MTU Determination . . . . .	25
5.4.10. Detecting Path MTU Changes . . . . .	25
5.5. ETE Specification . . . . .	25
5.5.1. Reassembly Buffer Requirements . . . . .	25
5.5.2. Tunnel Neighbor Soft State . . . . .	26
5.5.3. IP-Layer Reassembly . . . . .	26
5.5.4. Decapsulation, SEAL-Layer Reassembly, and Re-Encapsulation . . . . .	27
5.6. The SEAL Control Message Protocol (SCMP) . . . . .	28
5.6.1. Generating SCMP Messages . . . . .	29
5.6.2. Processing SCMP Messages . . . . .	31
6. Link Requirements . . . . .	33
7. End System Requirements . . . . .	33
8. Router Requirements . . . . .	33
9. Nested Encapsulation Considerations . . . . .	34
10. Reliability Considerations . . . . .	34
11. Integrity Considerations . . . . .	34
12. IANA Considerations . . . . .	35
13. Security Considerations . . . . .	35
14. Related Work . . . . .	36
15. Implementation Status . . . . .	36
16. Acknowledgments . . . . .	37
17. References . . . . .	37
17.1. Normative References . . . . .	37
17.2. Informative References . . . . .	38
Author's Address . . . . .	42

## 1. Introduction

As Internet technology and communication has grown and matured, many techniques have developed that use virtual topologies (manifested by tunnels of one form or another) over an actual network that supports the Internet Protocol (IP) [RFC0791][RFC2460]. Those virtual topologies have elements that appear as one network layer hop, but are actually multiple IP or sub-IP layer hops. These multiple hops often have quite diverse properties that are often not even visible to the endpoints of the virtual hop. This introduces failure modes that are not dealt with well in current approaches.

The use of IP encapsulation (also known as "tunneling") has long been considered as the means for creating such virtual topologies (e.g., see [RFC2003][RFC2473]). Tunnels serve a wide variety of purposes, including mobility, security, routing control, traffic engineering, multihoming, etc., and will remain an integral part of the architecture moving forward. However, the encapsulation headers often include insufficiently provisioned per-packet identification values. IP encapsulation also allows an attacker to produce encapsulated packets with spoofed source addresses even if the source address in the encapsulating header cannot be spoofed. A denial-of-service vector that is not possible in non-tunneled subnetworks is therefore presented.

Additionally, the insertion of an outer IP header reduces the effective path MTU visible to the inner network layer. When IPv6 is used as the encapsulation protocol, original sources expect to be informed of the MTU limitation through IPv6 Path MTU discovery (PMTUD) [RFC1981]. When IPv4 is used, this reduced MTU can be accommodated through the use of IPv4 fragmentation, but unmitigated in-the-network fragmentation has been found to be harmful through operational experience and studies conducted over the course of many years [FRAG][FOLK][RFC4963]. Additionally, classical IPv4 PMTUD [RFC1191] has known operational issues that are exacerbated by in-the-network tunnels [RFC2923][RFC4459].

The following subsections present further details on the motivation and approach for addressing these issues.

### 1.1. Motivation

Before discussing the approach, it is necessary to first understand the problems. In both the Internet and private-use networks today, IP is ubiquitously deployed as the Layer 3 protocol. The primary functions of IP are to provide for routing, addressing, and a fragmentation and reassembly capability used to accommodate links with diverse MTUs. While it is well known that the IP address space

is rapidly becoming depleted, there is also a growing awareness that other IP protocol limitations have already or may soon become problematic.

First, the Internet historically provided no means for discerning whether the source addresses of IP packets are authentic. This shortcoming is being addressed more and more through the deployment of site border router ingress filters [RFC2827], however the use of encapsulation provides a vector for an attacker to circumvent filtering for the encapsulated packet even if filtering is correctly applied to the encapsulation header. Secondly, the IP header does not include a well-behaved identification value unless the source has included a fragment header for IPv6 or unless the source permits fragmentation for IPv4. These limitations preclude an efficient means for routers to detect duplicate packets and packets that have been re-ordered within the subnetwork. Additionally, recent studies have shown that the arrival of fragments at high data rates can cause denial-of-service (DoS) attacks on performance-sensitive networking gear, prompting some administrators to configure their equipment to drop fragments unconditionally [I-D.taylor-v6ops-fragdrop].

For IPv4 encapsulation, when fragmentation is permitted the header includes a 16-bit Identification field, meaning that at most  $2^{16}$  unique packets with the same (source, destination, protocol)-tuple can be active in the network at the same time [RFC6864]. (When middleboxes such as Network Address Translators (NATs) re-write the Identification field to random values, the number of unique packets is even further reduced.) Due to the escalating deployment of high-speed links, however, these numbers have become too small by several orders of magnitude for high data rate packet sources such as tunnel endpoints [RFC4963].

Furthermore, there are many well-known limitations pertaining to IPv4 fragmentation and reassembly - even to the point that it has been deemed "harmful" in both classic and modern-day studies (see above). In particular, IPv4 fragmentation raises issues ranging from minor annoyances (e.g., in-the-network router fragmentation [RFC1981]) to the potential for major integrity issues (e.g., mis-association of the fragments of multiple IP packets during reassembly [RFC4963]).

As a result of these perceived limitations, a fragmentation-avoiding technique for discovering the MTU of the forward path from a source to a destination node was devised through the deliberations of the Path MTU Discovery Working Group (MTUDWG) during the late 1980's through early 1990's which resulted in the publication of [RFC1191]. In this negative feedback-based method, the source node provides explicit instructions to routers in the path to discard the packet and return an ICMP error message if an MTU restriction is

encountered. However, this approach has several serious shortcomings that lead to an overall "brittleness" [RFC2923].

In particular, site border routers in the Internet have been known to discard ICMP error messages coming from the outside world. This is due in large part to the fact that malicious spoofing of error messages in the Internet is trivial since there is no way to authenticate the source of the messages [RFC5927]. Furthermore, when a source node that requires ICMP error message feedback when a packet is dropped due to an MTU restriction does not receive the messages, a path MTU-related black hole occurs. This means that the source will continue to send packets that are too large and never receive an indication from the network that they are being discarded. This behavior has been confirmed through documented studies showing clear evidence of PMTUD failures for both IPv4 and IPv6 in the Internet today [TBIT] [WAND] [SIGCOMM] [RIPE].

The issues with both IP fragmentation and this "classical" PMTUD method are exacerbated further when IP tunneling is used [RFC4459]. For example, an ingress tunnel endpoint (ITE) may be required to forward encapsulated packets into the subnetwork on behalf of hundreds, thousands, or even more original sources. If the ITE allows IP fragmentation on the encapsulated packets, persistent fragmentation could lead to undetected data corruption due to Identification field wrapping and/or reassembly congestion at the ETE. If the ITE instead uses classical IP PMTUD it must rely on ICMP error messages coming from the subnetwork that may be suspect, subject to loss due to filtering middleboxes, or insufficiently provisioned for translation into error messages to be returned to the original sources.

Although recent works have led to the development of a positive feedback-based end-to-end MTU determination scheme [RFC4821], they do not excuse tunnels from accounting for the encapsulation overhead they add to packets. Moreover, in current practice existing tunneling protocols mask the MTU issues by selecting a "lowest common denominator" MTU that may be much smaller than necessary for most paths and difficult to change at a later date. Therefore, a new approach to accommodate tunnels over links with diverse MTUs is necessary.

## 1.2. Approach

This document concerns subnetworks manifested through a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes. Example connected network routing regions include Mobile Ad hoc Networks (MANETs), enterprise networks and the global public Internet itself. Subnetwork border

nodes forward unicast and multicast packets over the virtual topology across multiple IP and/or sub-IP layer forwarding hops that may introduce packet duplication and/or traverse links with diverse Maximum Transmission Units (MTUs).

This document introduces a Subnetwork Encapsulation and Adaptation Layer (SEAL) for tunneling inner network layer protocol packets over IP subnetworks that connect Ingress and Egress Tunnel Endpoints (ITEs/ETEs) of border nodes. It provides a modular specification designed to be tailored to specific associated tunneling protocols. (A transport-mode of operation is also possible but out of scope for this document.)

SEAL provides a mid-layer encapsulation that accommodates links with diverse MTUs, and allows routers in the subnetwork to perform efficient duplicate packet and packet reordering detection. The encapsulation further ensures message origin authentication, packet header integrity and anti-replay in environments in which these functions are necessary.

SEAL treats tunnels that traverse the subnetwork as ordinary links that must support network layer services. Moreover, SEAL provides dynamic mechanisms (including limited segmentation and reassembly) to ensure a maximal path MTU over the tunnel. This is in contrast to static approaches which avoid MTU issues by selecting a lowest common denominator MTU value that may be overly conservative for the vast majority of tunnel paths and difficult to change even when larger MTUs become available.

### 1.3. Differences with RFC5320

This specification of SEAL is descended from an experimental independent RFC publication of the same name [RFC5320]. However, this specification introduces a number of fundamental differences from the earlier publication. This specification therefore obsoletes (i.e., and does not update) [RFC5320].

First, this specification includes a protocol version field in the SEAL header whereas [RFC5320] does not, and therefore cannot be updated by future revisions. Secondly, [RFC5320] forms a 32-bit Identification value by concatenating the 16-bit IPv4 Identification field with a 16-bit Identification "extension" field in the SEAL header. This means that [RFC5320] can only operate over IPv4 networks (since IPv6 headers do not include a 16-bit version number) and that the SEAL Identification value can be corrupted if the Identification in the outer IPv4 header is rewritten. In contrast, this specification includes a 32-bit Identification value that is independent of any identification fields found in the inner or outer

IP headers, and is therefore compatible with any inner and outer IP protocol version combinations.

Additionally, the SEAL segmentation and reassembly procedures defined in [RFC5320] differ significantly from those found in this specification. In particular, this specification defines an 13-bit Offset field that allows for finer-grained segment sizes when SEAL segmentation is necessary. In contrast, [RFC5320] includes only a 3-bit Segment field and performs reassembly through concatenation of consecutive segments.

This version of SEAL also includes an optional Integrity Check Vector (ICV) that can be used to digitally sign the SEAL header and the leading portion of the encapsulated inner packet. This allows for a lightweight integrity check and a loose message origin authentication capability. The header further includes new control bits as well as a link identification field for additional control capabilities.

Finally, this version of SEAL includes a new messaging protocol known as the SEAL Control Message Protocol (SCMP), whereas [RFC5320] performs signalling through the use of SEAL-encapsulated ICMP messages. The use of SCMP allows SEAL-specific departures from ICMP, as well as a control messaging capability that extends to other specifications, including Virtual Enterprise Traversal (VET) [I-D.templin-intarea-vet].

## 2. Terminology

The following terms are defined within the scope of this document:

### subnetwork

a virtual topology configured over a connected network routing region and bounded by encapsulating border nodes.

### IP

used to generically refer to either Internet Protocol (IP) version, i.e., IPv4 or IPv6.

### Ingress Tunnel Endpoint (ITE)

a portal over which an encapsulating border node (host or router) sends encapsulated packets into the subnetwork.

### Egress Tunnel Endpoint (ETE)

a portal over which an encapsulating border node (host or router) receives encapsulated packets from the subnetwork.



**SEAL Path**

a subnetwork path from an ITE to an ETE beginning with an underlying link of the ITE as the first hop. Note that, if the ITE's interface connection to the underlying link assigns multiple IP addresses, each address represents a separate SEAL path.

**inner packet**

an unencapsulated network layer protocol packet (e.g., IPv4 [RFC0791], OSI/CLNP [RFC0994], IPv6 [RFC2460], etc.) before any outer encapsulations are added. Internet protocol numbers that identify inner packets are found in the IANA Internet Protocol registry [RFC3232]. SEAL protocol packets that incur an additional layer of SEAL encapsulation are also considered inner packets.

**outer IP packet**

a packet resulting from adding an outer IP header (and possibly other outer headers) to a SEAL-encapsulated inner packet.

**packet-in-error**

the leading portion of an invoking data packet encapsulated in the body of an error control message (e.g., an ICMPv4 [RFC0792] error message, an ICMPv6 [RFC4443] error message, etc.).

**Packet Too Big (PTB) message**

a control plane message indicating an MTU restriction (e.g., an ICMPv6 "Packet Too Big" message [RFC4443], an ICMPv4 "Fragmentation Needed" message [RFC0792], etc.).

**Don't Fragment (DF) bit**

a bit that indicates whether the packet may be fragmented by the network. The DF bit is explicitly included in the IPv4 header [RFC0791] and may be set to '0' to allow fragmentation or '1' to disallow further in-network fragmentation. The bit is absent from the IPv6 header [RFC2460], but implicitly set to '1' because fragmentation can occur only at IPv6 sources.

The following abbreviations correspond to terms used within this document and/or elsewhere in common Internetworking nomenclature:

HLEN - the length of the SEAL header plus outer headers

ICV - Integrity Check Vector

MAC - Message Authentication Code

MTU - Maximum Transmission Unit

SCMP - the SEAL Control Message Protocol

SDU - SCMP Destination Unreachable message

SPP - SCMP Parameter Problem message

SPTB - SCMP Packet Too Big message

SEAL - Subnetwork Encapsulation and Adaptation Layer

TE - Tunnel Endpoint (i.e., either ingress or egress)

VET - Virtual Enterprise Traversal

### 3. Requirements

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]. When used in lower case (e.g., must, must not, etc.), these words MUST NOT be interpreted as described in [RFC2119], but are rather interpreted as they would be in common English.

### 4. Applicability Statement

SEAL was originally motivated by the specific case of subnetwork abstraction for Mobile Ad hoc Networks (MANETs), however the domain of applicability also extends to subnetwork abstractions over enterprise networks, mobile networks, ISP networks, SO/HO networks, the global public Internet itself, and any other connected network routing region.

SEAL provides a network sublayer for encapsulation of an inner network layer packet within outer encapsulating headers. SEAL can also be used as a sublayer within a transport layer protocol data payload, where transport layer encapsulation is typically used for Network Address Translator (NAT) traversal as well as operation over subnetworks that give preferential treatment to certain "core" Internet protocols, e.g., TCP, UDP, etc. (However, note that TCP encapsulation may not be appropriate for all use cases; particularly those that require low delay and/or delay variance.) The SEAL header is processed in the same manner as for IPv6 extension headers, i.e., it is not part of the outer IP header but rather allows for the creation of an arbitrarily extensible chain of headers in the same way that IPv6 does.

To accommodate MTU diversity, the Ingress Tunnel Endpoint (ITE) may need to perform limited segmentation which the Egress Tunnel Endpoint (ETE) reassembles. The ETE further acts as a passive observer that informs the ITE of any packet size limitations. This allows the ITE to return appropriate PMTUD feedback even if the network path between the ITE and ETE filters ICMP messages.

SEAL further provides mechanisms to ensure message origin authentication, packet header integrity, and anti-replay. The SEAL framework is therefore similar to the IP Security (IPsec) Authentication Header (AH) [RFC4301][RFC4302], however it provides only minimal hop-by-hop authenticating services while leaving full data integrity, authentication and confidentiality services as an end-to-end consideration.

In many aspects, SEAL also very closely resembles the Generic Routing Encapsulation (GRE) framework [RFC1701]. SEAL can therefore be applied in the same use cases that are traditionally addressed by GRE, but goes beyond GRE to also provide additional capabilities (e.g., path MTU accommodation, message origin authentication, etc.) as described in this document. The SEAL header is also exactly analogous to the IPv6 Fragment Header, and in fact shares the same format. SEAL can therefore re-use most existing code that implements IPv6 fragmentation and reassembly.

In practice, SEAL is typically used as an encapsulation sublayer in conjunction with existing tunnel types such as IPsec, GRE, IP-in-IPv6 [RFC2473], IP-in-IPv4 [RFC4213][RFC2003], etc. When used with existing tunnel types that insert mid-layer headers between the inner and outer IP headers (e.g., IPsec, GRE, etc.), the SEAL header is inserted between the mid-layer headers and outer IP header.

## 5. SEAL Specification

The following sections specify the operation of SEAL:

### 5.1. SEAL Tunnel Model

SEAL is an encapsulation sublayer used within point-to-point, point-to-multipoint, and non-broadcast, multiple access (NBMA) tunnels. Each SEAL path is configured over one or more underlying interfaces attached to subnetwork links. The SEAL tunnel connects an ITE to one or more ETE "neighbors" via encapsulation across an underlying subnetwork, where the tunnel neighbor relationship may be bidirectional, partially unidirectional or fully unidirectional.

A bidirectional tunnel neighbor relationship is one over which both

TEs can exchange both data and control messages. A partially unidirectional tunnel neighbor relationship allows the near end ITE to send data packets forward to the far end ETE, while the far end only returns control messages when necessary. Finally, a fully unidirectional mode of operation is one in which the near end ITE can receive neither data nor control messages from the far end ETE.

Implications of the SEAL bidirectional and unidirectional models are the same as discussed in [I-D.templin-intarea-vet].

## 5.2. SEAL Model of Operation

SEAL-enabled ITEs encapsulate each inner packet in any ancillary tunnel protocol headers, a SEAL header, any outer header encapsulations and in some instances a SEAL trailer as shown in Figure 1:

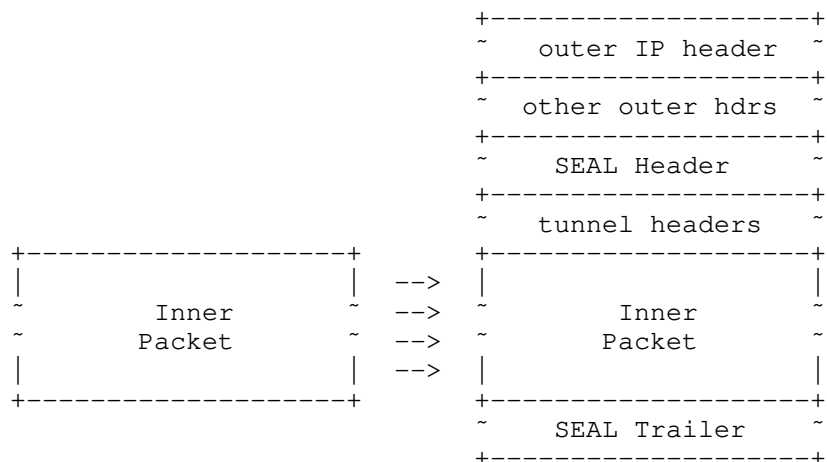


Figure 1: SEAL Encapsulation

The ITE inserts the SEAL header according to the specific tunneling protocol. For simple encapsulation of an inner network layer packet within an outer IP header, the ITE inserts the SEAL header following the outer IP header and before the inner packet as: IP/SEAL/{inner packet}.

For encapsulations over transports such as UDP, the ITE inserts the SEAL header following the outer transport layer header and before the inner packet, e.g., as IP/UDP/SEAL/{inner packet}. In that case, the UDP header is seen as an "other outer header" as depicted in Figure 1 and the outer IP and transport layer headers are together seen as the outer encapsulation headers. (Note that outer transport layer

headers such as UDP must sometimes be included to ensure that SEAL packets will traverse the path to the ETE without loss due filtering middleboxes. The ETE MUST accept both IP/SEAL and IP/UDP/SEAL as equivalent packets so that the ITE can discontinue outer transport layer encapsulation if the path supports raw IP/SEAL encapsulation.)

For SEAL encapsulations that involve tunnel types that include ancillary tunnel headers (e.g., GRE, IPsec, etc.) the ITE inserts the SEAL header as a leading extension to the tunnel headers, i.e., the SEAL encapsulation appears as part of the same tunnel and not a separate tunnel. For example, for GRE the ITE inserts the SEAL header as IP/SEAL/GRE/{inner packet}, and for IPsec the ITE inserts the SEAL header as IP/SEAL/IPsec-header/{inner packet}/IPsec-trailer. In such cases, SEAL considers the length of the inner packet only (i.e., and not the other tunnel headers and trailers) when performing its packet size calculations.

SEAL supports both "nested" tunneling and "re-encapsulating" tunneling. Nested tunneling occurs when a first tunnel is encapsulated within a second tunnel, which may then further be encapsulated within additional tunnels. Nested tunneling can be useful, and stands in contrast to "recursive" tunneling which is an anomalous condition incurred due to misconfiguration or a routing loop. Considerations for nested tunneling and avoiding recursive tunneling are discussed in Section 4 of [RFC2473] as well as in Section 9 of this document.

Re-encapsulating tunneling occurs when a packet arrives at a first ETE, which then acts as an ITE to re-encapsulate and forward the packet to a second ETE connected to the same subnetwork. In that case each ITE/ETE transition represents a segment of a bridged path between the ITE nearest the source and the ETE nearest the destination. Considerations for re-encapsulating tunneling are discussed in [I-D.templin-ironbis]. Combinations of nested and re-encapsulating tunneling are also naturally supported by SEAL.

The SEAL ITE considers each underlying interface as the ingress attachment point to a separate SEAL path to the ETE. The ITE therefore may experience different path MTUs on different SEAL paths.

Finally, the SEAL ITE ensures that the inner network layer protocol will see a minimum MTU of 1500 bytes over each SEAL path regardless of the outer network layer protocol version, i.e., even if a small amount of segmentation and reassembly are necessary. This is to avoid path MTU "black holes" for the minimum MTU configured by the vast majority of links in the Internet. Note that in some scenarios, however, reassembly may place a heavy burden on the ETE. In that case, the ITE can avoid invoking segmentation and instead report an

MTU smaller than 1500 bytes to the original source.

### 5.3. SEAL Encapsulation Format

SEAL encapsulates each inner packet within any ancillary tunneling protocol headers and a SEAL header. The SEAL header shares the same format as the IPv6 Fragment Header [RFC2460] and is identified by the same IP protocol number assigned for the IPv6 Fragment Header (type '44') [I-D.ietf-6man-ext-transmit]. The SEAL header is differentiated from the IPv6 Fragment Header by including a non-zero value in the most significant two bits of the IPv6 Fragment Header "Reserved" field; these two bits will heretofore serve as a SEAL protocol version number. SEAL therefore updates the IPv6 Fragment Header specification found in [RFC2460].

The SEAL header is formatted as shown in Figure 2:

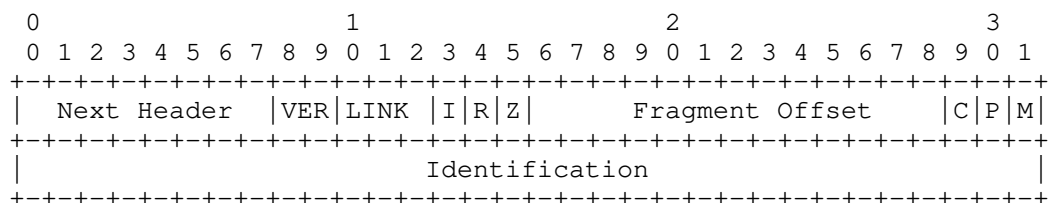


Figure 2: SEAL Encapsulation Format

The fields of the SEAL header are formatted as follows:

**Next Header (8)** an 8-bit field that encodes the next header Internet Protocol number the same as for the IPv4 protocol and IPv6 next header fields.

**VER (2)**  
a 2-bit version field. This document specifies Version 1 of the SEAL protocol, i.e., the VER field encodes the value '01'.

**LINK (3)**  
a 3-bit link identification value, set to a unique value by the ITE for each SEAL path over which it will send encapsulated packets to the ETE (up to 8 SEAL paths per ETE are therefore supported). Note that, if the ITE's interface connection to the underlying link assigns multiple IP addresses, each address represents a separate SEAL path that must be assigned a separate link ID.

- I (1)  
the "Integrity Check Vector (ICV) included" bit.
- R (1)  
the "Redirects Permitted" bit when used by VET (see: [I-D.templin-intarea-vet]); reserved for future use in other contexts.
- Z (1)  
a 1-bit Reserved field. Initialized to zero for transmission; ignored on reception.
- Fragment Offset (13) a 13-bit Offset field. The offset, in 8-octet units, of the data following this header.
- C (1)  
the "Control/Data" bit. Set to 1 by the ITE in SEAL Control Message Protocol (SCMP) control messages, and set to 0 in ordinary data packets.
- P (1)  
The "Probe" bit when C=0; set to 1 by the ITE in SEAL probe data packets for which it wishes to receive an explicit acknowledgement from the ETE. The "Pass" bit when C=1; set to 1 by the ETE in SCMP messages it relays to the ITE on behalf of another SEAL path.
- M (1) the "More Segments" bit. Set to 1 in a non-final segment and set to 0 in the final segment of the SEAL packet.
- Identification (32)  
a 32-bit per-packet identification field. Set to a randomly-initialized 32-bit value that is monotonically-incremented for each SEAL packet transmitted to this ETE.

When an Integrity Check Vector (ICV) is included, it is added as a trailing field at the end of the SEAL packet. The ICV is formatted as shown in Figure 3:

```

+-----+
|F|Key|Algorithm|           Message Authentication Code (MAC)           |
+-----+-----+

```

Figure 3: Integrity Check Vector (ICV) Format

As shown in the figure, the ICV begins with a 1-octet control field with a 1-bit (F)lag, a 2-bit Key identifier and a 5-bit Algorithm identifier. The control octet is followed by a variable-length Message Authentication Code (MAC). The ITE maintains a per ETE

algorithm and secret key to calculate the MAC in each packet it will send to this ETE. (By default, the ITE sets the F bit and Algorithm fields to 0 to indicate use of the HMAC-SHA-1 algorithm with a 160 bit shared secret key to calculate an 80 bit MAC per [RFC2104] over the leading 128 bytes of the packet. Other values for F and Algorithm are out of scope.)

#### 5.4. ITE Specification

##### 5.4.1. Tunnel MTU

The tunnel must present a stable MTU value to the inner network layer as the size for admission of inner packets into the tunnel. Since tunnels may support a large set of SEAL paths that accept widely varying maximum packet sizes, however, a number of factors should be taken into consideration when selecting a tunnel MTU.

Due to the ubiquitous deployment of standard Ethernet and similar networking gear, the nominal Internet cell size has become 1500 bytes; this is the de facto size that end systems have come to expect will either be delivered by the network without loss due to an MTU restriction on the path or a suitable ICMP Packet Too Big (PTB) message returned. When large packets sent by end systems incur additional encapsulation at an ITE, however, they may be dropped silently within the tunnel since the network may not always deliver the necessary PTBs [RFC2923]. The ITE SHOULD therefore set a tunnel MTU of at least 1500 bytes and provide accommodations to ensure that packets up to that size are successfully conveyed to the ETE.

The inner network layer protocol consults the tunnel MTU when admitting a packet into the tunnel. For non-SEAL inner IPv4 packets with the IPv4 Don't Fragment (DF) bit cleared (i.e, DF==0), if the packet is larger than the tunnel MTU the inner IPv4 layer uses IPv4 fragmentation to break the packet into fragments no larger than the MTU. The ITE then admits each fragment into the tunnel as an independent packet.

For all other inner packets, the inner network layer admits the packet if it is no larger than the tunnel MTU; otherwise, it drops the packet and sends a PTB error message to the source with the MTU value set to the MTU. The message contains as much of the invoking packet as possible without the entire message exceeding the network layer minimum MTU size.

The ITE can alternatively set an indefinite tunnel MTU such that all inner packets are admitted into the tunnel regardless of their size (theoretical maximums are 64KB for IPv4 and 4GB for IPv6 [RFC2675]). For ITEs that host applications that use the tunnel directly, this



option must be carefully coordinated with protocol stack upper layers since some upper layer protocols (e.g., TCP) derive their packet sizing parameters from the MTU of the outgoing interface and as such may select too large an initial size. This is not a problem for upper layers that use conservative initial maximum segment size estimates and/or when the tunnel can reduce the upper layer's maximum segment size, e.g., by reducing the size advertised in the MSS option of outgoing TCP messages (sometimes known as "MSS clamping").

In light of the above considerations, the ITE SHOULD configure an indefinite MTU on \*router\* tunnels so that SEAL performs all subnetwork adaptation from within the tunnel as specified in the following sections. The ITE MAY instead set a smaller MTU on \*host\* tunnels; in that case, the RECOMMENDED MTU is the maximum of 1500 bytes and the smallest MTU among all of the underlying links minus the size of the encapsulation headers.

#### 5.4.2. Tunnel Neighbor Soft State

The ITE maintains a number of soft state variables for each ETE and for each SEAL path.

The ITE maintains a per ETE window of Identification values for the packets it has recently sent to this ETE as well as a per ETE window of Identification values for the packets it has recently received from this ETE. The ITE then includes an Identification in each packet it sends to this ETE.

When message origin authentication and integrity checking is required, the ITE sets a variable "USE\_ICV" to TRUE, and includes a trailing ICV in each packet it sends to this ETE; otherwise, it sets USE\_ICV to FALSE.

For each SEAL path, the ITE must also account for encapsulation header lengths. The ITE therefore maintains the per SEAL path constant values "SHLEN" set to the length of the SEAL header and trailer, "THLEN" set to the length of the outer encapsulating transport layer headers (or 0 if outer transport layer encapsulation is not used), "IHLEN" set to the length of the outer IP layer header, and "HLEN" set to (SHLEN+THLEN+IHLEN). (The ITE must include the length of the uncompressed headers even if header compression is enabled when calculating these lengths.) When SEAL is used in conjunction with another tunnel type such as GRE or IPsec, the length of the headers associated with those tunnels is also included in the HLEN calculation for the first segment only and the length of the associated trailers is included in the HLEN calculation for the final segment only.

The ITE maintains a per SEAL path variable "MAXMTU" initialized to the maximum of (1500+HLEN) bytes and the MTU of the underlying link. The ITE further sets a variable 'MINMTU' to the minimum MTU for the SEAL path over which encapsulated packets will travel. For IPv6 paths, the ITE sets MINMTU=1280 per [RFC2460]. For IPv4 paths, the ITE sets MINMTU=576 based on practical interpretation of [RFC1122] even though the theoretical MINMTU for IPv4 is only 68 bytes [RFC0791].

The ITE can also set MINMTU to a larger value if there is reason to believe that the minimum path MTU is larger, or to a smaller value if there is reason to believe the MTU is smaller, e.g., if there may be additional encapsulations on the path. If this value proves too large, the ITE will receive PTB message feedback either from the ETE or from a router on the path and will be able to reduce its MINMTU to a smaller value. (Note that since IPv4 links with MTUs smaller than 1280 are presumably performance-constrained, the ITE can instead initialize MINMTU to 1280 the same as for IPv6. If this value proves too large, standard IPv4 fragmentation and reassembly will provide short term accommodation for the sizing constraints while the ITE readjusts its MINMTU estimate.)

The ITE may instead maintain the packet sizing variables and constants as per ETE (rather than per SEAL path) values. In that case, the values reflect the smallest MTU size across all of the SEAL paths associated with this ETE.

#### 5.4.3. SEAL Layer Pre-Processing

The SEAL layer is logically positioned between the inner and outer network protocol layers, where the inner layer is seen as the (true) network layer and the outer layer is seen as the (virtual) data link layer. Each packet to be processed by the SEAL layer is either admitted into the tunnel by the inner network layer protocol as described in Section 5.4.1 or is undergoing re-encapsulation from within the tunnel. The SEAL layer sees the former class of packets as inner packets that include inner network and transport layer headers, and sees the latter class of packets as transitional SEAL packets that include the outer and SEAL layer headers that were inserted by the previous hop SEAL ITE. For these transitional packets, the SEAL layer re-encapsulates the packet with new outer and SEAL layer headers when it forwards the packet to the next hop SEAL ITE.

We now discuss the SEAL layer pre-processing actions for these two classes of packets.

#### 5.4.3.1. Inner Packet Pre-Processing

For each for non-SEAL IPv4 inner packet with DF==0 in the IP header and IPv6 inner packet with a fragment header and with (MF=0; Offset=0), if the packet is larger than (MINMTU-HLEN) the ITE uses IP fragmentation to fragment the packet into N pieces, where N is minimized. (For IPv6 as the inner protocol, the first fragment MUST be at least as large as the IPv6 minimum of 1280 bytes so that the entire IPv6 header chain is likely to fit within the first segment.) The ITE then submits each fragment for SEAL encapsulation as specified in Section 5.4.4.

For all other inner packets, if the packet is no larger than (MAXMTU-HLEN) for the corresponding SEAL path the ITE submits it for SEAL encapsulation as specified in Section 5.4.4. Otherwise, the ITE drops the packet and sends an ordinary PTB message appropriate to the inner protocol version (subject to rate limiting) with the MTU field set to (MAXMTU-HLEN). (For IPv4 SEAL packets with DF==0, the ITE SHOULD set DF=1 and re-calculate the IPv4 header checksum before generating the PTB message in order to avoid bogon filters.) After sending the PTB message, the ITE discards the inner packet.

#### 5.4.3.2. Transitional SEAL Packet Pre-Processing

For each transitional packet that is to be processed by the SEAL layer from within the tunnel, if the packet is larger than MAXMTU bytes for the next hop SEAL path the ITE sends an SCMP Packet Too Big (SPTB) message to the previous hop subject to rate limiting with the MTU field set to MAXMTU and with (C=1; P=1) in the SEAL header (see: Section 5.6.1.1). After sending the SPTB message, the ITE discards the packet. Otherwise, the ITE sets aside the encapsulating SEAL and outer headers and submits the inner packet for SEAL re-encapsulation as specified in Section 5.4.4. (Note that in the calculation for MAXMTU, HLEN for the next hop SEAL path may be different than HLEN for the previous hop. In that case, MAXMTU must reflect the smaller of the two HLEN values.)

#### 5.4.4. SEAL Encapsulation and Segmentation

For each inner packet/fragment submitted for SEAL encapsulation, the ITE next encapsulates the packet in a SEAL header formatted as specified in Section 5.3. The ITE next sets (C=0; P=0), sets LINK to the value assigned to the underlying SEAL path, and sets the Next Header field to the protocol number corresponding to the address family of the encapsulated inner packet. For example, the ITE sets the Next Header field to the value '4' for encapsulated IPv4 packets [RFC2003], '41' for encapsulated IPv6 packets [RFC2473][RFC4213], '47' for GRE [RFC1701], '80' for encapsulated OSI/CLNP packets

[RFC1070], etc.

Next, if the inner packet is no larger than (MINMTU-HLEN) or larger than 1500, the ITE sets (M=0; Fragment Offset=0). Otherwise, the ITE breaks the inner packet into N non-overlapping segments, where N is minimized. For IPv6 as the inner protocol, the resulting encapsulated SEAL packet containing the first segment MUST be at least as large as the IPv6 minimum of 1280 bytes so that the entire IPv6 header chain is likely to fit within the first segment. (Since the Fragment Offset field indicates the number of 8 byte units, however, if HLEN is not an integer multiple of 8 bytes the encapsulated SEAL packet MAY contain up to 7 bytes less than 1280 so that the IPv6 minimum MTU is not exceeded.)

The ITE then appends a clone of the SEAL header from the first segment onto the head of each additional segment. The ITE then sets (M=1; Fragment Offset=0) in the first segment, sets (M=0/1; Fragment Offset=0(1)) in the second segment, sets (M=0/1; Fragment Offset=0(2)) in the third segment (if needed), etc., then finally sets (M=0; Fragment Offset=0(n)) in the final segment (where O(i) is the number of 256 byte blocks that preceded this segment).

The ITE then writes a monotonically-incrementing integer value for this ETE in the Identification field beginning with a randomly-initialized value in the first packet transmitted. (For SEAL packets that have been split into multiple pieces, the ITE writes the same Identification value in each piece.) The monotonically-incrementing requirement is to satisfy ETEs that use this value for anti-replay purposes. The value is incremented modulo  $2^{32}$ , i.e., it wraps back to 0 when the previous value was  $(2^{32} - 1)$ .

When USE\_ICV is FALSE, the ITE next sets I=0. Otherwise, the ITE sets I=1, includes a trailing ICV and calculates the MAC using HMAC-SHA-1 with a 160 bit secret key and 80 bit MAC field. Beginning with the SEAL header, the ITE calculates the MAC over the leading 128 bytes of the packet (or up to the end of the packet if there are fewer than 128 bytes) and places the result in the MAC field. (For SEAL packets that have been split into multiple pieces, each piece calculates its own MAC.) The ITE then writes the value 0 in the F flag and 0x00 in the Algorithm field of the ICV control octet (other values for these fields, and other MAC calculation disciplines, are outside the scope of this document and may be specified in future documents.)

If the packet is undergoing SEAL re-encapsulation, the ITE then copies the R value from the SEAL header of the packet to be re-encapsulated. Otherwise, it sets R=0 unless otherwise specified in other documents that employ SEAL. The ITE then adds the outer

encapsulating headers as specified in Section 5.4.5.

#### 5.4.5. Outer Encapsulation

Following SEAL encapsulation, the ITE next encapsulates each segment in the requisite outer transport (when necessary) and IP layer headers. When a transport layer header such as UDP or TCP is included, the ITE writes the port number for SEAL in the transport destination service port field.

When UDP encapsulation is used, the ITE sets the UDP checksum field to zero for IPv4 packets and also sets the UDP checksum field to zero for IPv6 packets even though IPv6 generally requires UDP checksums. Further considerations for setting the UDP checksum field for IPv6 packets are discussed in [RFC6935][RFC6936].

The ITE then sets the outer IP layer headers the same as specified for ordinary IP encapsulation (e.g., [RFC1070][RFC2003], [RFC2473], [RFC4213], etc.) except that for ordinary SEAL packets the ITE copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the inner network layer header into the corresponding fields in the outer IP header. For transitional SEAL packets undergoing re-encapsulation, the ITE instead copies the "TTL/Hop Limit", "Type of Service/Traffic Class" and "Congestion Experienced" values in the original outer IP header of the transitional packet into the corresponding fields in the new outer IP header of the packet to be forwarded (i.e., the values are transferred between outer headers and *\*not\** copied from the inner network layer header).

The ITE also sets the IP protocol number to the appropriate value for the first protocol layer within the encapsulation (e.g., UDP, TCP, SEAL, etc.). When IPv6 is used as the outer IP protocol, the ITE then sets the flow label value in the outer IPv6 header the same as described in [RFC6438]. When IPv4 is used as the outer IP protocol, the ITE sets DF=0 in the IPv4 header to allow the packet to be fragmented if it encounters a restricting link (for IPv6 SEAL paths, the DF bit is absent but implicitly set to 1).

The ITE finally sends each outer packet via the underlying link corresponding to LINK.

#### 5.4.6. Path Probing and ETE Reachability Verification

All SEAL data packets sent by the ITE are considered implicit probes that detect MTU limitations on the SEAL path, while explicit probe packets can be constructed to probe the path MTU and/or verify ETE reachability. These probes will elicit an SCMP message from the ETE

if it needs to send an acknowledgement and/or report an error condition. The probe packets may also be dropped by either the ETE or a router on the path, which may or may not result in an ICMP message being returned to the ITE.

To generate an explicit probe packet, the ITE creates a duplicate of an actual data packet and uses the duplicate as a probe. (Alternatively, the ITE can create a packet buffer beginning with the same outer headers, SEAL header and inner network layer headers that would appear in an ordinary data packet, then pad the packet with random data.) The ITE then sets (C=0; P=1) in the SEAL header of the probe packet, and also sets DF=1 in the outer IP header when IPv4 is used.

The ITE sends periodic explicit probes to determine whether SEAL segmentation is still necessary (see Section 5.4.4). In particular, if a probe packet of 1500 bytes (i.e., a packet that becomes (1500+HLEN) bytes after encapsulation) succeeds without incurring fragmentation the ITE is assured that the path MTU is large enough so that the segmentation/reassembly process can be suspended. This probing discipline can therefore be considered as Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] applied to tunnels, which operates independently of any application of PLPMTUD between end systems. Note that the explicit probe size of 1500 bytes is chosen since probe packets smaller than this size may be fragmented by a nested ITE further down the path. For example, a successful probe for a packet size of 1400 bytes does not guarantee that fragmentation is not occurring at another ITE.

The ITE can also send probes to detect whether an outer transport layer header is no longer necessary to reach this ETE. For example, if the ITE sends its initial packets as IP/UDP/SEAL/\*, it can send probes constructed as IP/SEAL/\* to determine whether the ETE is reachable without the added layer of encapsulation. If so, the ITE should also re-probe the path MTU since switching to a new encapsulation type may result in a path change.

While probing, the ITE processes ICMP messages as specified in Section 5.4.7 and processes SCMP messages as specified in Section 5.6.2.

#### 5.4.7. Processing ICMP Messages

When the ITE sends SEAL packets, it may receive ICMP error messages [RFC0792][RFC4443] from a router on the path to the ETE. Each ICMP message includes an outer IP header, followed by an ICMP header, followed by a portion of the SEAL data packet that generated the error (also known as the "packet-in-error"). Note that the ITE may

receive an ICMP message from another ITE that is at the head end of a nested level of encapsulation. The ITE has no security associations with this nested ITE, hence it should consider the message the same as if it originated from an ordinary router on the path to the ETE.

The ITE should process ICMPv4 Protocol Unreachable messages and ICMPv6 Parameter Problem messages with Code "Unrecognized Next Header type encountered" as a hint that the ETE does not implement SEAL. The ITE can optionally ignore other ICMP messages that do not include sufficient information in the packet-in-error, or process them as a hint that the SEAL path to the ETE may be failing. The ITE then discards these types of messages.

For other ICMP messages, the ITE first examines the SEAL data packet within the packet-in-error field. If the IP source and/or destination addresses are invalid, or if the value in the SEAL header Identification field (if present) is not within the window of packets the ITE has recently sent to this ETE, or if the MAC value in the ICV field (if present) is incorrect, the ITE discards the message.

Next, if the received ICMP message is a PTB the ITE sets the temporary variable "PMTU" for this SEAL path to the MTU value in the PTB message. If the outer IP length value in the packet-in-error is no larger than (1500+HLEN) bytes the ITE sets MAXMTU=(1500+HLEN) and discards the message. If the outer IP length value in the packet-in-error is larger than (1500+HLEN) bytes and PMTU is no smaller than MINMTU the ITE sets MAXMTU to the maximum of (1500+HLEN) and PMTU; otherwise the ITE consults a plateau table (e.g., as described in [RFC1191]) to determine a new value for MAXMTU. For example, if the ITE receives a PTB message with small PMTU and packet-in-error length 8KB, it can set MAXMTU=4KB. If the ITE subsequently receives a PTB message with small PMTU and length 4KB, it can set MAXMTU=2KB, etc., to a minimum value of MAXMTU=(1500+HLEN). Next, if the packet-in-error was an explicit probe (i.e., one with P=1 in the SEAL header), the ITE discards the message. Finally, if the ITE is using a MINMTU value larger than 1280 for IPv6 or 576 for IPv4, it may need to reduce MINMTU if the PMTU value is small.

If the ICMP message was not discarded, the ITE transcribes it into a message appropriate for the SEAL data packet within the packet-in-error. If the previous hop toward the inner source address within the SEAL data packet is reached via the same SEAL tunnel, the ITE transcribes the message into an SCMP message the same as described for ETE generation of SCMP messages in Section 5.6.1, i.e., it copies the SEAL data packet within the packet-in-error into the packet-in-error field of the new message. (In this process, the ETE also sets (C=1; P=1) in the SEAL header of the SCMP message.) Otherwise, the ITE seeks beyond the SEAL header within the packet-in-error and

transcribes the inner packet into a message appropriate for the inner protocol version (e.g., ICMPv4 for IPv4, ICMPv6 for IPv6, etc.).

The ITE finally forwards the transcribed message to the previous hop toward the inner source address.

#### 5.4.8. IPv4 Middlebox Reassembly Testing

The ITE can perform a qualification exchange to ensure that the subnetwork correctly delivers fragments to the ETE. This procedure can be used, e.g., to determine whether there are middleboxes on the path that violate the [RFC1812], Section 5.2.6 requirement that: "A router MUST NOT reassemble any datagram before forwarding it". Examples of middleboxes that may perform reassembly include stateful NATs and firewalls. Such devices could still allow for stateless MTU determination if they gather the fragments of a fragmented SEAL data packet for packet analysis purposes but then forward the fragments on to the final destination rather than forwarding the reassembled packet. (This process is often referred to as "Virtual Fragmentation Reassembly" (VFR)).

The ITE should use knowledge of its topological arrangement as an aid in determining when middlebox reassembly testing is necessary. For example, if the ITE is aware that the ETE is located somewhere in the public Internet, middlebox reassembly testing should not be necessary. If the ITE is aware that the ETE is located behind a NAT or a firewall, however, then reassembly testing can be used to detect middleboxes that do not conform to specifications.

The ITE can perform a middlebox reassembly test by sending explicit probe packets. The ITE should only send probe packets that are smaller than (576-HLEN) before encapsulation since the least an ordinary node can be expected to reassemble is 576 bytes. To generate a probe, the ITE either creates a clone of an ordinary data packet or creates a packet buffer beginning with the same outer headers, SEAL header and inner network layer header that would appear in an ordinary data packet. The ITE then pads the probe packet with random data to a length that is at least 128 bytes but smaller than (576-HLEN) bytes.

The ITE then sets (C=0; P=1) in the SEAL header of the probe packet and sets the Next Header field to the inner network layer protocol type. Next, the ITE sets LINK to the appropriate value for this SEAL path, sets the Identification field, then finally calculates the ICV and sets I=1 (when USE\_ICV is TRUE).

The ITE then encapsulates the probe packet in the appropriate outer headers, splits it into two outer IP fragments, then sends both



fragments over the same SEAL path.

The ITE should send a series of probe packets (e.g., 3-5 probes with 1sec intervals between tests) instead of a single isolated probe in case of packet loss. If the ETE returns an SCMP PTB message with the original first fragment in the packet-in-error, then the SEAL path correctly supports fragmentation; otherwise, the ITE enables stateful MTU determination for this SEAL path as specified in Section 5.4.9.

#### 5.4.9. Stateful MTU Determination

SEAL supports a stateless MTU determination capability, however the ITE may in some instances wish to impose a stateful MTU limit on a particular SEAL path. For example, when the ETE is situated behind a middlebox that performs reassembly in violation of the specs (see: Section 5.4.8) it is imperative that fragmentation be avoided. In other instances (e.g., when the SEAL path includes performance-constrained links), the ITE may deem it necessary to cache a conservative static MTU in order to avoid sending large packets that would only be dropped due to an MTU restriction somewhere on the path.

To determine a static MTU value, the ITE can send a series of probe packets of various sizes to the ETE with (C=0; P=1) in the SEAL header and DF=1 in the outer IP header. The ITE then caches the size 'S' of the largest packet for which it receives a probe reply from the ETE by setting  $MAXMTU = \text{MAX}((S, (1500 + HLEN)))$  for this SEAL path.

For example, the ITE could send probe packets of 8KB, followed by 4KB, followed by 2KB, etc. While probing, the ITE processes any ICMP PTB message it receives as a potential indication of probe failure then discards the message.

#### 5.4.10. Detecting Path MTU Changes

When stateful MTU determination is used, the ITE SHOULD periodically reset MAXMTU and/or re-probe the path to determine whether MAXMTU has increased. If the path still has a too-small MTU, the ITE will receive a PTB message that reports a smaller size.

### 5.5. ETE Specification

#### 5.5.1. Reassembly Buffer Requirements

For IPv6, the ETE MUST configure a minimum reassembly buffer size of (1500 + HLEN) bytes for the reassembly of outer IPv6 packets, i.e., even though the true minimum reassembly size for IPv6 is only 1500 bytes [RFC2460]. For IPv4, the ETE also MUST configure a minimum

reassembly buffer size of (1500 + HLEN) bytes for the reassembly of outer IPv4 packets, i.e., even though the true minimum reassembly size for IPv4 is only 576 bytes [RFC1122].

In addition to this outer reassembly buffer requirement, the ETE further MUST configure a minimum SEAL reassembly buffer size of (1500 + HLEN) bytes for the reassembly of segmented SEAL packets (see: Section 5.5.4).

Note that the value "HLEN" may be variable and initially unknown to the ETE, and would typically range from a few bytes to a few tens of bytes or even more. It is therefore RECOMMENDED that the ETE configure slightly larger minimum IP/SEAL reassembly buffer sizes of 2048 bytes (2KB).

#### 5.5.2. Tunnel Neighbor Soft State

When message origin authentication and integrity checking is required, the ETE maintains a per-ITE MAC calculation algorithm and a symmetric secret key to verify the MAC. The ETE also maintains a window of Identification values for the packets it has recently received from this ITE as well as a window of Identification values for the packets it has recently sent to this ITE.

When the tunnel neighbor relationship is bidirectional, the ETE further maintains a per SEAL path mapping of outer IP and transport layer addresses to the LINK value that appears in packets received from the ITE.

#### 5.5.3. IP-Layer Reassembly

The ETE reassembles fragmented IP packets that are explicitly addressed to itself. For IP fragments that are received via a SEAL tunnel, the ETE SHOULD maintain conservative reassembly cache high- and low-water marks. When the size of the reassembly cache exceeds this high-water mark, the ETE SHOULD actively discard stale incomplete reassemblies (e.g., using an Active Queue Management (AQM) strategy) until the size falls below the low-water mark. The ETE SHOULD also actively discard any pending reassemblies that clearly have no opportunity for completion, e.g., when a considerable number of new fragments have arrived before a fragment that completes a pending reassembly arrives.

The ETE processes non-SEAL IP packets as specified in the normative references, i.e., it performs any necessary IP reassembly then discards the packet if it is larger than the reassembly buffer size or delivers the (fully-reassembled) packet to the appropriate upper layer protocol module.

For SEAL packets, the ETE performs any necessary IP reassembly then submits the packet for SEAL decapsulation as specified in Section 5.5.4. (Note that if the packet is larger than the reassembly buffer size, the ETE still examines the leading portion of the (partially) reassembled packet during decapsulation.)

#### 5.5.4. Decapsulation, SEAL-Layer Reassembly, and Re-Encapsulation

For each SEAL packet accepted for decapsulation, the ETE first examines the Identification field. If the Identification is not within the window of acceptable values for this ITE, the ETE silently discards the packet.

Next, if  $I==1$  the ETE SHOULD verify the MAC value and silently discard the packet if the value is incorrect. (Note that this means that the ETE would need to receive all IP fragments if the packet was fragmented at the outer IP layer, since the MAC is included as a trailing field.)

Next, if the packet arrived as multiple IP fragments, the ETE sends an SPTB message back to the ITE with MTU set to the size of the largest fragment received (see: Section 5.6.1.1).

Next, if the packet arrived as multiple IP fragments and the inner packet is larger than 1500 bytes, the ETE silently discards the packet; otherwise, it continues to process the packet.

Next, if there is an incorrect value in a SEAL header field (e.g., an incorrect "VER" field value), the ETE discards the packet. If the SEAL header has  $C==0$ , the ETE also returns an SCMP "Parameter Problem" (SPP) message (see Section 5.6.1.2).

Next, if the SEAL header has  $C==1$ , the ETE processes the packet as an SCMP packet as specified in Section 5.6.2. Otherwise, the ETE continues to process the packet as a SEAL data packet.

Next, if the SEAL header has  $(M==1 \ || \ \text{Fragment Offset}!=0)$  the ETE checks to see if the other segments of this already-segmented SEAL packet have arrived, i.e., by looking for additional segments that have the same outer IP source address, destination address, source port number and SEAL Identification value. If all other segments have already arrived, the ETE discards the SEAL header and other outer headers from the non-initial segments and appends the segments onto the end of the first segment according to their offset value. Otherwise, the ETE caches the new segment for at most 60 seconds while awaiting the arrival of its partners. During this process, the ETE discards any segments that are overlapping with respect to segments that have already been received, and also discards any

segments that have  $M=1$  in the SEAL header but do not contain an integer multiple of 8 bytes. The ETE further SHOULD manage the SEAL reassembly cache the same as described for the IP-Layer Reassembly cache in Section 5.5.3, i.e., it SHOULD perform an early discard for any pending reassemblies that have low probability of completion.

Next, if the SEAL header in the (reassembled) packet has  $P=1$ , the ETE drops the packet unconditionally and sends an SPTB message back to the ITE (see: Section 5.6.1.1) if it has not already sent an SPTB message based on IP fragmentation. (Note that the ETE therefore sends only a single SPTB message for a probe packet that also experienced IP fragmentation, i.e., it does not send multiple SPTB messages.)

Finally, the ETE discards the outer headers and processes the inner packet according to the header type indicated in the SEAL Next Header field. If the next hop toward the inner destination address is via a different interface than the SEAL packet arrived on, the ETE discards the SEAL header and delivers the inner packet either to the local host or to the next hop if the packet is not destined to the local host.

If the next hop is on the same tunnel the SEAL packet arrived on, however, the ETE submits the packet for SEAL re-encapsulation beginning with the specification in Section 5.4.3 above and without decrementing the value in the inner (TTL / Hop Limit) field.

#### 5.6. The SEAL Control Message Protocol (SCMP)

SEAL provides a companion SEAL Control Message Protocol (SCMP) that uses the same message types and formats as for the Internet Control Message Protocol for IPv6 (ICMPv6) [RFC4443]. The SCMP messaging protocol operates over bidirectional and partially unidirectional tunnels. (For fully unidirectional tunnels, SEAL must operate without the benefit of SCMP meaning that steady-state fragmentation and reassembly may be necessary in extreme cases. In that case, the ITE must select a conservative MINMTU to ensure that IPv4 fragmentation is avoided in order to avoid reassembly errors at high data rates [RFC4963].)

As for ICMPv6, each SCMP message includes a 32-bit header and a variable-length body. The ITE encapsulates the SCMP message in a SEAL header and outer headers as shown in Figure 4:

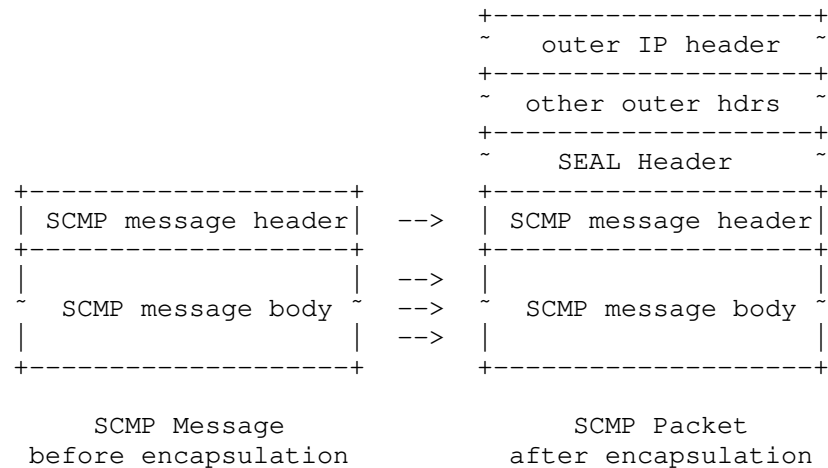
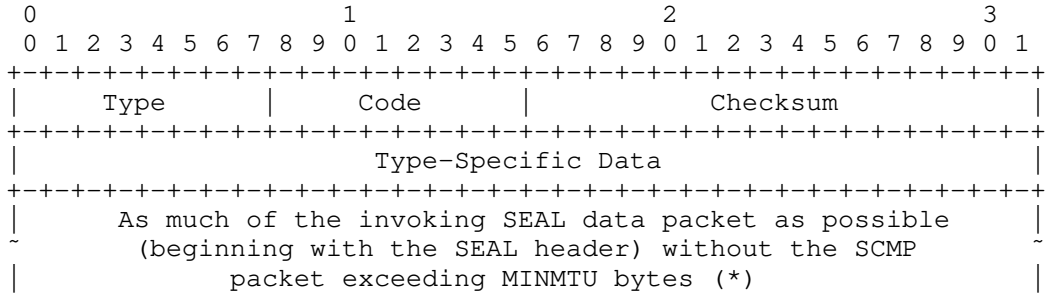


Figure 4: SCMP Message Encapsulation

The following sections specify the generation, processing and relaying of SCMP messages.

5.6.1. Generating SCMP Messages

ETEs generate SCMP messages in response to receiving certain SEAL data packets using the format shown in Figure 5:



(\*) also known as the "packet-in-error"

Figure 5: SCMP Message Format

The error message includes the 32-bit SCMP message header, followed by a 32-bit Type-Specific Data field, followed by the leading portion of the invoking SEAL data packet beginning with the SEAL header as the "packet-in-error". The packet-in-error includes as much of the invoking packet as possible extending to a length that would not cause the entire SCMP packet following outer encapsulation to exceed

MINMTU bytes.

When the ETE processes a SEAL data packet for which the Identification and ICV values are correct but an error must be returned, it prepares an SCMP message as shown in Figure 5. The ETE sets the Type and Code fields to the same values that would appear in the corresponding ICMPv6 message [RFC4443], but calculates the Checksum beginning with the SCMP message header using the algorithm specified for ICMPv4 in [RFC0792].

The ETE next encapsulates the SCMP message in the requisite SEAL and outer headers as shown in Figure 4. During encapsulation, the ETE sets the outer destination address/port numbers of the SCMP packet to the values associated with the ITE and sets the outer source address/port numbers to its own outer address/port numbers.

The ETE then sets (C=1; M=0; Fragment Offset=0) in the SEAL header, then sets I, Next Header and LINK to the same values that appeared in the SEAL header of the data packet. The ETE next sets the Identification field to the next Identification value scheduled for this ITE, then increments the next Identification value. When I==1, the ETE then prepares the ICV field the same as specified for SEAL data packet encapsulation in Section 5.4.4. If this message is in direct response to a SEAL data packet sent by the ITE, the ETE next sets P=0 and sends the resulting SCMP packet to the ITE the same as specified for SEAL data packets in Section 5.4.5.

If the message is in response to an SCMP message received from a next hop ETE or to an ICMP message received from a router on the path to a next hop ETE, the ETE instead sets P=1 and passes the message to the ITE in a "reverse re-encapsulation" process. In particular, when the previous hop toward the source of the inner packet within the packet-in-error in a received SCMP/ICMP message is reached via the same tunnel as the message arrived on, the ETE replaces the outer headers of the message (up to and including the SEAL header) with headers that will be recognized and accepted by the previous hop and sends the resulting packet to the previous hop.

The following sections describe additional considerations for various SCMP error messages:

#### 5.6.1.1. Generating SCMP Packet Too Big (SPTB) Messages

An ETE generates an SPTB message when it receives a SEAL probe packet (i.e., one with C=0; P=1 in the SEAL header) or when it receives a SEAL packet that arrived as multiple outer IP fragments. The ETE prepares the SPTB message the same as for the corresponding ICMPv6 PTB message, and writes the length of the largest outer IP fragment

received in the MTU field of the message (or the full length of the outer IP packet if the packet was unfragmented). In that case, the ETE sets (C=1; P=0) in the SEAL header.

An ETE also generates an SPTB message when it attempts to forward a SEAL data packet to a next hop ETE via the same tunnel the data packet arrived on, but for which MAXMTU for that SEAL path is insufficient to accommodate the packet (See Section 5.4.3.2). In that case, the ETE sets (C=1; P=1) in the SEAL header.

An ETE finally generates an SPTB message when it receives an ICMP PTB message from a router on the path to a next hop ETE (See Section 5.4.7). In that case, the ETE also sets (C=1; P=1) in the SEAL header.

#### 5.6.1.2. Generating Other SCMP Messages

An ETE generates an SCMP "Destination Unreachable" (SDU) message under the same conditions that an IPv6 system would generate an ICMPv6 Destination Unreachable message.

An ETE generates an SCMP "Parameter Problem" (SPP) message when it receives a SEAL packet with an incorrect value in the SEAL header.

TEs generate other SCMP message types using methods and procedures specified in other documents. For example, SCMP message types used for tunnel neighbor coordinations are specified in VET [I-D.templin-intarea-vet].

#### 5.6.2. Processing SCMP Messages

An ITE may receive SCMP messages with C==1 in the SEAL header after sending packets to an ETE. The ITE first verifies that the outer addresses of the SCMP packet are correct, and that the Identification field contains an acceptable value. The ITE next verifies that the SEAL header fields are set correctly as specified in Section 5.6.1. When I==1, the ITE then verifies the ICV. The ITE next verifies the Checksum value in the SCMP message header. If any of these values are incorrect, the ITE silently discards the message; otherwise, it processes the message as follows:

##### 5.6.2.1. Processing SCMP PTB Messages

After an ITE sends a SEAL packet to an ETE, it may receive an SPTB message with a packet-in-error containing the leading portion of the packet (see: Section 5.6.1.1). If the SEAL header has P==1 the ITE consults its forwarding information base to pass the message to the previous hop toward the source address of the encapsulated inner

packet. When the previous hop is reached via the same SEAL tunnel, the ITE passes the SPTB message to the previous hop as specified in Section 5.6.1. Otherwise, the ITE transcribes the inner packet within the packet-in-error into a message appropriate for the inner protocol version (e.g., ICMPv4 for IPv4, ICMPv6 for IPv6, etc.).

If the SEAL header has  $P==0$ , the ITE instead processes the message as an MTU limitation on the SEAL path to this ETE. In that case, the ITE first sets the temporary variable "PMTU" for this SEAL path to the MTU value in the SPTB message and processes the message as follows:

- o If PMTU is no smaller than  $(1500+HLEN)$ , the ITE suspends the SEAL segmentation/reassembly process for this SEAL path so that whole (unfragmented) SEAL packets can be used. If the packet is a probe being used to establish a stateful MTU for this SEAL path (see: section 5.4.9), the ITE also sets  $MAXMTU=PMTU$ .
- o If PMTU is smaller than  $(1500+HLEN)$  but no smaller than MINMTU the ITE sets  $MAXMTU$  to  $(1500+HLEN)$  and resumes the SEAL segmentation/reassembly process for this SEAL path.
- o If PMTU is smaller than MINMTU and the packet-in-error is a probe used for the purpose of middlebox reassembly detection (see: section 5.4.8), the ITE notes the results of the probe. Otherwise, the ITE consults a plateau table to determine a new value for  $MAXMTU$ . For example, if the ITE receives a PTB message with small PMTU and packet-in-error length 8KB, it can set  $MAXMTU=4KB$ . If the ITE subsequently receives a PTB message with small PMTU and length 4KB, it can set  $MAXMTU=2KB$ , etc., to a minimum value of  $MAXMTU=(1500+HLEN)$ . Finally, if the ITE is using a MINMTU value larger than 1280 for IPv6 or 576 for IPv4, it may need to reduce MINMTU if the PMTU value is small.

Next, if the packet-in-error was no larger than  $(1500+HLEN)$  or the packet-in-error was an explicit probe (i.e., one with  $(C==0; P==1)$  in the SEAL header of the packet-in-error), the ITE discards the SPTB message.

#### 5.6.2.2. Processing Other SCMP Error Messages

An ITE may receive an SDU message with an appropriate code under the same circumstances that an IPv6 node would receive an ICMPv6 Destination Unreachable message. The ITE transcribes the message and forwards it toward the source address of the inner packet within the packet-in-error the same as specified for SPTB messages with  $P==1$  in Section 5.6.2.1.



An ITE may receive an SPP message when the ETE receives a SEAL packet with an incorrect value in the SEAL header. The ITE should examine the SEAL header within the packet-in-error to determine whether different settings should be used in subsequent packets, but does not relay the message further.

TEs process other SCMP message types using methods and procedures specified in other documents. For example, SCMP message types used for tunnel neighbor coordinations are specified in VET [I-D.templin-intarea-vet].

## 6. Link Requirements

Subnetwork designers are expected to follow the recommendations in Section 2 of [RFC3819] when configuring link MTUs.

## 7. End System Requirements

End systems are encouraged to implement end-to-end MTU assurance (e.g., using Packetization Layer Path MTU Discovery (PLPMTUD) per [RFC4821]) even if the subnetwork is using SEAL.

When end systems use PLPMTUD, SEAL will ensure that the tunnel behaves as a link in the path that assures an MTU of at least 1500 bytes while not precluding discovery of larger MTUs. The PLPMTUD mechanism will therefore be able to function as designed in order to discover and utilize larger MTUs.

## 8. Router Requirements

Routers within the subnetwork are expected to observe the standard IP router requirements, including the implementation of IP fragmentation and reassembly as well as the generation of ICMP messages [RFC0792] [RFC1122] [RFC1812] [RFC2460] [RFC4443] [RFC6434].

Note that, even when routers support existing requirements for the generation of ICMP messages, these messages are often filtered and discarded by middleboxes on the path to the original source of the message that triggered the ICMP. It is therefore not possible to assume delivery of ICMP messages even when routers are correctly implemented.

## 9. Nested Encapsulation Considerations

SEAL supports nested tunneling - an example would be a recursive nesting of mobile networks, where the first network receives service from an ISP, the second network receives service from the first network, the third network receives service from the second network, etc. It is imperative that such nesting not extend indefinitely; SEAL tunnels therefore honor the Encapsulation Limit option defined in [RFC2473].

In such nested arrangements, the SEAL ITE has a tunnel neighbor relationship only with ETes at its own nesting level, i.e., it does not have a tunnel neighbor relationship with TEs at other nesting levels. Therefore, when an ITE 'A' within an outer nesting level needs to return an error message to an ITE 'B' within an inner nesting level, it generates an ordinary ICMP error message the same as if it were an ordinary router within the subnetwork. 'B' can then perform message validation as specified in Section 5.4.7, but full message origin authentication is not possible.

(Note that the SCMP protocol could instead be extended to allow an outer nesting level ITE 'A' to return an SCMP message to an inner nesting level ITE 'B' rather than return an ICMP message. This would conceptually allow the control messages to pass through firewalls and NATs, however it would give no more message origin authentication assurance than for ordinary ICMP messages. It was therefore determined that the complexity of extending the SCMP protocol was of little value within the context of the anticipated use cases for nested encapsulations.)

## 10. Reliability Considerations

Although a SEAL tunnel may span an arbitrarily-large subnetwork expanse, the IP layer sees the tunnel as a simple link that supports the IP service model. Links with high bit error rates (BERs) (e.g., IEEE 802.11) use Automatic Repeat-ReQuest (ARQ) mechanisms [RFC3366] to increase packet delivery ratios, while links with much lower BERs typically omit such mechanisms. Since SEAL tunnels may traverse arbitrarily-long paths over links of various types that are already either performing or omitting ARQ as appropriate, it would therefore be inefficient to require the tunnel endpoints to also perform ARQ.

## 11. Integrity Considerations

The SEAL header includes an integrity check field that covers the SEAL header and at least the inner packet headers. This provides for

header integrity verification on a segment-by-segment basis for a segmented re-encapsulating tunnel path.

Fragmentation and reassembly schemes must also consider packet-splicing errors, e.g., when two fragments from the same packet are concatenated incorrectly, when a fragment from packet X is reassembled with fragments from packet Y, etc. The primary sources of such errors include implementation bugs and wrapping IPv4 ID fields.

In particular, the IPv4 16-bit ID field can wrap with only 64K packets with the same (src, dst, protocol)-tuple alive in the system at a given time [RFC4963]. When the IPv4 ID field is re-written by a middlebox such as a NAT or Firewall, ID field wrapping can occur with even fewer packets alive in the system. It is therefore essential that IPv4 fragmentation and reassembly be detected early and tuned out through proper application of SEAL segmentation and reassembly.

## 12. IANA Considerations

The IANA is requested to allocate a User Port number for "SEAL" in the 'port-numbers' registry. The Service Name is "SEAL", and the Transport Protocols are TCP and UDP. The Assignee is the IESG (iesg@ietf.org) and the Contact is the IETF Chair (chair@ietf.org). The Description is "Subnetwork Encapsulation and Adaptation Layer (SEAL)", and the Reference is the RFC-to-be currently known as 'draft-templin-intarea-seal'.

## 13. Security Considerations

SEAL provides a segment-by-segment message origin authentication, integrity and anti-replay service. The SEAL header is sent in-the-clear the same as for the outer IP and other outer headers. In this respect, the threat model is no different than for IPv6 extension headers. Unlike IPv6 extension headers, however, the SEAL header can be protected by an integrity check that also covers the inner packet headers.

An amplification/reflection/buffer overflow attack is possible when an attacker sends IP fragments with spoofed source addresses to an ETE in an attempt to clog the ETE's reassembly buffer and/or cause the ETE to generate a stream of SCMP messages returned to a victim ITE. The SCMP message ICV, Identification, as well as the inner headers of the packet-in-error, provide mitigation for the ETE to detect and discard SEAL segments with spoofed source addresses.

Security issues that apply to tunneling in general are discussed in [RFC6169].

#### 14. Related Work

Section 3.1.7 of [RFC2764] provides a high-level sketch for supporting large tunnel MTUs via a tunnel-level segmentation and reassembly capability to avoid IP level fragmentation.

Section 3 of [RFC4459] describes inner and outer fragmentation at the tunnel endpoints as alternatives for accommodating the tunnel MTU.

Section 4 of [RFC2460] specifies a method for inserting and processing extension headers between the base IPv6 header and transport layer protocol data. The SEAL header is inserted and processed in exactly the same manner.

IPsec/AH is [RFC4301][RFC4301] is used for full message integrity verification between tunnel endpoints, whereas SEAL only ensures integrity for the inner packet headers. The AYIYA proposal [I-D.massar-v6ops-ayiya] uses similar means for providing message authentication and integrity.

SEAL, along with the Virtual Enterprise Traversal (VET) [I-D.templin-intarea-vet] tunnel virtual interface abstraction, are the functional building blocks for the Interior Routing Overlay Network (IRON) [I-D.templin-ironbis] and Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) [RFC5720][RFC6139] architectures.

The concepts of path MTU determination through the report of fragmentation and extending the IPv4 Identification field were first proposed in deliberations of the TCP-IP mailing list and the Path MTU Discovery Working Group (MTUDWG) during the late 1980's and early 1990's. An historical analysis of the evolution of these concepts, as well as the development of the eventual PMTUD mechanism, appears in [RFC5320].

#### 15. Implementation Status

An early implementation of the first revision of SEAL [RFC5320] is available at: <http://isatap.com/seal>.

## 16. Acknowledgments

The following individuals are acknowledged for helpful comments and suggestions: Jari Arkko, Fred Baker, Iljitsch van Beijnum, Oliver Bonaventure, Teco Boot, Bob Braden, Brian Carpenter, Steve Casner, Ian Chakeres, Noel Chiappa, Remi Denis-Courmont, Remi Despres, Ralph Droms, Aurnaud Ebalard, Gorry Fairhurst, Washam Fan, Dino Farinacci, Joel Halpern, Brian Haberman, Sam Hartman, John Heffner, Thomas Henderson, Bob Hinden, Christian Huitema, Eliot Lear, Darrel Lewis, Joe Macker, Matt Mathis, Erik Nordmark, Dan Romascanu, Dave Thaler, Joe Touch, Mark Townsley, Ole Troan, Margaret Wasserman, Magnus Westerlund, Robin Whittle, James Woodyatt, and members of the Boeing Research & Technology NST DC&NT group.

Discussions with colleagues following the publication of [RFC5320] have provided useful insights that have resulted in significant improvements to this, the Second Edition of SEAL.

This document received substantial review input from the IESG and IETF area directorates in the February 2013 timeframe. IESG members and IETF area directorate representatives who contributed helpful comments and suggestions are gratefully acknowledged. Discussions on the IETF IPv6 and Intarea mailing lists in the summer 2013 timeframe also stimulated several useful ideas.

Path MTU determination through the report of fragmentation was first proposed by Charles Lynn on the TCP-IP mailing list in 1987. Extending the IP identification field was first proposed by Steve Deering on the MTUDWG mailing list in 1989. Steve Deering also proposed the IPv6 minimum MTU of 1280 bytes on the IPng mailing list in 1997.

## 17. References

### 17.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.
- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [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.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [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.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

## 17.2. Informative References

- [FOLK] Shannon, C., Moore, D., and k. claffy, "Beyond Folklore: Observations on Fragmented Traffic", December 2002.
- [FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful", October 1987.
- [I-D.ietf-6man-ext-transmit]  
Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers",  
draft-ietf-6man-ext-transmit-05 (work in progress),  
October 2013.
- [I-D.massar-v6ops-ayiya]  
Massar, J., "AYIYA: Anything In Anything",  
draft-massar-v6ops-ayiya-02 (work in progress), July 2004.
- [I-D.taylor-v6ops-fragdrop]  
Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", draft-taylor-v6ops-fragdrop-01 (work in progress), June 2013.
- [I-D.templin-intarea-vet]  
Templin, F., "Virtual Enterprise Traversal (VET)",  
draft-templin-intarea-vet-40 (work in progress), May 2013.
- [I-D.templin-ironbis]  
Templin, F., "The Interior Routing Overlay Network (IRON)", draft-templin-ironbis-15 (work in progress), May 2013.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,

August 1980.

- [RFC0994] International Organization for Standardization (ISO) and American National Standards Institute (ANSI), "Final text of DIS 8473, Protocol for Providing the Connectionless-mode Network Service", RFC 994, March 1986.
- [RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.
- [RFC1070] Hagens, R., Hall, N., and M. Rose, "Use of the Internet as a subnetwork for experimentation with the OSI network layer", RFC 1070, February 1989.
- [RFC1146] Zweig, J. and C. Partridge, "TCP alternate checksum options", RFC 1146, March 1990.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.
- [RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.
- [RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.
- [RFC2764] Gleeson, B., Heinanen, J., Lin, A., Armitage, G., and A. Malis, "A Framework for IP Based Virtual Private Networks", RFC 2764, February 2000.
- [RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers",

BCP 37, RFC 2780, March 2000.

- [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.
- [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.
- [RFC3232] Reynolds, J., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.
- [RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, August 2002.
- [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.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, April 2006.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.



- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5320] Templin, F., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", RFC 5320, February 2010.
- [RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.
- [RFC5720] Templin, F., "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)", RFC 5720, February 2010.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.
- [RFC6139] Russert, S., Fleischman, E., and F. Templin, "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) Scenarios", RFC 6139, February 2011.
- [RFC6169] Krishnan, S., Thaler, D., and J. Hoagland, "Security Concerns with IP Tunneling", RFC 6169, April 2011.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, December 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.
- [RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, February 2013.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, April 2013.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.
- [RIPE] De Boer, M. and J. Bosma, "Discovering Path MTU Black Holes on the Internet using RIPE Atlas", July 2012.

- [SIGCOMM] Luckie, M. and B. Stasiewicz, "Measuring Path MTU Discovery Behavior", November 2010.
- [TBIT] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", October 2004.
- [WAND] Luckie, M., Cho, K., and B. Owens, "Inferring and Debugging Path MTU Discovery Failures", October 2005.

## Author's Address

Fred L. Templin (editor)  
Boeing Research & Technology  
P.O. Box 3707  
Seattle, WA 98124  
USA

Email: fltemplin@acm.org



6MAN  
Internet-Draft  
Intended status: Standards Track  
Expires: April 18, 2014

P. Thubert  
E. Levy-Abegnoli  
Cisco  
October 17, 2013

Wireless Neighbor Discovery Stateful Address Identification and Location  
exchange  
draft-thubert-6man-wind-sail-00

## Abstract

This draft proposes an extension to IPv6 Neighbor Discovery to exchange Stateful Address Identification and Location between State Maintaining Entities located over a backbone link about attached nodes that are attached to the backbone via a Wireless Link, in order to maintain all the entities up-to-date and maintain reachability as the attached nodes move.

## 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, 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.

## Table of Contents

1. Introduction . . . . .	2
---------------------------	---

2. Terminology . . . . .	4
3. Overview . . . . .	6
4. General Context . . . . .	8
4.1. Efficient ND . . . . .	8
4.2. Proxying classical ND . . . . .	10
4.3. Federating Large LLNs . . . . .	11
5. New types and formats . . . . .	12
6. Validation Interface Operations . . . . .	14
6.1. Child to Parent Operations . . . . .	15
6.2. Parent to Child Operations . . . . .	16
6.2.1. Address validation and registration . . . . .	16
6.2.2. Registration update . . . . .	17
6.3. Registration deletion . . . . .	18
7. Security Considerations . . . . .	18
8. IANA Considerations . . . . .	18
9. Acknowledgments . . . . .	19
10. References . . . . .	19
10.1. Normative References . . . . .	19
10.2. Informative References . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction

"Neighbor Discovery for IP version 6" [RFC4861] (IPv6 ND) relies heavily on multicast signaling messages on the local Link. Conceptually, multicast is supposed to avoid broadcast messages, but, in most practical cases, its operation at the link level is that of a broadcast. This did not matter much at the time ND was originally designed, when an Ethernet network was more or less a single shared wire, but since then, large scale switched fabrics, low-power sleeping devices, mobile wireless devices and virtual machines have changed the landscape dramatically.

The overhead of multicast in IPv6 ND has become significant and is now a major annoyance in multiple scenarios, in particular for wireless nodes. With WIFI, a multicast message will consume the wireless link on all Access Points around a switched fabric and will be transmitted at the lowest speed possible in order to ensure the maximum reception by all other wireless nodes. This means that in an environment where bandwidth is scarce, a single multicast packet may consume the bandwidth for hundreds of unicast packets. Sadly, IPv6 ND is a major source of multicast messages in wireless devices, since such messages are triggered each time a wireless device changes its point of attachment.

A similar situation can be seen in a datacenter, where Virtual Machine (VM) mobility also triggers floods of multicast messages,

which become a major hassle as the number of VMs grows to the tens of thousands and above. At the IETF, a Working Group was created to discuss Address Resolution in Massive Datacenters (ARMD), but the work did not go to completion. The problem with IPv6 ND multicast is still present, only getting worse as the scale and degree of mobility augments with the massive introduction of new mobile devices such as virtualized appliances, IoT and BYOD.

At the same time, the need to better control the ownership, utilization and location of IP addresses has become predominant in managed networks. The Source-Address Validation Improvements (SAVI) Working Group has proposed methods to locate, validate the ownership, and police the utilization of IPv6 addresses by snooping IPv6 ND and DHCP operations. But snooping requires being on the path of the protocols and is limited in particular in and for unicast responses.

Mobile nodes such as BYOD may change their point of attachment in the network but an eventual renumbering can be disruptive to existing connections. Virtual devices - typically VMs in a datacenter - also move though in a different fashion, from a physical device to the next. In any case, the need to maintain a same IPv6 address across movements implies the creation of very large, eventually multi-link, subnets. In such a large subnet, it might be difficult with the existing protocols to differentiate duplication from a rapid sequence of movements. And if it is indeed a sequence of movements, then it might be difficult to select the freshest information, and additional signaling is required to obtain the actual location of an address in a deterministic fashion.

In a modern managed switched fabric, a number of devices host IPv6 State Maintaining Entities (6SMEs) that hold Stateful Address Identification and Location (SAIL) information about the entity that owns an IPv6 address. A 6SME needs to reascertain periodically the state that it maintains and eliminate stale information. It is of common interest between all 6SMEs to share their information and help one another learn new state, update existing state and remove stale state rapidly. A Binding Table maintained by a secured registration protocol is certainly a more robust basis for 6SME activity than a classical IPv6 NDP [RFC4861] Neighbor Cache management coupled with protocol snooping as currently found with SAVI [RFC6620].

Mobile IPv6 [RFC6275] introduced such a registration protocol to maintain a tunnel and enable an IPv6 ND proxy operation over a Home Network. Applied to IPv6 Neighbor Discovery, the registration model balances the benefits of distributed Stateless Address AutoConfiguration (SLAAC) [RFC4862] for scalability and autonomic behaviours with the capability to reject or recuse an autoconfigured address on an exception basis - based for instance on administrative

policies -, which is a desired feature for managed networks that classically are operated with DHCPv6 [RFC3115]. In that sense, the ND registration allows a scalable hybrid of managed and non-managed networks while minimizing the total number of multicast messages between hosts, as well as between hosts and routers.

An IPv6 ND registration mechanism was standardized as Neighbor Discovery Optimization for Low-power and Lossy Networks [RFC6775]. The host to SME router operation is generalized by wireless ND [I-D.chakrabarti-nordmark-6man-efficient-nd] for devices that are not necessarily attached to a LLN but may still benefit from registration. [RFC6775] also introduces a protocol between SMEs based on new ICMP messages. This draft extends that model in order to allow for a distributed, eventually hierarchical set of SMEs to share and maintain SAIL states.

## 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].

Readers are expected to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861], "IPv6 Stateless Address Autoconfiguration" [RFC4862], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], Neighbor Discovery Optimization for Low-power and Lossy Networks [RFC6775] and "Multi-link Subnet Support in IPv6" [I-D.ietf-ipv6-multilink-subnets].

Readers may benefit from reading the "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550] specification; "Multi-Link Subnet Issues" [RFC4903]; "Mobility Support in IPv6" [RFC6275]; "Neighbor Discovery Proxies (ND Proxy)" [RFC4389]; "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses" [RFC6620]; and "Optimistic Duplicate Address Detection" [RFC4429] prior to this specification for a clear understanding of the art in ND-proxying and binding.

Additionally, this document uses terminology from [I-D.ietf-roll-terminology], and reuses or introduces the following terminology:

6LoWPAN Router (6LR): Please refer to [RFC6775].

6LoWPAN Border Router (6LBR): Please refer to [RFC6775].

DAR and DAC messages: Please refer to [RFC6775].

Multi-link subnet: Please refer to [I-D.ietf-ipv6-multilink-subnets].

**Backbone:** A link that forms the core of a multi-link subnet. All ARs and legacy devices are connected to the Backbone and classical ND operation ensure connectivity over that link.

**attached link:** An abstract link in a multi-link subnet other than the backbone. That link is classically not implemented as a fixed wire, and may only provide non-continuous connectivity, in particular with support for mobility. An attached link may be for instance a classical WiFi (IEEE802.11) link, a link in a wireless mesh network, or an overlay tunnel.

**attached node:** A device in a multi-link subnet that is not directly connected to the Backbone but reachable via an attached link.

**Backbone Router (BBR):** A BBR is an IPv6 router that connects attached links to a Backbone link and enables the connectivity of an attached node by proxying IPv6 NDP over the Backbone for that node, either with the node MAC address, or its own. The BBR is a 6SME that can obtain attachment states from attached nodes by different methods, for instance by snooping IPv6 NDP or DHCPv6, by learning host routes acting as a RPL root, by accepting ND registrations acting as an AR or an IR.

**Stateful Address Identification and Location (SAIL):** As opposed to a cache entry, a SAIL state is Stateful in that it is obtained and maintained through a (secured) registration mechanism. A SAIL state may include for instance a secured identification of the owner of the address (e.g. a trusted token, a public key or a certificate), the position of the IPv6 address in the network (e.g. VLAN, Access Switch or Access Point), or the mapping of the IPv6 address with a MAC address. Some of this information may be stable, for instance a owner Identification, while other may be transient, for instance the Access Point identifier in a mobility scenario or the MAC address mapping in the case of NDP proxy operations.

**State Maintaining Entity (SME):** An entity that hold SAIL information. SMEs are implemented in devices such as security appliances such as Network Access Controllers (NACs), SAVI switches that protect the ownership of an IPv6 address and control the ingress of the network, Wireless LAN Controllers (WLCs) that terminate a CAPWAP tunnel and must rapidly re-enable reachability for a mobile device both at layer 2 and layer 3, as well as overlay terminators such as used for network virtualization (NVO3). Overlay termination may operate both at layer 2 or layer 3, and may be found in data centers and enterprise networks to support mobility or extend the layer 2 fabric over a Layer 3 infrastructure, as well as in Service Provider networks to support IPv6 mobility.



**Binding:** The association of an IPv6 address with some SAIL state. A registrar maintains a binding table to store and query such associations.

**Registering Node (RN):** An IPv6 node that obtains and retains ownership of an IPv6 address through the process of IPv6 ND registration.

**Authoritative Registrar (AR):** A 6SME that stores authoritative information about a registration. An AR is the reference for address and SAIL state binding within its domain of authority, e.g. a specific subset of addresses within a subnet. There can be multiple ARs in a subnet and domains may overlap for redundancy and balancing.

**Intermediate Registrar (IR):** A 6SME that stores information about a registration as part of the registration flow. IRs form a directed acyclic graph (DAG) that is directed towards ARs. A registration from an RN will be addressed to an IR and will follow the IR DAG till it reaches a node that can grant the ownership, typically a AR.

**Registration Interface (RIF):** The interface between an RN and an IR. The RIF is typically implemented using Wireless ND, but can also be implicitly implemented by snooping IPv6 ND, e.g. as suggested by SAVI.

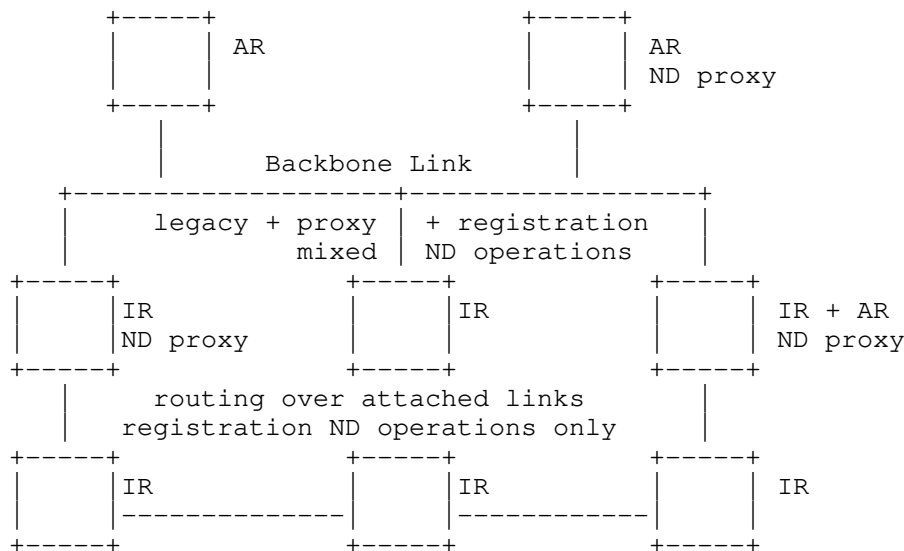
**Validation Interface (VIF):** The interface between a child IR and a parent IR or AR. The VIF is typically implemented using this specification which extends the DAR and DAC messages as defined in [RFC6775].

**Determination Interface (DIF):** The interface between ARs. It can be implemented using LISP, routing protocol extensions, or using IPv6 ND proxy extensions such as suggested by [I-D.thubert-6lowpan-backbone-router] .

### 3. Overview

The scope of this draft is a potentially large and potentially multi-link subnet [I-D.ietf-ipv6-multilink-subnets] formed by a high speed Backbone that federates additional links of heterogeneous MAC/PHY types, for instance an Ethernet switched fabric federating a Route-Over mesh that may interconnect thousands of LLN devices over multiple wireless hops.

In order to avoid floods of multicast packets inherent to a reactive discovery, a node - referred to as a Registering Node (RN) - needs to claim its addresses proactively, binding them with its location in the network and a Lower Layer Address (LLA), over confirmed exchanges with a neighborhood Intermediate Registrar (IR).

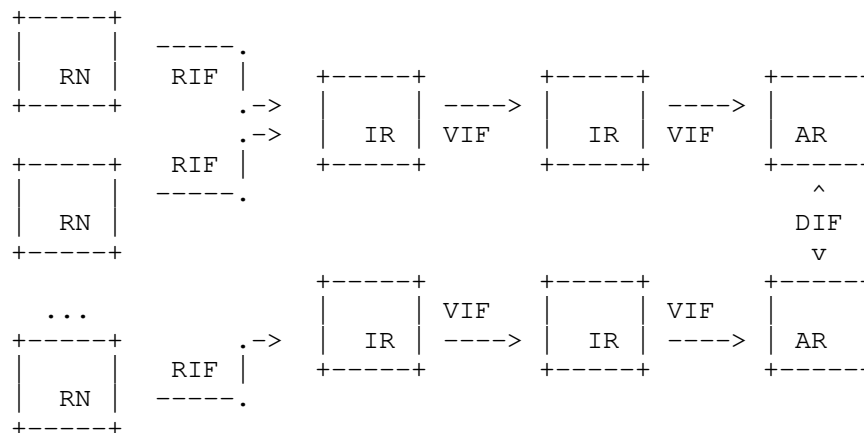


The IR may confirm the claim with an Authoritative Registrar (AR), eventually over a graph of other IRs. If the ownership is granted, the RN may use the address for a lifetime that is associated with the grant, at which point it will need to reconfirm the address by registering again.

In the case of a meshed 6LoWPAN [RFC6282] [RFC6775] LLN topology, the neighborhood IR is a 6LoWPAN Router (6LR) and the AR is a 6LoWPAN Border Router (6LBR). When the topology grows, the IPv6 ND registration model as described in [RFC6775], with a single AR (the 6LBR), may not scale.

With this draft, all registrars maintain an abstract Binding Table of their registered addresses. The Binding Table operates as a distributed database of information related to addresses whether the address owner reside on the attached links or on the Backbone. ARs use extensions to the Neighbor Discovery Protocol to exchange that information across the Backbone either in the classical ND reactive fashion, or through a new pub/sub mechanism that is introduced by this specification.

With this specification, multiple IRs and one AR can be deployed so as to scale the IPv6 ND registration model yet avoiding any broadcast beyond one Layer-2 hop; IRs cover the whole multi-link subnet in a fashion that any node in the network has at least one neighborhood IR one Layer-2 hop away so it may perform an NDP registration with that IR using link local addresses regardless of the link type, wired or wireless.



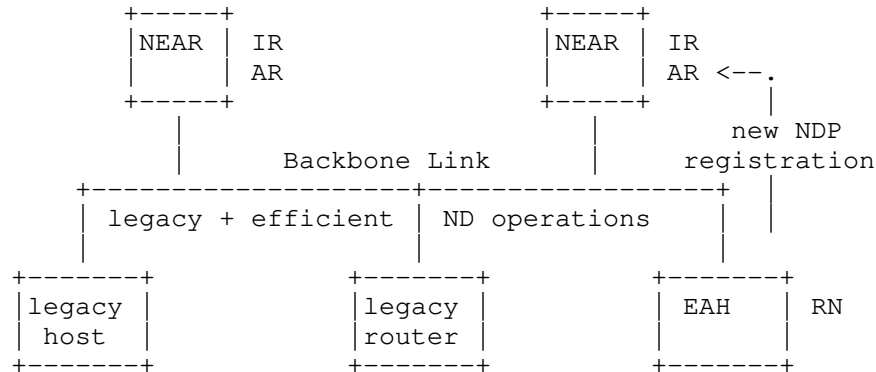
IRs and ARs form a DAG directed towards the AR(s); but how this DAG is set up is out of scope for this specification.

If more than one AR is deployed, a strategy (e.g. a distributed hash table (DHT) or a DNS-like hierarchy) and a method to distribute and synchronize the individual domains of authority between ARs, must be put in place. Such method is out of scope for this document. In the case where an overlap of domain is acceptable, a protocol must be put in place between ARs so as to resolve conflicts, and clean up stale states. Such a protocol is out of scope for this document.

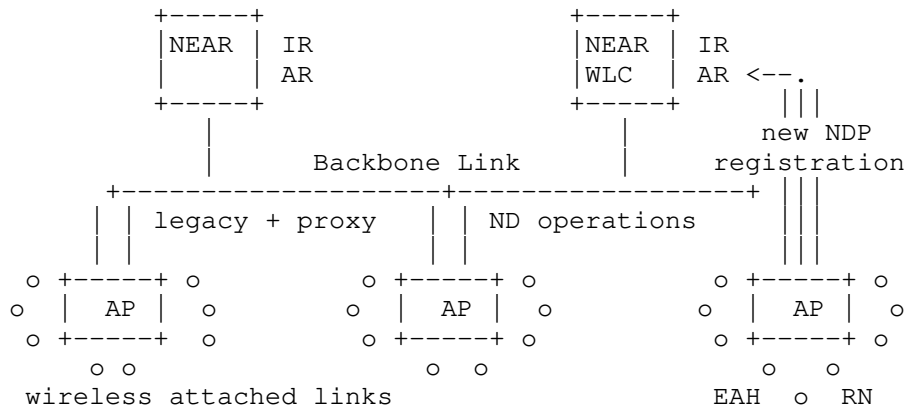
#### 4. General Context

##### 4.1. Efficient ND

[I-D.chakrabarti-nordmark-6man-efficient-nd] updates the specification of the RIF interface between the RN and the IR, that was initially defined in [RFC6775] for 6LoWPAN devices. The draft details the operation of a IPv6 ND-efficiency-aware Router (NEAR), that is the neighborhood IR to which an RN, which is called a Efficiency-Aware Host (EAH), registers. A NEAR is also an AR as it has the exclusive authority on the bindings for its registered EAHs.



In the case of a WIFI connection, the NEAR is a BBR for the wireless device, and may be collocated with a standalone AP or a Wireless LAN Controller.



This specification extends [I-D.chakrabarti-nordmark-6man-efficient-nd], by allowing the separation of IR and AR functions, which are collapsed inside the NEAR. This draft introduces the VIF interface between IRs, and between IRs and ARs, as well as the DIF interface between ARs with potentially overlapping domain, and other 6SMEs.

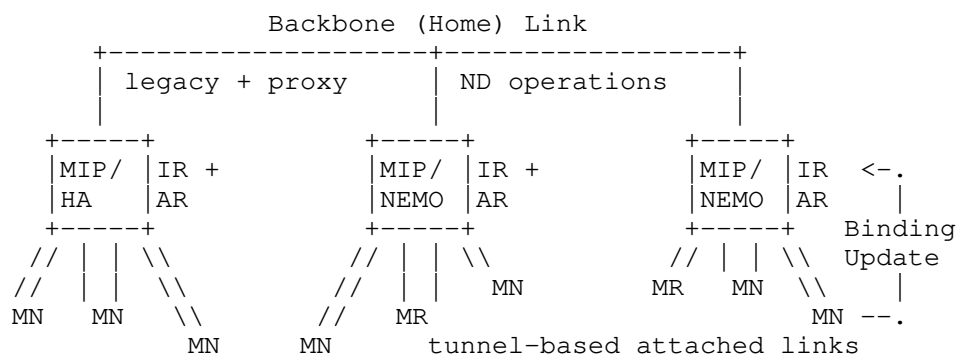
For the purpose of the new NDP registration, [I-D.chakrabarti-nordmark-6man-efficient-nd] defines an extended ARO option that is advertised by an EAH. The new ARO option includes a sequence counter called TID that enables a short term freshness assertion between rapid re-registrations of a mobile device, and a unique ID that is used for the Duplicate Address Detection (DAD).

A 6SME may maintain a state for a longer time than covered by the ARO TID, so a coarse age information is needed to compare old state information over the VIF and DIF interfaces. Additionally, the 6SME may qualify information with additional metadata to help resolve conflicts. For instance, in the case of a duplicated IPv6 address, additional meta information such as the protocol that was used to establish the state (SLAAC vs. DHCPv6), a device type (trusted server vs. unknown host), or a Secure ND cryptographic address ownership validation ([RFC3971], [RFC3972]) can help protect the address where it is assigned in a more trusted fashion even if a rogue managed to grab the address while the more trusted owner was not able to defend it.

This specification proposes a new ND option that contains such information and complements the information in the ARO option for use on the VIF and DIF interfaces.

#### 4.2. Proxying classical ND

A 6SME such as an IR, a AR or a BBR may proxy classical IPv6 NDP [RFC4861] on behalf of a virtual, a wireless, or a low power device so as to offload the device, to dampen the network load such as induced by the multicast operations of the proxied protocol, or simply to attract over the backbone and then relay its traffic to the mobile or sleeping device even if the device is not reachable at that particular time.



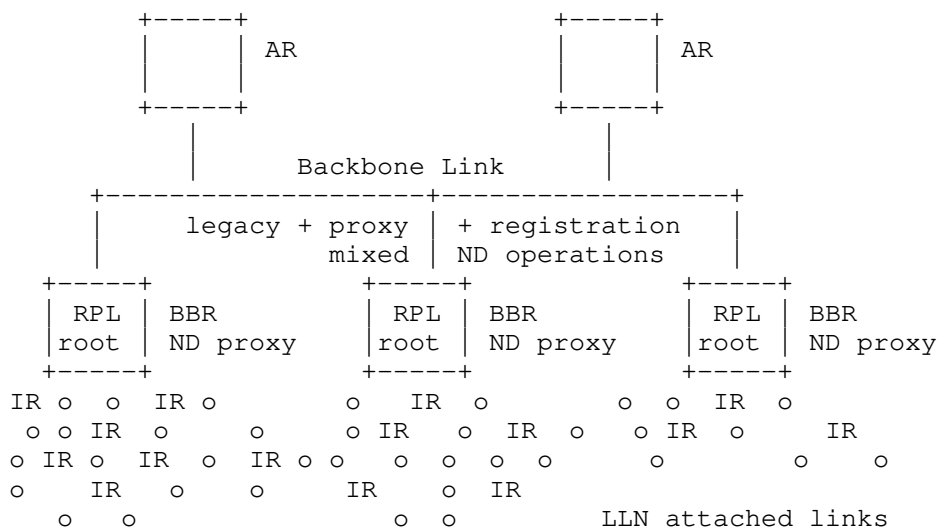
The 6SME may perform the proxy operation on behalf of an original device using the original device LLA, or may proxy the Layer-2 information with their own LLA and either rewrite it later in the packets, or route the packet again over an attached link, as exemplified by a MIPv6 [RFC6275] or a NEMO [RFC3963] Home Agent (HA).

The HA is authoritative for any Mobile Node (MN) that successfully registers to it through a Binding Update/Ack flow and its domain of authority is the subnet(s) on the Home Link, potentially overlapping with other HAs. ND proxy operations are used over the Home Link to resolve collisions.

The MN is thus an RN and the HA cumulates IR, AR and proxy functionalities. With NEMO [RFC3963], the model is conserved but the RN is now a Mobile Router that registers a prefix together with its own address, so the operation in the Backbone link is a mix of ND proxy and routing. Network Mobility Home Network Models [RFC3963] provides more information on that model.

### 4.3. Federating Large LLNs

In the case of a large multi-link subnet, this specification expects that a Backbone link is deployed to interconnect all the ARs and legacy NDP devices. Each interconnected attached link, whether it is a WIFI access, a mesh network, a 6LoWPAN/RPL LLN or an overlaid tunnel, is anchored to the Backbone at a Backbone Router (BBR). The BBRs interconnect the multi-link subnet over the Backbone Link at layer 3, enabling connectivity within the subnet over IP.



If the LLN uses a Route-Over model based on RPL [RFC6550], the Backbone Router (BBR) that connects the LLN to the Backbone is the root for the RPL LLN. The BBR proxies the ND protocol over the backbone for the addresses that it has learnt through RPL as host routes, using its own LLA and location to attract traffic for the attached nodes and route it over the LLN.

Over the Backbone, this setup implements the "simple scenario" in [I-D.ietf-ipv6-multilink-subnets] whereby the router acts "as an asymmetric Neighbor Discovery proxy"; over the RPL-based LLN mesh, the setup implements the more "complex scenario" whereby "an arbitrary topology exists, and routers within the subnet communicate using some means of exchanging host routes".

[I-D.thubert-6lowpan-backbone-router] describes this mixed model, and how a Backbone Router perform ND proxy operation for their attached nodes over the The Backbone Link regardless of the mode of registration for the attached nodes. The operation described in the draft is compatible with that of a MIPv6 [RFC6275] Home Agent. This enables mobility support for wireless attached nodes that would move outside of the network delimited by the Backbone link and back. In any case, it is expected that the registration provides a sequence counter, a lifetime and a unique identifier of the attached nodes in such a fashion that they can be matched or compared across protocols.

This specifications indicates how the new ND option can be used in conjunction with ND proxy techniques over the Backbone to implement the DIF interface.

## 5. New types and formats

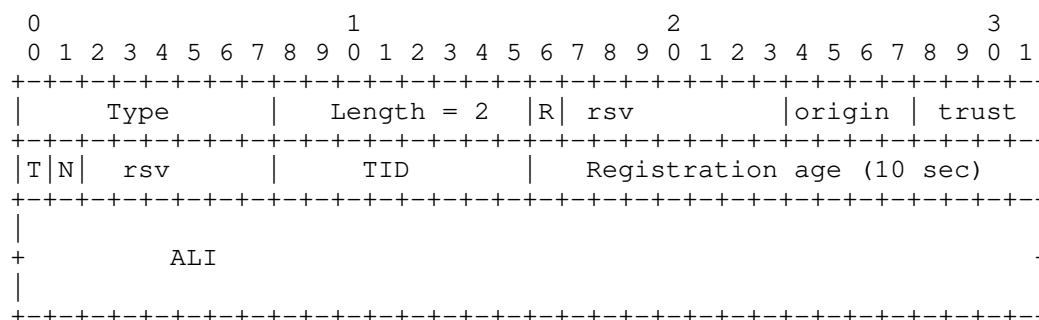
This section introduces message formats for all messages used in this specification.

The specification expects that the protocol running on the LLN can provide a sequence number called Transaction ID (TID) that is associated to the registration. When a node registers to multiple registrars (IRs or ARs), it is expected that the same TID is used, to enable the registrar to correlate the registrations as being a single one, and differentiate that situation from a movement. Otherwise, the resolution makes it so that only the most recent registration was perceived from the highest TID is kept.

The specification expects that the protocol running on the LLN can provide a unique ID for the owner of the address that is being registered. The Owner Unique ID enables to differentiate a duplicate registration from a double registration. In case of a duplicate, the last registration loses. The Owner Unique ID can be as simple as a EUI-64 burnin address, if the device manufacturer is convinced that there can not be a manuf error that would cause duplicate EUI64

- o Burn in address.
- o configured address, id, security keys...
- o (pseudo) Random number, radio link metrics ...

The unique ID and the sequence number are placed in a new ND option that is used by the Backbone Routers over the Backbone link to detect duplicates and movements. The option format is as follows:



```
0 - SLACC: Address was auto-configured on the RN [RF4862]
1 - DHCP: Address was assigned to the RN by DHCP
2 - LOCAL: Address was manually configured on the RN
```



3 - STATIC: Address was manually configured on the IR as a downstream address, i.e. an address assigned to a downstream node

4 - DATA: Address was gleaned on the first IR as the source of a data packet

trust : 4-bit unsigned integer. Indicates the level of trust the attaching node place in the entry

0 - NO\_TRUST: No particular trust associated with the entry

1 - L2L3\_MATCH: The layer-2 source MAC and Link-layer-Address claimed in the registration match

2 - TRUSTED\_BY\_POLICY: The address is trusted by policy on the attaching node

3 - AUTHENTICATED The address has been authenticated by a cryptographic protocol (CGA, etc.)>

T: One bit flag. Set if the next octet is a used as a TID following follow section 7 of RPL [RFC6550] for sequence counters. If the bit is not set, a unsigned char is expected.

N: One bit flag. Set if the device moved. If not set, the router will refrain from sending gratuitous NA(O) over the backbone, for instance after the DAD operation upon entry creation.

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

TID: 1-byte integer; a transaction id that is maintained by the device and incremented with each transaction. it is recommended that the device maintains the TID in a persistent storage. The TID is incremented at each registration.

Registration Age: 2-byte integer; the duration since the last update of TID in units of 10 seconds.

Attaching Location Identifier: A locally unique identifier for the IR interface attaching the registering host.

## 6. Validation Interface Operations

The Validation Interface (VIF) is the interface between a child Intermediate Registrar (IR) and a parent registrar, whether an Intermediate Registrar or Authoritative Registrar (AR). An IR parent or upstream chain is defined as the set of IRs along the DAG starting from this IR, directed to (and including) the AR. An IR child or downstream chain is defined as the set of IRs from this IR an entry was learnt from including the IR attaching to the RN. The goal of VIF is to perform one or several of the followings:

- o Validate a new registration along the parent chain
- o Record a registration along the parent chain
- o Update a registration along the parent chain
- o Cancel a registration along the parent chain
- o Delete a record along the parent chain
- o Delete a record along the child chain

#### 6.1. Child to Parent Operations

The first IR in the chain (attached to the RN) initiates validation and registration of an address registered by the RN or snooped on the interface attaching the RN to the node, by building a DAR message, including a SLLA and a SAIL option where:

- o Source of the DAR is an IP address of the IR interface to the next IR in the chain.
- o Destination of the DAR is an IP address of the next IR.
- o R bit set to zero. If the IR acts as an RN, the the bit is set to 1
- o Origin can take any of the values defined, based on how the address was assigned
- o If the registration was received from the RN (or gleaned) on an IR interface administratively trusted, the field "trust" is set to TRUSTED\_BY\_POLICY. Otherwise, if the registration carried CGA [RFC3971] credential that the IR successfully verified, the field "trust" is set to AUTHENTICATED. Otherwise, if the source mac of the registration message received by the IR is identical to the Link-Layer Address provided by the message in the SLLA option, the trust field is set to L2L3\_MATCH. Otherwise, it is set to NO\_TRUST.
- o An SLLA option MUST be included in the DAR message along with the SAIL option, that contains the Link-Layer address bound to the IP address bein registered.

While waiting for a response, a TENTATIVE entry is created on the IR. Several attributes are stored next to the entry: the EUI64 provided by the RN, Link Layer Address provided in the SLLA option, the origin and trust values (computed by the IR, based on the registration, and local policies), the ALI the registration was received from, the lifetime. In the absence of a DAC response, DAR messages sent in the context of VIF from the IR are retransmitted after 250ms [DAR\_INTERVAL], up to 2 times [MAX\_DAR\_RETRANSMIT]. Upon receiving a negative response (duplicate address status) or when the maximum retransmit is exhausted, the entry is removed from the IR.

The IR can also initiate update and delete operations. An update is no different from an address validation: the DAR will carry the same address and EUI-64 as the one provided in a previous validation, while any other attribute such as SLLA option, Lifetime, Origin, Trust or ALI will eventually be different from the value previously provided

In order to cancel a registration, a DAR is sent, with a lifetime set to zero. It should carry a SAIL option to allow the receiving IR or AR to validate the delete.

## 6.2. Parent to Child Operations

### 6.2.1. Address validation and registration

Upon receiving a DAR message with a SAIL option, an IR will lookup in the local table to verify whether the address already exist.

If it does not exist, two cases arise.

1. The IR is not an AR. It creates the entry as "TENTATIVE", together with attributes such as EUI64, IR address it came from, origin and trust values. It then builds and sends a DAR message sourced with one address of its interface to the next IR, set destination address to the next IR address, sets the R bit to 1 and copies all other fields from the received DAR. It also starts a 250ms TENTATIVE\_TIMER timer of 250ms [DAR\_INTERVAL]. Should this timer expire, the DAR is re-transmitted up to 2 times, then the entry is deleted. If a negative response (DAC with status 1) is received, a DAC with status 1 is sent to the downstream IR, and the TENTATIVE entry is deleted. If a positive response (DAC with status 0) is received, the timer TENTATIVE\_TIMER is stopped, the entry state moved to REACHABLE and a DAC with status 0 is sent to downstream IR.

2. The IR is also the AR: it queries other ARs over the DIF interface. While waiting for a response, it may create the entry in "TENTATIVE" state. Upon confirmation from another AR that the entry exist elsewhere, the entry in TENTATIVE is deleted, and a DAC message is sent back to the source of the DAR message (previous IR), with a status set to 1 (Duplicate address). Upon receiving this DAC, each downstream IR deletes its own TENTATIVE entry, and sends a DAC, status 1, to the next child IR until it reaches the IR attaching the RN, which builds an NA with ARO option, and status set to duplicate address. If the DIF interface returns no conflict on the address, the entry state is moved to REACHABLE, and a DAC with status 0 is sent to the downstream IRs which move their TENTATIVE entry to REACHABLE. When the DAC reaches the attaching IR, it send an NA with ARO option, status 0 to the RN.

If the same address carried in the DAR exist on one of the IR or the AR, with a different EUI-64 interface identifier, the two entries attributes are compared. A trustlevel value is computed for each entry (as a function of the trust value, the origin and the R bit). The two trustlevel values are compared numerically as follows:

1. If the trustlevel of the existing entry is bigger or equal than the one carried by the DAR, the DAR is not propagated, and a DAC with status 1 (duplicate address) is sent back to downstream IR, up to the attaching IR which sends a NA with an ARO option, status 1 to the RN.
2. If the trustlevel of the existing entry is strictly smaller than the one carried by the DAR, it replaces it, and the DAR is propagated towards the upstream IR up to the AR. Again, DAR follow the rule of hop-by-hop retransmission and acknowledgment already described.
3. At the same time, if the entry being replaced was associated with a different IR than the one this DAR came from, another DAR, with the previous EUI64 value, and a lifetime set to zero is sent to downstream IR the previous registration came from. This message causes the downstream IR to remove the entry, provided that the EUI64 match, to build and send a DAR to the next IR, and to acknowledge the deletion with a DAC, status 0. If the EUI64 don't match, it means the entry has already been replaced, and the DAR need not to be propagated from this IR. DAR retransmission follow the same pattern already described. DAC are not propagated. Upon receiving the DAR with lifetime set to zero, the attaching IR sends an unsolicited NA to the RN with an ARO option, status 1 (duplicate address).

#### 6.2.2. Registration update

An update message is a DAR that carries an address and EUI64 interface identifier matching an IR or AR table entry, and at least one of the following field different from the previously registered value: LifeTime, Origin, trust, ALI or IR child. Upon receiving an update, the IR or AR updates its entry and propagate to the upstream IR or AR. The AR will in return send a DAC message with a status of 0 to acknowledge the update.

If the attribute being updated is the IR address the DAR is coming from (child IR), the host has moved to a different downstream IR chain, and the entry along the previous chain must be cleaned up. A DAR message, with a lifetime set to zero is sent (and retried if not acknowledged) to the old downstream IR. This message causes the downstream IR to remove the entry. The downstream IR should propagate the DAR to the next IR in chain, and acknowledge it with a DAC.

### 6.3. Registration deletion

A registration deletion can come from the IR attaching to the RN, because the RN left the link, from any IR as the result of an administrative action, or from the AR because the lifetime has expired or again following an administrative action. In all cases, a DAR message with a lifetime set to zero is sent either upstream or downstream, retried and acknowledged at each hop along the chain, if necessary. When the deletion is initiated on the IR attaching to the RN, a SAIL option MUST be provided to enable any upstream registrar to verify that the deletion is coming from the location the RN was attached to. For deletion following the parent chain, the ALI value carried in the SAIL option is compared with the ALI value registered for this address, and entry is deleted if the two match. For deletion following the child chain, this check is not required. Upon deleting the entry, the IR builds and sends a DAC to acknowledge the deletion, then build and send a DAR to propagate the deletion, downstream or upstream.

## 7. Security Considerations

This specification expects that the link layer is sufficiently protected, either by means of physical or IP security for the Backbone Link or MAC sublayer cryptography. In particular, it is expected that the LLN MAC provides secure unicast to/from the Backbone Router and secure BBroadcast from the Backbone Router in a way that prevents tempering with or replaying the RA messages.

The use of EUI-64 for forming the Interface ID in the link local address prevents the usage of Secure ND ([RFC3971] and [RFC3972]) and address privacy techniques. Considering the envisioned deployments and the MAC layer security applied, this is not considered an issue at this time.

## 8. IANA Considerations

A new type is requested for an ND option.

## 9. Acknowledgments

TBD

## 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.E. and R.M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.
- [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.
- [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.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J. and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP. and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.
- [RFC6620] Nordmark, E., Bagnulo, M. and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, May 2012.

- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E. and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

## 10.2. Informative References

- [I-D.chakrabarti-nordmark-6man-efficient-nd]  
Chakrabarti, S., Nordmark, E. and M. Wasserman,  
"Efficiency aware IPv6 Neighbor Discovery Optimizations",  
Internet-Draft draft-chakrabarti-nordmark-6man-efficient-  
nd-01, November 2012.
- [I-D.ietf-ipv6-multilink-subnets]  
Thaler, D. and C. Huitema, "Multi-link Subnet Support in  
IPv6", Internet-Draft draft-ietf-ipv6-multilink-  
subnets-00, July 2002.
- [I-D.ietf-roll-terminology]  
Vasseur, J., "Terminology in Low power And Lossy  
Networks", Internet-Draft draft-ietf-roll-terminology-12,  
March 2013.
- [I-D.thubert-6lowpan-backbone-router]  
Thubert, P., "6LoWPAN Backbone Router", Internet-Draft  
draft-thubert-6lowpan-backbone-router-03, February 2013.
- [RFC3115] Dommety, G. and K. Leung, "Mobile IP Vendor/Organization-  
Specific Extensions", RFC 3115, April 2001.
- [RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A. and P.  
Thubert, "Network Mobility (NEMO) Basic Support Protocol",  
RFC 3963, January 2005.
- [RFC3971] Arkko, J., Kempf, J., Zill, B. and P. Nikander, "SEcure  
Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)",  
RFC 3972, March 2005.
- [RFC4389] Thaler, D., Talwar, M. and C. Patel, "Neighbor Discovery  
Proxies (ND Proxy)", RFC 4389, April 2006.
- [RFC4887] Thubert, P., Wakikawa, R. and V. Devarapalli, "Network  
Mobility Home Network Models", RFC 4887, July 2007.
- [RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, June  
2007.
- [RFC4919] Kushalnagar, N., Montenegro, G. and C. Schumacher, "IPv6  
over Low-Power Wireless Personal Area Networks (6LoWPANs):  
Overview, Assumptions, Problem Statement, and Goals", RFC  
4919, August 2007.

[RFC6275] Perkins, C., Johnson, D. and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

Authors' Addresses

Pascal Thubert  
Cisco Systems  
Village d'Entreprises Green Side  
400, Avenue de Roumanille  
Batiment T3  
Biot - Sophia Antipolis, 06410  
FRANCE

Phone: +33 4 97 23 26 34  
Email: pthubert@cisco.com

Eric Levy-Abegnoli  
Cisco Systems  
Village d'Entreprises Green Side  
400, Avenue de Roumanille  
Batiment T3  
Biot - Sophia Antipolis, 06410  
FRANCE

Phone: +33 4 97 23 26 34  
Email: elevyabe@cisco.com



6MAN  
Internet-Draft  
Intended status: Best Current Practice  
Expires: December 17, 2015

W. Kumari  
Google  
J. Jaeggli  
Zynga  
R. Bonica  
Juniper Networks  
J. Linkova  
Google  
June 15, 2015

Operational Issues Associated With Long IPv6 Header Chains  
draft-wkumari-long-headers-03

Abstract

This memo specifies requirements for IPv6 forwarders as they process packets with long header chains. It also provides guidance for application developers whose applications might rely on long headers chains.

As background, this memo explains how many ASIC-based IPv6 forwarders process packets and why processing of packets with long header chains might be problematic.

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].

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 17, 2015.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	4
2. Forwarder Information Requirements . . . . .	4
3. Requirements For IPv6 Forwarders . . . . .	5
4. Recommendations For Application Developers . . . . .	7
5. IANA Considerations . . . . .	7
6. Security Considerations . . . . .	7
7. Acknowledgements . . . . .	8
8. References . . . . .	8
8.1. Normative References . . . . .	8
8.2. Informative References . . . . .	8
Appendix A. Changes / Author Notes. . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

IPv6 [RFC2460] forwarders can acquire information from the following sources:

- o The IPv6 header
- o One or more IPv6 extension headers
- o An upper-layer header

Section 2 of this document explains how IPv6 forwarders use information from the IPv6 header and IPv6 extension headers to provide traditional forwarding services. It also explains how IPv6 forwarders use information from the upper-layer header to provide enhanced forwarding services.

When a software-based forwarder processes an IPv6 datagram, it parses the header chain, regardless of its length, acquires the required information and makes a forwarding decision. Typically, software-based forwarders process a relatively small number of packets per second. Therefore, they can perform the above mentioned procedure within the constraints of their processing budget.

By contrast, ASIC-based forwarders process many more packets per second. In order to fulfill this requirement, ASIC-based forwarders copy a fixed number of bytes from the beginning of the packet to on-chip memory. Forwarders do this because they can access on-chip memory much more quickly than they can access off-chip memory. Once the beginning of the packet has been transferred to on-chip memory, subsequent processing can proceed very quickly.

The act of copying bytes from the beginning of a packet to on-chip memory consumes:

- o Processor cycles
- o On-chip memory
- o Wall-time

Therefore, the number of bytes copied to on-chip memory must be chosen wisely. If a forwarder copies more bytes than it needs, it wastes resources and adversely impacts performance. If it copies too few bytes, it may not have sufficient information to make a correct forwarding decision.

The IPv6 header chain is a variable-length data structure, whose size can exceed 64 kilobytes. However, packets with header chains exceeding 256 bytes are rarely observed on the Internet. Therefore, most ASIC-based forwarders copy a relatively small number of bytes from the beginning of a packet into on-chip memory. While this small number varies from platform to platform, it is generally much closer to 256 bytes than it is to 64 kilobytes.

IPv6 forwarders MUST behave in a predictable manner when they process a packet whose header chain length exceeds the number of bytes copied to on-chip memory. Section 3 of this memo defines required behaviors.

Application developers should be aware of how ASIC-based forwarders process packets with long extension header chains. Therefore, Section 4 of this document provides guidance to application developers.

### 1.1. Terminology

For the purposes of this document, the terms "header chain" and "upper-layer" header are used as defined in [RFC7112].

This document also introduces the following terms:

- o forwarding service - a service that accepts a packet from one interface and forwards it through another
- o traditional forwarding service - a forwarding service in which all parameters to the forwarding algorithm are drawn from the IPv6 header, the hop-by-hop extension header, and the routing extension header
- o enhanced forwarding service - a forwarding service in which parameters to the forwarding algorithm can be drawn from any portion of the IPv6 header chain

## 2. Forwarder Information Requirements

When an IPv6 forwarder provides traditional forwarding services, it extracts all information required by the forwarding algorithm from the IPv6 header, the hop-by-hop extension header (if present), and the routing extension header (if present). In the nominal case, the IPv6 header contains all information required by the forwarding algorithm. However, the hop-by-hop and routing extension headers can also impact forwarding behavior.

Section 4.2 of [RFC2460] explains how the hop-by-hop extension header impacts forwarding behavior. When the forwarder processes a hop-by-hop extension header, it examines each option contained by the header. If forwarder encounters an unrecognized hop-by-hop option, and the high-order bits of the option type are "00", the forwarder skips over the option and continues to process subsequent options. However, if an forwarder encounters an unrecognized option, and the high-order bits of the option type are "01", "10" or "11", the forwarder discards the packet.

Section 4.4 of [RFC2460] explains how the routing extension header impacts forwarding behavior. When the forwarder processes a packet whose destination address is local to itself, it scans the header chain, searching for a routing extension header. If the packet contains a routing extension header and the forwarder recognizes the routing header type, it processes the header. If the forwarder does not recognize the routing header type, the required behavior depends upon the Segments Left field. If the Segments Left field is equal to zero, the forwarder ignores the routing extension header. Otherwise,

the forwarder discards the packet. [RFC6275] and [RFC6554] describe currently defined routing extension header types.

Some IPv6 forwarders provide enhanced forwarding services, such as firewall filtering, rate limiting and load balancing. In order to provide these services, the forwarder requires access to an upper layer header. The following are examples of enhanced services that require the forwarder to examine the upper layer header:

- o Discard all packets directed to TCP port 25
- o Rate limit packets destined for a particular address whose payload is TCP and have the TCP SYN bit set
- o Load balance packets across parallel links so that all packet belonging to particular TCP session traverse the same link.

### 3. Requirements For IPv6 Forwarders

The following requirements apply to all IPv6 forwarders:

- o REQ-1: By default an IPv6 forwarder SHOULD NOT discard a valid packet because of its header chain length. However, the forwarder MAY support a configuration option that causes it to discard packets whose header chain length exceeds a specified value.
- o REQ-2: When processing packet that contains a hop-by-hop extension header, an IPv6 forwarder MUST process the entire hop-by-hop extension header, regardless of its length. The forwarder MUST process each option as specified in Section 4.2 of [RFC2460]. If an IPv6 forwarder is not able to process the entire hop-by-hop extension header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them. If an IPv6 forwarder is not able to process the entire hop-by-hop extension header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them.
- o REQ-3: When processing a packet whose destination address is local to itself, an IPv6 forwarder MUST scan the entire header chain, regardless of its length, in order to determine whether the packet contains a routing extension header. If the packet contains a routing extension header, the forwarder MUST process routing extension header as specified in Section 4.4 of [RFC2460]. If an IPv6 forwarder is not able to process the entire routing extension

header, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source. The forwarder MAY have a configurable policy for sending ICMPv6 messages such as rate limiting or completely disabling them.

The length of the IPv6 header plus the length of the hop-by-hop extension header can exceed the number of bytes that an ASIC-based forwarder copies into on-chip memory. Therefore, in order to support REQ-2, ASIC-based forwarders typically support a special processing mechanism for packets containing hop-by-hop extensions.

Also, the combined length of all headers preceding the routing extension header may exceed the number of bytes that an ASIC-based forwarder copies into on-chip memory. Therefore, in order to support REQ-3, ASIC-based forwarders typically support a special processing mechanism for packets whose IPv6 destination address is local to the forwarder. This forwarding mechanism is capable of processing the routing extension header, even if it begins beyond of the portion of the packet that was copied to on-chip memory.

The following requirements apply to IPv6 forwarders that provide enhanced forwarding services:

- o REQ-4: If a forwarder's ability to deliver enhanced services is limited in any way by extension header length, that limitation MUST be reflected in user documentation. For example, assume that a forwarder provides a load balancing service, and that it acquires information required by the service from the IPv6 header and the upper-layer header. If the service behaves in one manner when all required information is contained by the first N bytes of the header chain and in another manner when all required information is not contained by the first N bytes of the header chain, user documentation MUST reflect both behaviors as well as the value of N.
- o REQ-5: If a forwarder's ability to deliver an enhanced service is limited by extension header length, the policy specification language used to configure the enhanced service MUST be sufficiently robust to address the limitation. For example, assume that the forwarder provides a firewall service. The firewall service is capable of filtering packets directed to a particular TCP port, but only if the TCP header is contained by the first N bytes of the header chain. In this case, it MUST be possible to configure one policy for packets directed to the specified port, another policy for packet not directed to the specified port, and a third policy for packets whose TCP destination port is unknown.

#### 4. Recommendations For Application Developers

Applications developers should be aware that many ISPs and enterprises filter or severely rate limit packets containing long header chains. They do this because of limitations imposed by the ASIC-based forwarders deployed at their edges. ISPs and enterprises accept these limitations as part of an engineering trade off, in which high-speed forwarding is achieved at the cost of limiting enhanced services for packets with long extension headers.

For example, assume that an enterprise deploys the following firewall filtering policy at its edge:

- o Permit all packets whose destination is TCP port 80
- o Discard all packets whose destination is not TCP port 80
- o Discard all packets whose header chain is so long that TCP port information is not accessible to the filtering function

In this case, the enterprise discards all packets whose destination cannot be determined by the filtering function.

Aside from the issue of header chain length, operators may filter packets containing extension headers that may either compromise the network's security posture or require inordinate processing resources.

This memo does not specify a maximum header chain length. However, this memo does note that at the time of its publication, the number of bytes that ASIC-based forwarders copy from the beginning of a packet to on-chip memory varies from platform to platform. Typical platforms copy between 128 and 384 bytes. Therefore, application developers should avoid sending packets whose header chain length is in that range, unless they have some assurance that their packets will not be discarded.

#### 5. IANA Considerations

This document makes no requests of the IANA

#### 6. Security Considerations

TBD

## 7. Acknowledgements

The authors wish to thank Paul Hoffman, KK and Fernando Gont. The authors also express their gratitude to an anonymous donor, without whom this document would not have been written.

## 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.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, January 2014.

### 8.2. Informative References

- [RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, March 2012.

## Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication ]

Template to -00

- o Initial submission.

- o

-00 to -01

- o Added maximum header chain recommendation.



- o Rewrite the forwarding description.
- 02 to -03
- o Updating REQ2 and REQ3 with sending ICMPv6 messages part.

Authors' Addresses

Warren Kumari  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: warren@kumari.net

Joel Jaeggli  
Zynga  
675 East Middlefield  
Mountain View, CA  
USA

Email: jjaeggli@zynga.com

Ronald P Bonica  
Juniper Networks  
2251 Corporate Park Drive  
Herndon, VA  
USA

Email: rbonica@juniper.net

J. Linkova  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
USA

Email: furry@google.com