INTAREA WG
Internet-Draft
Updates: 4861 (if approved)
Intended status: Standards Track
Expires: May 3, 2012

S. Chakrabarti
Ericsson
E. Nordmark
Cisco Systems
M. Wasserman
Painless Security

October 31, 2011

Energy Aware IPv6 Neighbor Discovery Optimizations
draft-chakrabarti-nordmark-energy-aware-nd-01

Abstract

IPv6 Neighbor Discovery (RFC 4861) protocol has been designed for neighbor’s address resolution, unreachability detection, address autoconfiguration, router advertisement and solicitation. With the progress of Internet adoption on various industries including home, wireless and machine-to-machine communications, there is a desire for optimizing legacy IPv6 Neighbor Discovery protocol for energy-efficient networks and nodes. Research indicates that often networked- nodes require more energy than stand-alone nodes because a node’s energy usage depends on network messages it receives and responds. While reducing energy consumption is essential for battery operated nodes in some machines, saving energy actually a cost factor in business in general as the explosion of more device usage is leading to usage of more servers and network infrastructure in all sectors of the society and business. This document describes a method of optimizations by reducing periodic multicast messages, frequent Neighbor Solicitation messages and discusses interoperability with legacy IPv6 nodes. This document also addresses the ND denial of service issues by introducing node Registration procedure.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

Thus optimizing the regular IPv6 Neighbor Discovery [ND] to minimize total number of related signaling messages without losing generality of Neighbor Discovery and autoconfiguration and making host and router communication reliable, is desirable in any IPv6 energy-aware networks such as Home or Enterprise building networks and as well as Data Centers.

The goal of this document is to provide energy-aware and optimized Neighbor Discovery Protocols in the IPv6 subnets and links. Thus this document does not provide a solution of router advertisements and registration for 'multi-level subnets' as indicated in 6LoWPAN [LOWPAN]. In the process, the node registration method is also useful for preventing Neighbor Discovery denial of service (DOS) attacks.

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

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

2. Definition Of Terms

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

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

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

IPv6 ND-energy-aware Rotuer(NEAR):
It is the default Router of the single hop IPv6 subnet. This router implements the optimizations specified in this document. This router should be able to handle both legacy IPv6 nodes and nodes that sends registration request.

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

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

Legacy IPv6 Router:
An IPv6 Router which implements RFC 4861 Neighbor Discovery protocols.

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

3. Assumptions for energy-aware Neighbor Discovery

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

4. The set of Requirements for Energy-awareness and optimization

In future homes, machine-to-machine networks and Data-center Virtual networks, it is essential to reduce unnecessary number of IPv6 Neighbor Discovery signalings for saving energy and saving bits in the network.

In the cloud computing environment, the concept of IPv6-subnet of link-local nodes is often extended across different networks over a Virtual LAN. Thus reducing Neighbor Discovery signaling messages is a key for enhanced services.

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

5. Basic Operations

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

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

6. Applicability Statement

This document aims to guide the implementors to choose an appropriate IPv6 neighbor discovery and Address configuration procedures suitable for any IPv6 energy-aware network. These optimization is useful for the classical IPv6 subnet and as well as future home networks, Data-Centers where saving Neighbor Discovery messages will reduce cost of control signaling and network bandwidth and as well as energy of the connected nodes. See use cases towards the end of the document.

Note that the specification allows ’Mixed-mode’ operation in the energy-aware nodes for backward compatibility and transitioning to a complete energy-aware network of hosts and routers. Though the energy-aware only nodes will minimize the ND signalling and DOS attacks in the LAN.

7. New Neighbor Discovery Options and Messages

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

7.1. Address Registration Option

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

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

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

The sender of the NS also includes the EUI-64 of the interface it is registering an address from. This is used as a unique ID for the detection of duplicate addresses. It is used to tell the difference between the same node re-registering its address and a different node (with a different EUI-64) registering an address that is already in use by someone else. The EUI-64 is also used to deliver an NA carrying an error Status code to the EUI-64 based link-local IPv6 address of the host.

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length = 2  |    Status     |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Reserved            |     Registration Lifetime     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+               EUI-64 or equivalent                            +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Type:	TBD1 (See [6LOWPAN-ND])
Length:	8-bit unsigned integer. The length of the option in units of 8 bytes. Always 2.
Status:	8-bit unsigned integer. Indicates the status of a registration in the NA response. MUST be set to 0 in NS messages. See below.
Reserved:	This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Registration Lifetime:	16-bit unsigned integer. The amount of time in a unit of 10 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option.
EUI-64:	64 bits. This field is used to uniquely identify the interface of the registered address by including the EUI-64 identifier assigned to it unmodified.

The Status values used in Neighbor Advertisements are:

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

Table 1

7.2. Refresh and De-registration

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

7.3. A New Router Advertisement Flag

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

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

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

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

This document assumes that an implementation will have configuration knobs to determine whether it is running in classical IPv6 ND [ND] or Optimized Energy Aware ND (this document) mode or both(Mixed mode).

8. Energy-aware Neighbor Discovery Messages

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

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

	routers. The time-periods for the RS varies on the deployment scenarios and the Default Router Lifetime advertised in the RAs.
Default Router Selection:	Same as in section 6.3.6 of RFC 4861[ND].
Neighbor Solicitation (NS):	Neighbor solicitation is used between the hosts and the default-router as part of NUD and registering the host’s address(es). An NS message MUST use the Address Registration option in order to accomplish the registration.
Neighbor Advertisement (NA):	As defined in [ND] and ARO option.
Redirect Messages:	A router SHOULD NOT send a Redirect message to a host since the link has non-transitive reachability. The host behavior is same as described in section 8.3 of RFC 4861[ND], i,e. a host MUST NOT send or accept redirect messages when in energy-aware mode.
	Same as in RFC 4861[ND]
MTU option:	As per the RFC 4861.
Address Resolution:	No NS/NA are sent as the prefixes are treated as off-link. Thus no address resolution is performed at the hosts. The routers keep track of Neighbor Solicitations with Address Registration options(ARO) and create an extended neighbor cache of reachable addresses. The router also knows the nexthop link-local address and corresponding link-layer address when it wants to route a packet.
Neighbor Unreachability Detection(NUD):	NUD is performed in "forward-progress" fashion as described in section 7.3.1 of RFC 4861[ND]. However, if Address Registration Option is used, the NUD SHOULD be combined with the Re-registration of the node. This way no extra message for NUD is required.

9. Energy-Aware Host Behavior

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

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

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

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

The energy-aware host MUST NOT send or accept re-direct messages. It does not join solicited node multicast address.

10. The Energy Aware Default Router (NEAR) Behavior

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

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

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

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

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

10.1. Router Configuration Modes

An energy-aware Router(NEAR) MUST be able to configure in energy-aware-only mode where it will expect all hosts register with the router following RS; thus will not support legacy hosts. However, it will create legacy NCE for NS messages for other routers in the network. This mode is able to prevent ND flooding on the link.

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

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

11. NCE Management in Energy-Aware Routers

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

Legacy:	Entries that are subject to the normal rules in [ND] that allow for garbage collection when low on memory. Legacy entries are created only when there is no duplicate NCE. In mixed-mode and energy-aware mode the legacy entries are converted to the registered entries upon successful processing of ARO. Legacy type can be considered as union of garbage-collectible and Tentative Type NCEs described in [6LOWPAN-ND].

Registered:	Entries that have an explicit registered lifetime and are kept until this lifetime expires or they are explicitly unregistered.

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

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

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

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

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

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

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

11.1. Handling ND DOS Attack

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

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

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

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

12. Mixed-Mode Operations

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

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

- The legacy IPv6 hosts and energy-aware-hosts in the network and a NEAR router
- legacy IPv6 default-router and energy-aware hosts(EAH) in the link

- one router is in mixed mode and the link contains both legacy IPv6 hosts and EAH
- A link contains both energy-aware IPv6 router and hosts and legacy IPv6 routers and hosts and each host should be able to communicate with each other.

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

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

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

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

13. Bootstrapping

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

```
      Node                                                  Router

        |                                                      |

   1.   |       ---------- Router Solicitation -------->       |

        |                    [SLLAO]                           |

   2.   |       <-------- Router Advertisement ---------       |

        |                    [PIO + SLLAO]                     |
        |                                                      |

   3.   |       ----- Address Registration (NS) -------->      |

        |                    [ARO + SLLAO]                     |

   4.   |       <-------- Neighbor Advertisement -------       |

        |                   [ARO with Status code]             |

   5. ====> Address Assignment Complete
```

Figure 1: Neighbor Discovery Address Registration and bootstrapping

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

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

14. Handling Sleepy Nodes

The solution allows the sleepy nodes to complete its sleep schedule without waking up due to periodic Router Advertisement messages or due to Multicast Neighbor Solicitation for address resolution. The node registration lifetime SHOULD be synchronized with its sleep interval period in order to avoid waking up in the middle of sleep for registration refresh. Depending on the application, the registration lifetime SHOULD be equal to or integral multiple of a node’s sleep interval period.

15. Use Case Analysis

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

15.1. Data Center Routers on the link

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

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

15.2. Edge Routers and Home Networks

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

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

15.3. M2M Networks

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

16. Mobility Considerations

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

17. Updated Neighbor Discovery Constants

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

Router Constants:

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

Host Constants:

MAX_RTR_SOLICITATION_INTERVAL(NEW)	60 seconds

18. Security Considerations

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

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

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

19. 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.

20. Changelog

Changes from 00 to 01:

- o Removed ABRO options and Multi-level subnet concept
- o Removed intermediate-router concept, behavior and definition
- o Added use-cases, Support for Mixed-mode operations and a diagram for bootstrapping scenario.
- o Added updates to ND constant values
- o A new co-author has beed added
- o Text for NCE Management and ND-DOS handling has been added
- o A new Router Advertisement flag has been added

21. Acknowledgements

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

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

22. References

22.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[6LOWPAN-ND] Shelby, Z., Chakrabarti, S., and E. Nordmark, "ND Optimizations for 6LoWPAN", draft-ietf-6lowpan-nd-17.txt (work in progress), June 2011.

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

[LOWPAN] Montenegro, G. and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 networks", RFC 4944, September 2007.

[LOWPANG] Kushalnagar, N. and G. Montenegro, "6LoWPAN: Overview, Assumptions, Problem Statement and Goals", RFC 4919, August 2007.

22.2. Informative References

[IPV6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6), Specification", RFC 2460, December 1998.

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

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

[AUTOADHOC] Baccelli, E. and M. Townsley, "IP Addressing Model in Adhoc Networks", draft-ietf-autoconf-adhoc-addr-model-02.txt (work in progress), December 2009.

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

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

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

[ULA] "Unique Local IPv6 Addresses", RFC 4193.

[IPV6M] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

Authors’ Addresses

Samita Chakrabarti
Ericsson
San Jose, CA
USA

Email: samita.chakrabarti@ericsson.com

Erik Nordmark
Cisco Systems
San Jose, CA
USA

Email: nordmark@cisco.com

Margaret Wasserman
Painless Security

Email: mrw@lilacglade.org

Network Working Group
Internet-Draft
Updates: 4862 (if approved)
Intended status: Standards Track
Expires: April 30, 2012

R. Asati
H. Singh
W. Beebee
Cisco Systems, Inc.
E. Dart
Lawrence Berkeley National Laboratory
W. George
Time Warner Cable
C. Pignatro
Cisco Systems, Inc.
October 28, 2011

Enhanced Duplicate Address Detection
draft-hsingh-6man-enhanced-dad-02.txt

Abstract

Appendix A of the IPv6 Duplicate Address Detection (DAD) document in RFC 4862 discusses Loopback Suppression and DAD. However, RFC 4862 does not settle on one specific automated means to detect loopback of Neighbor Discovery (ND of RFC 4861) messages used by DAD. Several service provider communities have expressed a need for automated detection of looped backed ND messages used by DAD. This document includes mitigation techniques and then outlines the Enhanced DAD algorithm to automate detection of looped back IPv6 ND messages used by DAD. For network loopback tests, the Enhanced DAD algorithm allows IPv6 to self-heal after a loopback is placed and removed. Further, for certain access networks the document automates resolving a specific duplicate address conflict.

Status of this Memo

Copyright Notice

Table of Contents

1. Terminology

- DAD-failed state - Duplication Address Detection failure as specified in [RFC4862]. Failure also includes if the Target Address is optimistic. Optimistic DAD is specified in [RFC4429].

- Looped back message - also referred to as a reflected message. The message sent by the sender is received by the sender due to the network or a Upper Layer Protocol on the sender looping the message back.

- Loopback - A function in which the router’s interface to the network is looped back, resulting in interface unavailability for regular data traffic forwarding. See more details in section 9.1 of [RFC1247]. Loopback function is commonly used to gain information on the quality of this interface, by employing mechanisms such as ICMPv6 pings, bit-error test etc. Loopback function may be done locally or remotely.

- NS(DAD) - shorthand notation to denote an NS with unspecified IPv6 source-address issued during DAD.

2. Introduction

Appendix A of [RFC4862] discusses Loopback Suppression and Duplicate Address Detection (DAD). However, [RFC4862] does not settle on one specific automated means to detect loopback of ND messages used by DAD. One specific DAD message is a Neighbor Solicitation (NS), specified in [RFC4861]. The NS is issued by the network interface of an IPv6 node for DAD. Another message involved in DAD is a Neighbor Advertisement (NA). The Enhanced DAD algorithm proposed in this document focuses on detecting an NS looped back to the transmitting interface during the DAD operation. Detecting a looped back NA is of no use because no problems with DAD will occur if a node receives a looped back NA. Detecting of any other looped back ND messages outside of the DAD operation is not critical and thus this document does not cover such detection. The document also includes a Mitigation section that discusses means already available to mitigate the loopback problem.

Recently service providers have reported a DAD loopback problem. Loopback testing is underway on a circuit connected to an interface on a router. The interface on the router is enabled for IPv6. The interface issues a NS for the IPv6 link-local address DAD. The NS is reflected back to the router interface due to the loopback condition of the circuit, and the router interface enters a DAD-failed state. In contrast to IPv4, IPv6 will not return to operation on the interface when the loopback condition is cleared without manual intervention. In another service provider network, two broadband modems in a home have the Ethernet ports of each modem connected to a network hub. The access concentrator serving the modems is the first-hop IPv6 router for the modems. The access concentrator also supports proxying of DAD messages. Each modem is IPv4 online. The network interface of the access concentrator serving the two broadband modems is enabled for IPv6 and the interface issues a NS(DAD) message for the IPv6 link-local address. The NS message reaches one modem first and this modem sends the message to the hub which sends the message to the second modem which forwards the message back to the access concentrator. The looped back NS message causes the network interface on the access concentrator to be in a DAD-failed state. Such a network interface typically serves over six thousand broadband modems causing all the modems (and hosts behind the modems) to fail to get IPv6 online on the access network. Additionally, it may be tedious for the access concentrator to find out which of the six thousand or more homes looped back the DAD message. Clearly there is a need for automated detection of looped back NS messages during DAD operations by a node.

3. Operational Mitigation Options

Two mitigation options are described below. The mechanisms do not require any change to existing implementations.

3.1. Disable DAD on Interface

One can disable DAD on an interface and then there is no NS(DAD) issued to be looped back. DAD is disabled by setting the interface’s DupAddrDetectTransmits variable to zero. While this mitigation may be the simplest the mitigation has three drawbacks.

It would likely require careful analysis of configuration on such point-to-point interfaces, a one-time manual configuration on each of such interfaces, and more importantly, genuine duplicates in the link will not be detected.

A network operator MAY use this mitigation.

3.2. Dynamic Disable/Enable of DAD Using Layer 2 Protocol

It is possible that one or more layer 2 protocols include provisions to detect the existence of a loopback on an interface circuit, usually by comparing protocol data sent and received. For example, PPP uses magic number (section 6.4 of [RFC1661]) to detect a loopback on an interface.

When a layer 2 protocol detects that a loopback is present on an interface circuit, the device MUST temporarily disable DAD on the interface, and when the protocol detects that a loopback is no longer present (or the interface state has changed), the device MUST (re-)enable DAD on that interface.

This solution requires no protocol changes. This solution SHOULD be enabled by default, and MUST be a configurable option.

This mitigation has several benefits. They are

1. It leverages layer 2 protocol’s built-in loopback detection capability, if available.

2. It scales better (since it relies on an event-driven), requires no additional state, timer etc. This may be a significant scaling consideration on devices with hundreds or thousands of interfaces that may be in loopback for long periods of time (such as while awaiting turn-up or during long-duration intrusive bit error rate tests).

3.3. Operational Considerations

The mitigation options discussed in the document do not require the devices on both ends of the circuit to support the mitigation functionality simultaneously, and do not propose any capability negotiation. Suffice to say that the mitigation options are well effective for the unidirectional loopback.

The mitigation options may not be effective for the bidirectional loopback (i.e. the loopback is placed in both directions of the circuit interface, so as to identify the faulty segment) if only one device followed a mitigation option specified in this document, since the other device would follow current behavior and disable IPv6 on that interface due to DAD until manual intervention restores it.

This is nothing different from what happens today (without the solutions proposed by this document) in case of unidirectional loopback. Hence, it is expected that an operator would resort to manual intervention for the devices not compliant with this document, as usual.

4. The Enhanced DAD Algorithm

The Enhanced DAD algorithm covers detection of a looped back NS(DAD) message. The document proposes use of the Nonce Option specified in the SEND document of [RFC3971]. The nonce is a random number as specified in [RFC3971]. If SEND is enabled on the router and the router also supports the new automated ND loopback detection (specified in this document), there is integration with the Enhanced DAD algorithm and SEND. See more details in the Impact on SEND section.

When the IPv6 network interface issues a NS(DAD) message, the interface includes the Nonce Option in the NS(DAD) message and saves the nonce in local store. Subsequently if the interface receives an identical NS(DAD) message, the interface logs a system management message, updates any statistics counter, and drops the looped back NS(DAD). If the DupAddrDetectTransmits variable for the interface is greater than one, subsequent NS(DAD) messages for the same Target Address should be suppressed. If the interface receives a NS(DAD) message with a different nonce but TargetAddress matches a tentative or optimistic address on the interface, the interface logs a DAD-failed system management message, updates any statistics, and behaves identical to the behavior specified in [RFC4862] for DAD failure.

Six bytes of random nonce is sufficiently large for nonce collisions. However if there is a collision because two nodes generated the same random nonce (that are using the same Target address in their NS(DAD)), then the algorithm will incorrectly detect a looped back NS(DAD) when the NS(DAD) was issued to signal a genuine duplicate. Since each looped back NS(DAD) event is logged to system management, the administrator of the network will have to intervene manually.

The algorithm is capable of detecting any ND solicitation (NS and Router Solicitation) or advertisement (NA and Router Advertisement) that is looped back. However, saving a nonce and nonce related data for all ND messages has impact on memory of the node and also adds the algorithm state to a substantially larger number of ND messages. Therefore this document does not recommend using the algorithm outside of the DAD operation by an interface on a node.

4.1. General Rules

A node MUST implement detection of looped back NS(DAD) messages during DAD for an interface address.

4.2. Processing Rules for Senders

If a node has been configured to use the Enhanced DAD algorithm, when sending a NS(DAD) for a tentative or optimistic interface address the sender MUST generate a random nonce associated with the interface address, MUST save the nonce, and MUST include the nonce in the Nonce Option included in the NS(DAD). If a looped back NS(DAD) is detected by the interface, and if the DupAddrDetectTransmits variable for the interface is greater than one, subsequent NS(DAD) messages for the same Target Address SHOULD be suppressed.

4.3. Processing Rules for Receivers

If the the node has been configured to use the Enhanced DAD algorithm and an interface on the node receives any NS(DAD) message that matches the interface address (in tentative or optimistic state), the receiver compares the nonce in the message with the saved nonce. If a match is found, the node SHOULD log a system management message, SHOULD update any statistics counter, and MUST drop the received message. If the received NS(DAD) message includes a nonce and no match is found with the saved nonce, the node SHOULD log a system management message for DAD-failed and SHOULD update any statistics counter.

4.4. Impact on SEND

The SEND document uses the Nonce Option in the context of matching an NA with an NS. However, no text in SEND has an explicit mention of detecting looped back ND messages. If this document updates [RFC4862], SEND should be updated to integrate with the Enhanced DAD algorithm. A minor update to SEND would be to explicitly mention that the nonce in SEND is also used by SEND to detect looped back NS messages during DAD operations by the node. In a mixed SEND environment with SEND and unsecured nodes, the lengths of the nonce used by SEND and unsecured nodes MUST be identical.

4.5. Changes to RFC 4862

The following text is added to [RFC4862] at a yet to be determined location in [RFC4862].

A router that supports IPv6 DAD MUST implement the detection of looped back NS messages during DAD operation as specified in this document. A network interface on any other IPv6 node that is not a router SHOULD implement the detection of looped back NS messages during DAD operation as specified in this document.

4.6. Actions to Perform on Detecting a Genuine Duplicate

As described in paragraphs above the nonce can also serve to detect genuine duplicates even when the network has potential for looping back ND messages. When a genuine duplicate is detected, the node follows the manual intervention specified in section 5.4.5 of [RFC4862]. However, in certain networks such as an access network if the genuine duplicate matches the tentative or optimistic IPv6 address of a network interface of the access concentrator, automated actions are proposed.

One access network is a cable broadband deployment where the access concentrator is the first-hop IPv6 router to several thousand broadband modems. The router also supports proxying of DAD messages. The network interface on the access concentrator initiates DAD for an IPv6 address and detects a genuine duplicate due to receiving an NS(DAD) or an NA message. On detecting such a duplicate the access concentrator logs a system management message, drops the received ND message, and blocks the modem on whose layer 2 service identifier the NS(DAD) or NA message was received on.

The network described above follows a trust model where a trusted router serves un-trusted IPv6 host nodes. Operators of such networks have a desire to take automated action if a network interface of the trusted router has a tentative or optimistic address duplicate with a host served by trusted router interface. Any other network that follows the same trust model MAY use the automated actions proposed in this section.

5. Security Considerations

The nonce can be exploited by a rogue deliberately changing the nonce to fail the looped back detection specified by the Enhanced DAD algorithm. SEND is recommended for this exploit. For any mitigation suggested in the document such as disabling DAD has an obvious security issue before a remote node on the link can issue reflected NS(DAD) messages. Again, SEND is recommended for this exploit.

6. IANA Considerations

None.

7. Acknowledgements

Thanks to Eric Levy-Abegnoli, Erik Nordmark, and Fred Templin and Tassos Chatzithomaoglou for their guidance and review of the document. Thanks to Thomas Narten for encouraging this work. Thanks to Steinar Haug and Scott Beuker for describing the use cases.

8. Normative References

[RFC1247] Moy, J., "OSPF Version 2", RFC 1247, July 1991.

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 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.

Authors’ Addresses

Rajiv Asati
Cisco Systems, Inc.
7025 Kit Creek road
Research Triangle Park, NC 27709-4987
USA

Email: rajiva@cisco.com
URI: http://www.cisco.com/

Hemant Singh
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

Phone: +1 978 936 1622
Email: shemant@cisco.com
URI: http://www.cisco.com/

Wes Beebee
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

Phone: +1 978 936 2030
Email: wbeebee@cisco.com
URI: http://www.cisco.com/

Eli Dart
Lawrence Berkeley National Laboratory
ESnet Network Engineering Group
USA

Email: dart@es.net

Wes George
Time Warner Cable
13820 Sunrise Valley Drive
Herndon, VA 20171
USA

Email: wesley.george@twcable.com

Carlos Pignatro
Cisco Systems, Inc.
7025 Kit Creek Road
Research Triangle Park, NC 27709-4987
USA

Email: cpignataro@cisco.com
URI: http://www.cisco.com/

Draft Author:
Tim Chown<tjc@ecs.soton.ac.uk>

6man Working Group   A. Matsumoto
Internet-Draft   T. Fujisaki
Intended status: Standards Track   J. Kato
Expires: December 30, 2011   NTT
                                                               T. Chown
                                              University of Southampton
                                                          June 28, 2011

Distributing Address Selection Policy using DHCPv6
draft-ietf-6man-addr-select-opt-01.txt

Abstract

RFC 3484 defines default address selection mechanisms for IPv6 that allow nodes to select appropriate address when faced with multiple source and/or destination addresses to choose between. The RFC allowed for the future definition of methods to administratively configure the address selection policy information. This document defines a new DHCPv6 option for such configuration, allowing a site administrator to distribute address selection policy, and thus control the address selection behavior of nodes in their site. While RFC 3484 is in the process of being updated, with a revised default policy table, that table may not suit every scenario, and thus the DHCPv6 option defined in this text may be used to override that policy where desired.

Status of this Memo

This Internet-Draft will expire on December 30, 2011.

Copyright Notice

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

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

1. Introduction

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

Some problems have been identified with the default address selection policy detailed in RFC 3484 [RFC5220], and as a result the RFC is in the process of being updated, as per [I-D.ietf-6man-rfc3484-revise]. While this update provides a better default address selection policy, it is unlikely that such a default will suit all scenarios, and thus mechanisms to control the source address selection policy will be necessary. Requirements for those mechanisms are described in [RFC5221], while solutions are discussed in [I-D.ietf-6man-addr-select-sol] and [I-D.ietf-6man-addr-select-considerations]. Those documents have helped shape the improvements in [I-D.ietf-6man-rfc3484-revise] as well as the DHCPv6 option defined here.

1.1. Conventions Used in This Document

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

1.2. Terminology

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

2. Address Selection Policy Option

The Address Selection Policy Option provides the policy table for address selection rules as described in RFC 3484 and updated in [I-D.ietf-6man-rfc3484-revise].

Each end node is expected to configure its policy table, as described in RFC 3484, using the Address Selection Policy option information as described in the section below on processing the option.

The format of the Address Selection Policy option is given below:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          OPTION_DASP          |         option-len            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |    label      |  precedence   |z|  reserved   |   prefix-len  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               zone-index (if present (z = 1))                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      |                   Prefix   (Variable Length)                  |
      |                                                               |
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |    label      |  precedence   |z|  reserved   |   prefix-len  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               zone-index (if present (z = 1))                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      |                   Prefix   (Variable Length)                  |
      |                                                               |
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                                                               .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |    label      |  precedence   |z|  reserved   |   prefix-len  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               zone-index (if present (z = 1))                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      |                   Prefix   (Variable Length)                  |
      |                                                               |
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Fig. 1]

Fields:

option-code: OPTION_DASP (TBD)

option-len: The total length of the label fields, precedence fields, zone-index fields, prefix-len fields, and prefix fields in octets.

label: An 8-bit unsigned integer; this value is used to make a combination of source address prefixes and destination address prefixes.

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

z bit: 'zone-index' bit. If z bit is set to 1, 32 bit zone-index value is included right after the "prefix-len" field, and "Prefix" value continues after the "zone-index" field. If z bit is 0, "Prefix" value continues right after the "prefix-len" value.

reserved: 6-bit reserved field. Initialized to zero by sender, and ignored by receiver.

zone-index: If the z-bit is set to 1, this field is inserted between "prefix-len" field and "Prefix" field. The zone-index field is an 32-bit unsigned integer and used to specify zones for scoped addresses. This bit length is defined in RFC3493 [RFC3493] as 'scope ID'.

prefix-len: An 8-bit unsigned integer; the number of leading bits in the prefix that are valid. The value ranges from 0 to 128. The Prefix field is 0, 4, 8, 12, or 16 octets, depending on the length.

Prefix: A variable-length field containing an IP address or the prefix of an IP address. An IPv4-mapped address [RFC4291] must be used to represent an IPv4 address as a prefix value.

3. Appearance of this Option

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

4. Processing the Address Selection Policy Option

This section describes how to process received Address Selection Policy Options at the DHCPv6 client.

This option’s concept is to serve as a hint for a node about how to behave in the network. So, basically, it should be up to the node’s administrator how to make use of or even ignore the received policy information.

However, we need to define the default behavior of the receiving node in order to reduce operational complexity.

4.1. Handling the local policy table

RFC3484 defines the default policy for the policy table. Also, a user is usually able to configure the policy table to satisfy his requirement.

The client node SHOULD provide the following choices:

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

4.2. Processing multiple received policy tables

The policy table is node-global information by its nature. So, the node cannot use multiple received policy tables at the same time.

It should be noted that adopting a received policy table as the node-global information can cause security problems, such as DOS attack, and leak of privacy information.

Moreover, it also should be noted that, when a node is single-homed and has only one upstream line, adopting a received policy table does not degrade the security level.

Under the above assumptions, we specify how to handle multiple received policy tables below.

A node MAY use OPTION_DASP in any of the following two cases:

1: The address selection option is delivered across a secure, trusted channel.
2: The address selection option is not secured, but the node is single-homed.

In other cases the node MUST NOT use OPTION_DASP unless the node is specifically configured to do so.

5. Implementation Considerations

- The value 'label' is passed as an unsigned integer, but there is no special meaning for the value, that is whether it is a large or small number. It is used to select a preferred source address prefix corresponding to a destination address prefix by matching the same label value within the DHCP message. DHCPv6 clients need to convert this label to a representation specified by each implementation (e.g., string).

- Currently, the label and precedence values are defined as 8-bit unsigned integers. In almost all cases, this value will be enough.

- The maximum number of address selection rules that may be conveyed in one DHCPv6 message depends on the prefix length of each rule and the maximum DHCPv6 message size defined in RFC 3315. It is possible to carry over 3,000 rules in one DHCPv6 message (maximum UDP message size), but the usual number would be much smaller, e.g. the default policy table defined in RFC 3484 contains 5 rules.

- Since the number of selection rules could be large, an administrator configuring the policy to be distributed should consider the resulting DHCPv6 message size.

6. Security Considerations

A rogue DHCPv6 server could issue bogus address selection policies to a client. This might lead to incorrect address selection by the client, and the affected packets might be blocked at an outgoing ISP because of ingress filtering. Alternatively, an IPv6 transition mechanism might be preferred over native IPv6, even if it is available.

To guard against such attacks, both DCHP clients and servers SHOULD use DHCP authentication, as described in section 21 of RFC 3315, "Authentication of DHCP messages."

7. IANA Considerations

IANA is requested to assign option codes to OPTION_DASP from the option-code space as defined in section "DHCPv6 Options" of RFC 3315.

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.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

8.2. Informative References

[I-D.ietf-6man-addr-select-considerations] Chown, T., "Considerations for IPv6 Address Selection Policy Changes", draft-ietf-6man-addr-select-considerations-03 (work in progress), March 2011.

[I-D.ietf-6man-addr-select-sol] Matsumoto, A., Fujisaki, T., and R. Hiromi, "Solution approaches for address-selection problems", draft-ietf-6man-addr-select-sol-03 (work in progress), March 2010.

[I-D.ietf-6man-rfc3484-revise] Matsumoto, A., Kato, J., and T. Fujisaki, "Update to RFC 3484 Default Address Selection for IPv6", draft-ietf-6man-rfc3484-revise-03 (work in progress), June 2011.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

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

[RFC5220] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Problem Statement for Default Address Selection in Multi-Prefix Environments: Operational Issues of RFC 3484 Default Rules", RFC 5220, July 2008.

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

Appendix A. Past Discussion

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

- Since we got a comment that some implementations use 32-bit integers for zone index value, we extended the bit length of the ’zone index’ field. However, as described above, there might be few cases to specify ’zone index’ in policy distribution, we defined this field as optional, controlled by a flag.

- There may be some demands to control the use of special address types such as the temporary addresses described in RFC4941 [RFC4941], address assigned by DHCPv6 and so on. (e.g., informing not to use a temporary address when it communicate within the an organization’s network). It is possible to indicate the type of addresses using reserved field value.

Authors' Addresses

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

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

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

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

Jun-ya Kato
NTT SI Lab
3-9-11 Midori-Cho
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 2939
Email: kato@syce.net

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

Email: tjc@ecs.soton.ac.uk

Network Working Group M. Eubanks
Internet-Draft AmericaFree.TV LLC
Intended status: Standards Track P. Chimento
Expires: May 3, 2012 Johns Hopkins University Applied
Physics Laboratory
October 31, 2011

UDP Checksums for Tunneled Packets
draft-ietf-6man-udpchecksums-01

Abstract

This document provides an update of RFC 2460[RFC2460] in order to improve the performance of IPv6 in an increasingly important use case, the use of tunneling to carry new transport protocols. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for suitable tunneling protocol where header information is protected on the "inner" packet being carried. This relaxation removes the overhead associated with the computation of UDP checksums on tunneled IPv6 packets and thereby improves the efficiency of the traversal of firewalls and other network middleware by such new protocols. We describe how the IPv6 UDP checksum requirement can be relaxed in the situation where the encapsulated packet itself contains a checksum, the limitations and risks of this approach, and provides restrictions on the use of this relaxation to mitigate these risks.

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 will expire on May 3, 2012.

Copyright Notice

Table of Contents

1. Introduction

This work constitutes the first upgrade of RFC 2460[RFC2460], in order to improve the performance of IPv6 with transport layer protocols carried encapsulated in tunnels. With the rapid growth of the Internet, tunneling protocols have become increasingly important to enable the deployment of new transport layer protocols. Tunneled protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, switches and end hosts on the global Internet for a new transport protocol is now measured in decades. At the same time, the increasing use of firewalls and other security related middleware means that truly new tunnel protocols, with new protocol numbers, are also unlikely to be deployable in a reasonable time frame, which has resulted in an increasing interest in and use of UDP-based tunneling protocols. In such protocols, there is an encapsulated "inner" packet, and the "outer" packet carrying the tunneled inner packet is a UDP packet, which can pass through firewalls and other middleware filtering that is a fact of life on the current Internet.

As tunnel endpoints may be routers or middleware aggregating traffic from large numbers of tunnel users, the computation of an additional checksum on the outer UDP packet, when protected, is seen to be an unwarranted burden on the nodes implementing lightweight tunneling protocols, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum on the IP packet itself, and the checksum on the outer UDP packet can be set to zero. However in IPv6 there is not a checksum on the IP packet and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a 0 checksum. So, while sending a UDP packet with a 0 checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. In order to meet the needs of the deployers of IPv6 UDP tunnels, this document modifies RFC 2460 to allow for the ignoring of UDP checksums under constrained situations (IPv6 tunneling where the inner packet exists and has a checksum), based on the considerations set forth in [I-D.ietf-6man-udpzero].

While the origin of this I-D is the problem raised by the draft titled "Automatic IP Multicast Without Explicit Tunnels", also known as "AMT," [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability, immediately to LISP [I-D.ietf-lisp], and also to other tunneling protocols to come out of Softwires and other IETF Working Groups.

Since the first version of this document, the need for an efficient, lightweight UDP tunneling mechanism has increased. Indeed, other workgroups, notably LISP [I-D.ietf-lisp] and Softwires [RFC5619] have also expressed a need to have exceptions to the RFC 2460 prohibition.

Other users of UDP as a tunneling protocol, for example, L2TP and Softwires may benefit from a relaxation of the RFC 2460 restriction.

The third version of this document benefited from a close read by Magnus Westerlund and Gorry Fairhurst.

2. Some Terminology

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

Although we will try to avoid them when possible, we may use the terms "tunneling" and "tunneled" as adjectives when describing packets. When we refer to 'tunneling packets' we refer to the outer packet header that provides the tunneling function. When we refer to 'tunneled packets' we refer to the inner packet, i.e. the packet being carried in the tunnel.

3. Problem Statement

The argument is that since in the case of AMT multicast packets already have a UDP header with a checksum, there is no additional benefit and indeed some cost to nodes to both compute and check the UDP checksum of the outer (encapsulating) header. Consequently, IPv6 should make an exception to the rule that the UDP checksum MUST not be 0, and allow tunneling protocols to set the checksum field of the outer header only to 0 and skip both the sender and receiver computation.

4. Discussion

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

In Section 5.1 of [I-D.ietf-6man-udpzero], the authors propose nine (9) constraints on the usage of a zero checksum for UDP over IPv6. We agree with the restrictions proposed, and in fact proposed some of those restrictions ourselves in the previous version of the current draft. These restrictions are incorporated into the proposed changes below.

As has been pointed out in [I-D.ietf-6man-udpzero] and in many mailing lists, there is still the possibility of deep-inspection firewall devices or other middleboxes actually checking the UDP checksum field of the outer packet and discarding the tunneling packets. This is would be an issue also for legacy systems which have not implemented the change in the IPv6 specification. So in any case, there may be packet loss of lightweight tunneling packets because of mixed new-rule and old-rule nodes.

As an example, we discuss how can errors be detected and handled in a lightweight UDP tunneling protocol when the checksum protection is disabled. Note that other (non-tunneling) protocols may have different approaches. We suggest that the following could be an approach to this problem:

- Context (i.e. tunneling state) should be established via application PDUs that are carried in checksummed UDP packets. That is, any control packets flowing between the tunnel endpoints should be protected by UDP checksums. The control packets can also contain any negotiation that is necessary to set up the endpoint/adapters to accept UDP packets with a zero checksum.

- Only UDP packets containing tunneled packets should have a UDP checksum equal to zero.

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

- Corruption of the encapsulating IPv6 source address, destination address and/or the UDP source port, destination port fields : If the 9 restrictions in [I-D.ietf-6man-udpzero] are followed, the inner packets (tunneled packets) should be protected and run the usual (presumably small) risk of having undetected corruption(s). If lightweight tunneling protocol contexts contain (at a minimum) source and destination IP addresses and source and destination ports, there are 16 possible corruption outcomes. We note that these outcomes not equally likely, as most require multiple bit errors with errored bits in separate fields. The possible corruption outcomes fall out this way:

    * Half of the 16 possible corruption combinations have a corrupted destination address. If the incorrect destination is

reached and the node doesn’t have an application for the destination port, the packet will be dropped. If the application at the incorrect destination is the same lightweight tunneling protocol and if it has a matching context (which can be assumed to be a very low probability event) the inner packet will be decapsulated and forwarded. If it is some other application, with very high probability, the application will not recognize the contents of the packet.

  * Half of the 8 possible corruption combinations with a correct destination address have a corrupted source address. If the tunnel contexts contain all elements of the address-port 4-tuple, then the likelihood is that this corruption will be detected.

  * Of the remaining 4 possibilities, with valid source and destination IPv6 addresses, 1 has all 4 fields valid, the other three have one or both ports corrupted. Again, if the tunneling endpoint context contains sufficient information, these error should be detected with high probability.

- Corruption of source-fragmented encapsulating packets: In this case, a tunneling protocol may reassemble fragments associated with the wrong context at the right tunnel endpoint, or it may reassemble fragments associated with a context at the wrong tunnel endpoint, or corrupted fragments may be reassembled at the right context at the right tunnel endpoint. In each of these cases, the IPv6 length of the encapsulating header may be checked (though [I-D.ietf-6man-udpzero] points out the weakness in this check). In addition, if the encapsulated packet is protected by a transport (or other) checksum, these errors can be detected (with some probability).

While this is not a perfect solution, it can reduce the risks of relaxing the UDP checksum requirement for IPv6.

5. The Zero-Checksum Solution

The solution to the overhead associated with UDP packets carrying encapsulated tunnel traffic is to allow a UDP checksum of zero on the outer encapsulating packet of a lightweight tunneling protocol. UDP endpoints that implement this solution MUST change their behavior and not discard UDP packets received with a 0 checksum on the outer packet of tunneling protocols. If this is done constraints in Section 5.1 of [I-D.ietf-6man-udpzero] also MUST be adopted.

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

"Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error."

This item should be taken out of the bullet list and should be modified as follows:

> Whenever originating a UDP packet, an IPv6 node SHOULD compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers SHOULD discard UDP packets containing a zero checksum, and SHOULD log the error. However, some protocols, such as lightweight tunneling protocols that use UDP as a tunnel encapsulation, MAY omit computing the UDP checksum of the encapsulating UDP header and set it to zero, subject to the constraints described in [I-D.ietf-6man-udpzero]. In cases where the encapsulating protocol uses a zero checksum for UDP, the receiver of packets sent to a port enabled to receive zero-checksum packets MUST NOT discard packets solely for having a UDP checksum of zero. Note that these constraints apply only to encapsulating protocols that omit calculating the UDP checksum and set it to zero. An encapsulating protocol can always choose to compute the UDP checksum, in which case, its behavior should be as specified originally.

1. IPv6 protocol stack implementations SHOULD NOT by default allow the new method. The default node receiver behavior MUST discard all IPv6 packets carrying UDP packets with a zero checksum.

2. Implementations MUST provide a way to signal the set of ports that will be enabled to receive UDP datagrams with a zero checksum. An IPv6 node that enables reception of UDP packets with a zero-checksum, MUST enable this only for a specific port or port-range. This may be implemented via a socket API call, or similar mechanism.

3. RFC 2460 specifies that IPv6 nodes should log UDP datagrams with a zero-checksum. A port for which zero-checksum has been

enabled MUST NOT log zero-checksum datagrams for that reason (of course, there might be other reasons to log such packets).

4. A stack may separately identify UDP datagrams that are discarded with a zero checksum. It SHOULD NOT add these to the standard log, since the endpoint has not been verified.

5. UDP Tunnels that encapsulate IP may rely on the inner packet integrity checks provided that the tunnel will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel MUST provide an additional integrity verification mechanism. An integrity mechanism is always recommended at the tunnel layer to ensure that corruption rates of the inner most packet are not increased.

6. Tunnels that encapsulate Non-IP packets MUST have a CRC or other mechanism for checking packet integrity, unless the Non-IP packet specifically is designed for transmission over lower layers that do not provide any packet integrity guarantee. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

7. UDP applications that support use of a zero-checksum, SHOULD NOT rely upon correct reception of the IP and UDP protocol information (including the length of the packet) when decoding and processing the packet payload. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

8. If a method proposes recursive tunnels, it MUST provide guidance that is appropriate for all use-cases. Restrictions may be needed to the use of a tunnel encapsulations and the use of recursive tunnels (e.g. Necessary when the endpoint is not verified).

9. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum MUST provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and port numbers in the ICMPv6 message body).

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

6. Additional Observations

The persistence of this issue among a significant number of protocols being developed in the IETF requires a definitive policy. The authors would like to make the following observations:

- An empirically-based analysis of the probabilities of packet corruptions (with or without checksums) has not (to our knowledge) been conducted since about 2000. It is now 2011. We strongly suggest that an empirical study is in order, along with an extensive analysis of IPv6 header corruption probabilities.
- A key cause of this issue generally is the lack of protocol support in middleboxes. Specifically, new protocols, such as LISP, are being forced to use UDP tunnels just to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If this were not the case, the use of UDP-lite might become more viable for some (but not necessarily all) lightweight tunneling protocols.
- Another cause of this issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as it the TCP checksum for TCP flows). Protocols that do not use a pseudo-header approach to computing a checksum or CRC have essentially no protection from misdelivered packets.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

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

9. Acknowledgements

We would like to thank Brian Haberman, Magnus Westerlund and Gorry Fairhurst for discussions and reviews.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

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

10.2. Informative References

[I-D.ietf-6man-udpzero]
Fairhurst, G. and M. Westerlund, "IPv6 UDP Checksum Considerations", draft-ietf-6man-udpzero-04 (work in progress), October 2011.

[I-D.ietf-lisp]
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-15 (work in progress), July 2011.

[I-D.ietf-mboned-auto-multicast]

Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., and T. Pusateri, "Automatic IP Multicast Without Explicit Tunnels (AMT)", draft-ietf-mboned-auto-multicast-11 (work in progress), July 2011.

Authors' Addresses

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

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

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

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

Internet Engineering Task Force G. Fairhurst
Internet-Draft University of Aberdeen
Intended status: Informational M. Westerlund
Expires: April 27, 2012 Ericsson
October 25, 2011

IPv6 UDP Checksum Considerations
draft-ietf-6man-udpzero-04

Abstract

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

Status of this Memo

This Internet-Draft will expire on April 27, 2012.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

1.1. Document Structure

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

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

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

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

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

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

1.2. Background

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

1.2.1. The Role of a Transport Endpoint

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

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

1.2.2. The UDP Checksum

UDP, as defined in [RFC0768], supports two checksum behaviours when used with IPv4. The normal behaviour is for the sender to calculate a checksum over a block of data that includes a pseudo header and the

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

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

- Endpoint IP source address (always included in the pseudo header of the checksum)

- Endpoint IP destination address (always included in the pseudo header of the checksum)

- Upper layer payload type (always included in the pseudo header of the checksum)

- IP length of payload (always included in the pseudo header of the checksum)

- Length of the network layer extension headers (i.e. by correct position of the checksum bytes)

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

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

- Transport payload size (always included in the checksum)

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

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

1.2.3. Differences between IPv6 and IPv4

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

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

1.3. Use of UDP Tunnels

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

1.3.1. Motivation for new approaches

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

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

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

1.3.2. Reducing forwarding cost

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

Automatic IP Multicast Without Explicit Tunnels, known as AMT [I-D.ietf-mboned-auto-multicast] currently specifies UDP as the transport protocol for packets carrying tunneled IP multicast packets. The current specification for AMT requires that the UDP checksum in the outer packet header should be 0 (see Section 6.6 of [I-D.ietf-mboned-auto-multicast]). It argues that the computation of an additional checksum, when an inner packet is already adequately protected, is an unwarranted burden on nodes implementing lightweight tunneling protocols. The AMT protocol needs to replicate a multicast packet to each gateway tunnel. In this case, the outer IP addresses are different for each tunnel and therefore require a different pseudo header to be built for each UDP replicated encapsulation.

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

1.3.3. Need to inspect the entire packet

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

1.3.4. Interactions with middleboxes

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

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

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

1.3.5. Support for load balancing

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

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

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

2. Standards-Track Transports

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

2.1. UDP with Standard Checksum

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

[RFC5405].

2.2. UDP-Lite

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

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

2.2.1. Using UDP-Lite as a Tunnel Encapsulation

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

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

2.3. General Tunnel Encapsulations

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

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

3. Issues Requiring Consideration

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

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

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

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

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

3.1. Effect of packet modification in the network

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

Corruption in the network may result in:

- A datagram being mis-delivered to the wrong host/router or the wrong transport entity within an endpoint. Such a datagram needs to be discarded;

- A datagram payload being corrupted, but still delivered to the intended host/router transport entity. Such a datagram needs to be either discarded or correctly processed by an application that provides its own integrity checks;

- A datagram payload being truncated by corruption of the length field. Such a datagram needs to be discarded.

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

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

3.1.1. Corruption of the destination IP address

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

There are two possible outcomes:

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

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

3.1.2. Corruption of the source IP address

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

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

- An application that requires a per-established context may disregard the datagram as invalid, or could map this to another context (if a context for the modified source address was already activated).

- A stateless application will process the datagram outside of any context, a simple example is the ECHO server, which will respond with a datagram directed to the modified source address. This would create unwanted additional processing load, and generate traffic to the modified endpoint address.

- Some datagram applications build state using the information from packet headers. A previously unused source address would result in receiver processing and the creation of unnecessary transport-layer state at the receiver. For example, Real Time Protocol (RTP) [RFC3550] sessions commonly employ a source independent receiver port. State is created for each received flow. Reception of a datagram with a corrupted source address will therefore result in accumulation of unnecessary state in the RTP state machine, including collision detection and response (since the same synchronization source, SSRC, value will appear to arrive from multiple source IP addresses).

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

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

3.1.3. Corruption of Port Information

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

3.1.4. Delivery to an unexpected port

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

- Delivery to a port that is not in use. The packet is discarded, but could generate an ICMPv6 message (e.g. port unreachable).

- It could be delivered to a different node that implements the same application, where the packet may be accepted, generating side-effects or accumulated state.

- It could be delivered to an application that does not implement the tunnel protocol, where the packet may be incorrectly parsed, and may be misinterpreted, generating side-effects or accumulated state.

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

1. There is no handshake prior to communication and no sequence numbers (as in TCP, DCCP, or SCTP). Together, this makes it hard to verify that an application is given only the data associated with a transport session.

2. Applications writers often bind to wild-card values in endpoint identifiers and do not always validate correctness of datagrams they receive (guidance on this topic is provided in [RFC5405]).

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

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

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

3.1.5. Corruption of Fragmentation Information

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

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

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

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

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

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

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

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

3.2. Validating the network path

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

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

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

3.3. Applicability of method

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

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

The IETF should carefully consider constraints on sanctioning the use of any new transport mode. If this is specified and widely available, it may be expected to be used by applications that are perceived to gain benefit. Any solution that uses an end-to-end transport protocol, rather than an IP-in-IP encapsulation, needs to minimise the possibility that end-hosts could confuse a corrupted or wrongly delivered packet with that of data addressed to an application running on their endpoint unless they accept that behavior.

3.4. Impact on non-supporting devices or applications

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

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

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

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

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

4.1. Alternatives to the Standard Checksum

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

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

- UDP-Lite with the checksum coverage set to only the header portion of a packet. This requires a pseudo header checksum calculation only on the encapsulating packet header. The computed checksum value may be cached (before adding the Length field) for each flow/destination and subsequently combined with the Length of each packet to minimise per-packet processing. This value is combined with the UDP payload length for the pseudo header, however this length is expected to be known when performing packet forwarding.

- The proposed UDP Tunnel Transport, UDPTT [UDPTT] suggested a method where UDP would be modified to derive the checksum only from the encapsulating packet protocol header. This value does not change between packets in a single flow. The value may be cached per flow/destination to minimise per-packet processing.

- There has been a proposal to simply ignore the UDP checksum value on reception at the tunnel egress, allowing a tunnel ingress to insert any value correct or false. For tunnel usage, a non standard checksum value may be used, forcing an RFC 2460 receiver to drop the packet. The main downside is that it would be impossible to identify a UDP packet (in the network or an endpoint) that is treated in this way compared to a packet that has actually been corrupted.

- A method has been proposed that uses a new (to be defined) IPv6 Destination Options Header to provide an end-to-end validation check at the network layer. This would allow an endpoint to verify delivery to an appropriate end point, but would also require IPv6 nodes to correctly handle the additional header, and would require changes to middlebox behavior (e.g. when used with a NAT that always adjusts the checksum value).

- UDP modified to disable checksum processing [I-D.ietf-6man-udpchecksums]. This requires no checksum calculation, but would require constraints on appropriate usage and updates to end-points and middleboxes.

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

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

4.2. Comparison

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

4.2.1. Middlebox Traversal

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

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

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

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

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

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

4.2.2. Load Balancing

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

4.2.3. Ingress and Egress Performance Implications

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

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

4.2.4. Deployability

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

- The solution with the best deployability is regular UDP. This requires no changes and has good middlebox traversal characteristics.

- The next easiest to deploy is the delta checksum solution. This does not modify the protocol on the wire and only needs changes in tunnel ingress.

- IP-in-IP tunnels should not require changes to the end-points, but raise issues when traversing firewalls and other security-type devices, which are expected to require updates.

- Ignoring the checksum on reception will require changes at both end-points. The never ceasing risk of path failure requires additional checks to ensure this solution is robust and will require changes or additions to the tunneling control protocol to negotiate support and validate the path.

- The remaining solutions offer similar deployability. UDP-Lite requires support at both end-points and in middleboxes. UDPTT and Zero-checksum with or without an Extension header require support at both end-points and in middleboxes. UDP-Lite, UDPTT, and Zero-checksum and Extension header may additionally require changes or additions to the tunneling control protocol to negotiate support and path validation.

4.2.5. Corruption Detection Strength

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

4.2.6. Comparison Summary

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

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

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

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

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

The remaining proposals may be described in similar terms:

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

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

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

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

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

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

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

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

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

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

5. Requirements on the specification of transported protocols

5.1. Constraints required on usage of a zero checksum

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

1. IPv6 protocol stack implementations should not by default allow the new method. The default node receiver behaviour must discard all IPv6 packets carrying UDP packets with a zero checksum.

2. Implementations must provide a way to signal the set of ports that will be enabled to receive UDP datagrams with a zero checksum. An IPv6 node that enables reception of UDP packets with a zero-checksum, must enable this only for a specific port or port-range. This may be implemented via a socket API call, or similar mechanism.

3. RFC 2460 specifies that IPv6 nodes should log UDP datagrams with a zero-checksum. This should remain the case for any datagram received on a port that does not explicitly enable zero-checksum processing. A port for which zero-checksum has been enabled must not log the datagram.

4. A stack may separately identify UDP datagrams that are discarded with a zero checksum. It should not add these to the standard log, since the endpoint has not been verified.

5. Tunnels that encapsulate IP may rely on the inner packet integrity checks provided that the tunnel will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel must provide an additional integrity verification mechanism. An integrity mechanisms is always recommended at the tunnel layer to ensure that corruption rates of the inner most packet are not increased.

6. Tunnels that encapsulate Non-IP packets must have a CRC or other mechanism for checking packet integrity, unless the Non-IP packet specifically is designed for transmission over lower layers that do not provide any packet integrity guarantee. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

7. UDP applications that support use of a zero-checksum, should not rely upon correct reception of the IP and UDP protocol information (including the length of the packet) when decoding and processing the packet payload. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

8. If a method proposes recursive tunnels, it needs to provide guidance that is appropriate for all use-cases. Restrictions may

be needed to the use of a tunnel encapsulations and the use of recursive tunnels (e.g. Necessary when the endpoint is not verified).

9. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum must provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and port numbers in the ICMPv6 message body).

Deployment of the new method needs to remain restricted to endpoints that explicitly enable this mode and adopt the above procedures. Any middlebox that examines or interact with the UDP header over IPv6 should support the new method.

6. Summary

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

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

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

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

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

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

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

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

7. Acknowledgements

Brian Haberman, Brian Carpenter, Magaret Wasserman, Lars Eggert,

others in the TSV directorate.

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

8. IANA Considerations

This document does not require any actions by IANA.

9. Security Considerations

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

10. References

10.1. Normative References

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

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

10.2. Informative References

[ECMP] "Using the IPv6 flow label for equal cost multipath routing in tunnels (draft-carpenter-flow-ecmp)".

[I-D.ietf-6man-udpchecksums]
Eubanks, M., "UDP Checksums for Tunneled Packets", draft-ietf-6man-udpchecksums-00 (work in progress), March 2011.

[I-D.ietf-intarea-tunnels]

Touch, J. and M. Townsley, "Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-00 (work in progress), March 2010.

[I-D.ietf-mboned-auto-multicast]
Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., Pusateri, T., and T. Morin, "Automatic IP Multicast Tunneling", draft-ietf-mboned-auto-multicast-11 (work in progress), July 2011.

[LISP] D. Farinacci et al, "Locator/ID Separation Protocol (LISP)", March 2009.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC1141] Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, January 1990.

[RFC1624] Rijsinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC5415] Calhoun, P., Montemurro, M., and D. Stanley, "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol

Specification", RFC 5415, March 2009.

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

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

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

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

Appendix A. Document Change History

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

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

Individual Draft 01

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

Individual Draft 02

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

Working Group Draft 00

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

Working Group Draft 01

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

Working Group Draft 02

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

Working Group Draft 03

- Editorial updates

Working Group Draft 04

- Resubmission only updating the AMT and RFC2765 references.

Authors' Addresses

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

Phone:
Email: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk/users/gorry

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

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

IPv6 Maintenance Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 12, 2012

K. Lynn, Ed.
Consultant
J. Martocci
Johnson Controls
C. Neilson
Delta Controls
S. Donaldson
Honeywell

October 10, 2011

Transmission of IPv6 over MS/TP Networks
draft-lynn-6man-6lobac-02

Abstract

MS/TP (Master-Slave/Token-Passing) is a contention-free access method for the TIA-485-A physical layer that is used extensively in building automation networks. This document describes the frame format for transmission of IPv6 packets and the method of forming link-local and statelessly autoconfigured IPv6 addresses on MS/TP networks.

Status of this Memo

This Internet-Draft will expire on April 12, 2012.

Copyright Notice

Table of Contents

1. Introduction

MS/TP (Master-Slave/Token-Passing) is a contention-free access method for the [TIA-485-A] physical layer that is used extensively in building automation networks. This document describes the frame format for transmission of IPv6 [RFC2460] packets and the method of forming link-local and statelessly autoconfigured IPv6 addresses on MS/TP networks. The general approach is to adapt elements of the 6LoWPAN [RFC4944] specification to constrained wired networks.

An MS/TP device is typically based on a low-cost microcontroller with limited processing power and memory. Together with low data rates and a small address space, these constraints are similar to those faced in 6LoWPAN networks and suggest some elements of that solution might be applied. MS/TP differs significantly from 6LoWPAN in at least three respects: a) MS/TP devices typically have a continuous source of power, b) all MS/TP devices on a segment can communicate directly so there are no hidden node or mesh routing issues, and c) proposed changes to MS/TP will support payloads of up to 1500 octets, eliminating the need for link-layer fragmentation and reassembly.

The following sections provide a brief overview of MS/TP, then describe how to form IPv6 addresses and encapsulate IPv6 packets in MS/TP frames. This document also specifies a header compression mechanism, based on [RFC6282], that is recommended in order to make IPv6 practical on low speed MS/TP networks.

1.1. Requirements Language

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

1.2. Abbreviations Used

ASHRAE:	American Society of Heating, Refrigerating, and Air-Conditioning Engineers (http://www.ashrae.org)
BACnet:	An ISO/ANSI/ASHRAE Standard Data Communication Protocol for Building Automation and Control Networks
CRC:	Cyclic Redundancy Check
MAC:	Medium Access Control
MSDU:	MAC Service Data Unit (MAC client data)
UART:	Universal Asynchronous Transmitter/Receiver

1.3. MS/TP Overview

This section provides a brief overview of MS/TP, which is specified in Clause 9 of ANSI/ASHRAE 135-2010 [BACnet] and included herein by reference. [BACnet] also covers physical layer deployment options.

MS/TP is designed to enable multidrop networks over shielded twisted pair wiring. It can support segments up to 1200 meters in length or data rates up to 115,200 baud (at this highest data rate the segment length is limited to 1000 meters). An MS/TP link requires only a UART, a 5ms resolution timer, and a [TIA-485-A] transceiver with a driver that can be disabled. These features combine to make MS/TP a cost-effective field bus for the most numerous and least expensive devices in a building automation network.

The differential signaling used by [TIA-485-A] requires a contention-free MAC. MS/TP uses a token to control access to a multidrop bus. A master node may initiate the transmission of a data frame when it holds the token. After sending at most a configured maximum number of data frames, a master node passes the token to the next master node (as determined by node address). Slave nodes transmit only when polled and are not considered part of this specification.

MS/TP frames have the following format*:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      0x55     |      0xFF     |  Frame Type*  |      DA       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      SA       |    Length (MS octet first)    |   Header CRC  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     /
     / Data*
     /
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |              Extended Data CRC* (LS octet first)              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | optional 0xFF |
     +-+-+-+-+-+-+-+-+
```

Figure 1: MS/TP Frame Format

*Note: BACnet [Addendum_an], now in public review, assigns a new Frame Type for IPv6, extends the maximum length of the Data field to 1500 octets, and specifies a 32-bit Extended Data CRC. The Data and Extended Data CRC fields are present only if Length is non-zero.

The MS/TP frame fields have the following descriptions**:

Preamble	two octet preamble: 0x55, 0xFF
Frame Type	one octet
Destination Address	one octet address
Source Address	one octet address
Length	two octets, most significant octet first
Header CRC	one octet
Data	0 - 1500 octets** (present only if Length is non-zero)
Extended Data CRC	four octets**, least significant octet first (present only if Length is non-zero)
(pad)	(optional) at most one octet of trailer: 0xFF

The Frame Type is used to distinguish between different types of MAC frames. Currently defined types (in decimal) are:

00	Token
01	Poll For Master
02	Reply To Poll For Master
	...
10	IPv6 over MS/TP Encapsulation**

**See previous note regarding the BACnet [Addendum_an] change proposal to support IPv6 over MS/TP Encapsulation.

Frame Types 11 through 127 are reserved for assignment by ASHRAE. All master nodes MUST understand Token, Poll For Master, and Reply to Poll For Master frames. See Section 2 for additional details.

The Destination and Source Addresses are each one octet in length. See Section 3 for additional details.

The Length field specifies the length of the Data field in octets and is transmitted most significant octet first. See Section 4 for additional details.

The Header CRC field covers the Frame Type, Destination Address, Source Address, and Length fields. The Header CRC generation and check procedures are specified in [BACnet].

The Data and Extended Data CRC fields are conditional on the Frame Type and the Length. (Note: The Data and Extended Data CRC fields will always be present in frames specified by this document.) The Extended Data CRC generation and check procedures are specified in the BACnet [Addendum_an] change proposal.

1.4. Goals and Non-goals

The primary goal of this specification is to enable IPv6 directly to wired end devices in building automation and control networks, while leveraging existing standards to the greatest extent possible. A secondary goal is to co-exist with legacy MS/TP implementations. Only the minimum changes necessary to support IPv6 over MS/TP are proposed in BACnet [Addendum_an] (see note in Section 1.3).

Non-goals include making changes to the MS/TP frame header format, control frames, Master Node state machine, or addressing modes. Also, while the techniques described here may be applicable to other data links, no attempt is made to define a general design pattern.

2. MS/TP Mode for IPv6

The BACnet [Addendum_an] change proposal allocates a new MS/TP Frame Type from the ASHRAE reserved range to indicate IPv6 encapsulation. The new Frame Type for IPv6 over MS/TP Encapsulation is 10 (0x0A).

All MS/TP master nodes (including those that support IPv6) must understand Token, Poll For Master, and Reply to Poll For Master control frames and support the Master Node state machine as specified in [BACnet]. MS/TP master nodes that support IPv6 must also support the Receive Frame state machine as specified in [BACnet] and extended by [Addendum_an].

3. Addressing Modes

MS/TP node (link-layer) addresses are one octet in length. The method of assigning node addresses is outside the scope of this document. However, each MS/TP node on the link MUST have a unique address or a misconfiguration condition exists.

[BACnet] specifies that addresses 0 through 127 are valid for master nodes. The method specified in Section 6 for creating the Interface Identifier (IID) ensures that an IID of all zeros can never result.

A Destination Address of 255 (0xFF) denotes a link-level broadcast (all nodes). A Source Address of 255 MUST NOT be used. MS/TP does not support multicast, therefore all IPv6 multicast packets MUST be sent as link-level broadcasts and filtered at the IPv6 layer.

This document assumes that each MS/TP link maps to a unique IPv6 subnet prefix. Hosts learn IPv6 prefixes via router advertisements according to [RFC4861].

4. Maximum Transmission Unit (MTU)

The BACnet [Addendum_an] change proposal specifies that the MSDU be increased to 1500 octets and covered by a 32-bit CRC. This is sufficient to convey an MTU of at least 1280 octets as required by IPv6 without the need for link-layer fragmentation and reassembly.

However, the relatively low data rates of MS/TP still make a compelling case for header compression. An adaptation layer to indicate compressed or uncompressed IPv6 headers is specified below in Section 5 and the compression scheme is specified in Section 10.

5. LoBAC Adaptation Layer

The encapsulation formats defined in this section (subsequently referred to as the "LoBAC" encapsulation) comprise the payload (MSDU) of an MS/TP frame. The LoBAC payload (e.g., an IPv6 packet) follows an encapsulation header stack. LoBAC is a subset of the LoWPAN encapsulation defined in [RFC4944], therefore the use of "LOWPAN" in literals below is intentional. The primary differences between LoBAC and LoWPAN are: a) exclusion of the Fragmentation, Mesh, and Broadcast headers, and b) use of LOWPAN_IPHC [RFC6282] in place of LOWPAN_HC1 header compression (which is deprecated by [RFC6282]).

All LoBAC encapsulated datagrams transmitted over MS/TP are prefixed by an encapsulation header stack. Each header in the stack consists of a header type followed by zero or more header fields. Whereas in an IPv6 header the stack would contain, in the following order, addressing, hop-by-hop options, routing, fragmentation, destination options, and finally payload [RFC2460]; in a LoBAC encapsulation the analogous sequence is (optional) header compression and payload. The header stacks that are valid in a LoBAC network are shown below.

A LoBAC encapsulated IPv6 datagram:

```
+---------------+-------------+---------+
| IPv6 Dispatch | IPv6 Header | Payload |
+---------------+-------------+---------+
```

A LoBAC encapsulated LOWPAN_IPHC compressed IPv6 datagram:

```
+---------------+-------------+---------+
| IPHC Dispatch | IPHC Header | Payload |
+---------------+-------------+---------+
```

All protocol datagrams (e.g., IPv6 or compressed IPv6 headers) SHALL be preceded by one of the valid LoBAC encapsulation headers. This permits uniform software treatment of datagrams without regard to their mode of transmission.

The definition of LoBAC headers consists of the dispatch value, the definition of the header fields that follow, and their ordering constraints relative to all other headers. Although the header stack structure provides a mechanism to address future demands on the LoBAC (LoWPAN) adaptation layer, it is not intended to provided general purpose extensibility. This format document specifies a small set of header types using the header stack for clarity, compactness, and orthogonality.

5.1. Dispatch Type and Header

A LoBAC Dispatch type begins with a "0" bit followed by a "1" bit. The Dispatch type and header are shown here:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1| Dispatch  |  Type-specific header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Dispatch	6-bit selector. Identifies the type of header immediately following the Dispatch Header.
Type-specific header	A header determined by the Dispatch Header.

Figure 2: Dispatch Type and Header

The Dispatch value may be treated as an unstructured namespace. Only a few symbols are required to represent current LoBAC functionality. Although some additional savings could be achieved by encoding additional functionality into the dispatch octet, these measures would tend to constrain the ability to address future alternatives.

Pattern	Header Type
00 xxxxxx	NALP - Not a LoWPAN (LoBAC) frame
01 000000	ESC - Additional Dispatch octet follows
01 000001	IPv6 - Uncompressed IPv6 Addresses
...	reserved - Defined or reserved by [RFC4944]
01 1xxxxx	LOWPAN_IPHC - LOWPAN_IPHC compressed IPv6 [RFC6282]
1x xxxxxx	reserved - Defined or reserved by [RFC4944]

Figure 3: Dispatch Value Bit Patterns

NALP: Specifies that the following bits are not a part of the LoBAC encapsulation, and any LoBAC node that encounters a Dispatch value of 00xxxxxx shall discard the packet. Non-LoBAC protocols that wish to coexist with LoBAC nodes should include an octet matching this pattern immediately following the MS/TP header.

ESC: Specifies that the following header is a single 8-bit field for the Dispatch value. It allows support for Dispatch values larger than 127 (see [RFC6282] section 5).

IPv6: Specifies that the following header is an uncompressed IPv6 header [RFC2460].

LOWPAN_IPHC: A value of 011xxxxx specifies a LOWPAN_IPHC compression header (see Section 10.)

Reserved: A LoBAC node that encounters a Dispatch value in the range 01000010 through 01011111 or 1xxxxxxx SHALL discard the packet.

6. Stateless Address Autoconfiguration

This section defines how to obtain an IPv6 Interface Identifier. The general procedure is described in Appendix A of [RFC4291], "Creating Modified EUI-64 Format Interface Identifiers".

The Interface Identifier may be based on an [EUI-64] identifier assigned to the device (but this is not typical for MS/TP). In this case, the Interface Identifier is formed from the EUI-64 by inverting the "u" (universal/local) bit according to [RFC4291]. This will result in a globally unique Interface Identifier.

If the device does not have an EUI-64, then the Interface Identifier MUST be formed by concatenating its 8-bit MS/TP node address to the seven octets 0x00, 0x00, 0x00, 0xFF, 0xFE, 0x00, 0x00. For example, an MS/TP node address of hexadecimal value 0x4F results in the following Interface Identifier:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|0000000000000000|0000000011111111|1111111000000000|0000000001001111|
+----------------+----------------+----------------+----------------+
```

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

An IPv6 address prefix used for stateless autoconfiguration [RFC4862] of an MS/TP interface MUST have a length of 64 bits.

7. IPv6 Link Local Address

The IPv6 link-local address [RFC4291] for an MS/TP interface is formed by appending the Interface Identifier, as defined above, to the prefix FE80::/64.

```
  10 bits            54 bits                  64 bits
+----------+-----------------------+----------------------------+
|1111111010|         (zeros)       |    Interface Identifier    |
+----------+-----------------------+----------------------------+
```

8. Unicast Address Mapping

The address resolution procedure for mapping IPv6 non-multicast addresses into MS/TP link-layer addresses follows the general description in Section 7.2 of [RFC4861], unless otherwise specified.

The Source/Target Link-layer Address option has the following form when the addresses are 8-bit MS/TP node (link-layer) addresses.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length=1   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00      | MS/TP Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-          Padding            -+
|         (all zeros)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option fields:

Type:

1: for Source Link-layer address.

2: for Target Link-layer address.

Length: This is the length of this option (including the type and length fields) in units of 8 octets. The value of this field is 1 for 8-bit MS/TP node addresses.

MS/TP Address: The 8-bit address in canonical bit order [RFC2469]. This is the unicast address the interface currently responds to.

9. Multicast Address Mapping

All IPv6 multicast packets MUST be sent to MS/TP Destination Address 255 (broadcast) and filtered at the IPv6 layer. When represented as a 16-bit address in a compressed header (see Section 10), it MUST be formed by padding on the left with a zero:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00      |     0xFF      |
+-+-+-+-+-+-+-+-+---------------+
```

10. Header Compression

LoBAC uses LOWPAN_IPHC IPv6 compression, which is specified in [RFC6282] and included herein by reference. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282].

In general the following substitutions should be made:

* Replace "6LoWPAN" with "MS/TP network"

* Replace "IEEE 802.15.4 address" with "MS/TP address"

When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short address") it MUST be formed by padding the MS/TP address to the left with a zero:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00      | MS/TP address |
+-+-+-+-+-+-+-+-+---------------+
```

11. IANA Considerations

This document uses values previously reserved by [RFC4944] and [RFC6282] and makes no further requests of IANA.

Note to RFC Editor: this section may be removed upon publication.

12. Security Considerations

The method of deriving Interface Identifiers from MAC addresses is intended to preserve global uniqueness when possible. However, there is no protection from duplication through accident or forgery.

13. Acknowledgments

We are grateful to the authors of [RFC4944] and members of the IETF 6LoWPAN working group; this document borrows extensively from their work.

14. References

14.1. Normative References

[BACnet] American Society of Heating, Refrigerating, and Air-Conditioning Engineers, "BACnet, A Data Communication Protocol for Building Automation and Control Networks (ANSI Approved)", ANSI/ASHRAE 135-2010, April 2011.

[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.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 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.

14.2. Informative References

[Addendum_an]
ASHRAE, "BSR/ASHRAE Addendum an to ANSI/ASHRAE Standard 135-2010, BACnet - A Data Communication Protocol for Building Automation and Control Networks (Advisory Public Review Draft)", September 2011, <https://osr.ashrae.org/default.aspx>.

[EUI-64] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", March 1997, <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

[RFC2469] Narten, T. and C. Burton, "A Caution On The Canonical Ordering Of Link-Layer Addresses", RFC 2469, December 1998.

[TIA-485-A]
Telecommunications Industry Association, "TIA-485-A, Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-485-A-98) (R2003)", March 2003.

Authors’ Addresses

Kerry Lynn (editor)
Consultant

Phone: +1 978 460 4253
Email: kerlyn@ieee.org

Jerry Martocci
Johnson Controls, Inc.
507 E. Michigan St
Milwaukee, WI 53202
USA

Phone: +1 414 524 4010
Email: jerald.p.martocci@jci.com

Carl Neilson
Delta Controls, Inc.
17850 56th Ave
Surrey, BC V3S 1C7
Canada

Phone: +1 604 575 5913

Email: cneilson@deltacontrols.com

Stuart Donaldson
Honeywell Automation & Control Solutions
6670 185th Ave NE
Redmond, WA 98052
USA

Email: stuart.donaldson@honeywell.com

Network Working Group D. Zhang
Internet-Draft S. Jiang
Intended status: Standards Track Huawei Technologies Co., Ltd
Expires: March 19, 2012 B. Carpenter
Univ. of Auckland
September 16, 2011

An Offset Indicating Option for IPv6
draft-zhang-6man-offset-option-01

Abstract

This document defines an Offset Indicating option (OI option) encapsulated within an IPv6 Options header. An OI option can provide offset information to locate the end of the IPv6 header chain so that a node receiving an IPv6 packet is able to skip over the IP header chain and access the transport header or other protocol data unit directly.

Status of this Memo

This Internet-Draft will expire on March 19, 2012.

Copyright Notice

Table of Contents

1. Introduction

According to [RFC2460], when a node intends to access the payload of an IPv6 packet, it needs to parse the extension headers one by one until it reaches the end of the header chain. This approach may be inefficient for nodes which have no interest in the extension headers and intend to quickly access the payload of IPv6 packets.

A common case is any form of flow classification requiring access to the basic IP header 5-tuple {destination address, source address, protocol, destination port, source port}. The last three elements are only available by following the extension header chain to its end. This could be required for various forms of quality of service support or for flow logging purposes. Another case would be any form of deep packet inspection requiring rapid access to the payload, which also requires skipping over the header chain. If packets must be processed at line speed, this can be a significant performance issue. A method is needed to short-circuit this process.

A brief discussion of this issue from a security standpoint is provided in Section 2.1.9.2 of [RFC4942]. In addition, most existing firewall implementations have the capability to verify the correctness of IP headers. Therefore, in some cases, it may be more efficient for the equipment behind a firewall, such as a host or a deep packet inspection device, to skip over the extension headers of the IP packets it receives and access the payload directly.

This document addresses this issue by introducing an Offset Indicating option (OI option for short) which indicates the end of the header chain. The option is transferred in an IPv6 Options header. If there is an existing Hop-by-Hop Options header, the OI option will be in it. Otherwise, it will be in a Destination Options header. According to the recommendations in [RFC2460], this will always place the OI option at the beginning of the header chain. Therefore, if necessary, a node receiving an IPv6 packet can jump over the whole header chain in a single step to directly access the transport header or other protocol data unit.

This option is an optimization option for certain forwarding nodes. It may be safely ignored by nodes that have no interest in the header chain. Hence, it does not create any performance degradation. In particular, unless there is a Hop-by-Hop Options header for some other reason, it does not create any overhead for simple forwarding nodes.

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. Format of the Offset Indicating option

The format of the Offset Indicating option (OI) option is described in Figure 1.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |   Offset                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| NH after Jump |
+-+-+-+-+-+-+-+-+
```

Figure 1. Option Format

Option Type: 8 bits. The value is TBD1.

Note to RFC Editor: please replace TBD1 with the value assigned by IANA and delete this note.

Opt Data Len: as defined in [RFC2460].

Offset: 16 bits. Indicates the distance (in octets) from the end of the option to the end of the header chain.

NH (Next Header) after Jump: 8 bits. Indicates the type of the transport header or other protocol data unit after the header chain. This MUST equal the Next Header value in the last Extension Header in the packet.

4. Processing Rules

IPv6 source nodes SHOULD insert this option in every packet that contains at least one extension header of any kind, in order to maximise its usefulness. However, it MUST NOT be inserted in packets that include a Fragment Header, to avoid the case where the offset points beyond the end of the first fragment. In any case, performance optimisation is impossible in the case of fragmented packets.

Because the options within a header must be processed strictly in the order that they appear, the OI option is RECOMMENDED to be the first option within an Options header. This arrangement will maximize the effect of optimization for those routers that use it.

A Hop-by-Hop Options header MUST NOT be created solely for the purpose of carrying the OI option. If and only if the packet contains a Hop-by-Hop Options header for some other reason, the OI option is placed in it. Otherwise it is placed in a Destination Options header.

This option has an alignment requirement of 4n + 2. (See Section 4.2 of [RFC2460] for discussion of option alignment.) If this option is located first within the Options header, the alignment reqirement is met naturally; otherwise the host stack that assembles the IPv6 header needs to meet the alignment requirement according to the context by inserting padding options.

The OI option is defined on the basis that the size of extension headers does not change en-route. However, if a future extension header type allows an intermediate device to add additional information in the IP extension header chain, this device MUST also update the value of the Offset field to point to the new position of the payload header.

If an intermediate device detects that the OI option does not point to a valid transport header, the IPv6 packet MUST be discarded.

5. Security Considerations

The OI option provides a method for nodes which have no interest in parsing the header chain to quickly process IP packets. Because transport layer security protocols do not cover extension headers, and the information in the IPv6 header is sufficient to generate the pseudo-header for upper layer protocols, the skipping of extension headers will not impact the security verification performed by transport layer security protocols. However, in IPsec the situation is a little different. Because the ESP header [RFC4303] or the AH header [RFC4302] consist of critical information to process the IPsec packet and the extension headers after the ESP or AH header may have to be authenticated or encrypted, these extension headers cannot be skipped over. Therefore, a IPsec implementation MUST NOT skip to the end of the header chain under the instruction of the OI option.

This specification disallows use of the OI option in fragmented packets. In addition to efficiency considerations, this prevents the option from becoming a vector for a buffer overflow attack.

Attackers cannot use the OI option to hide any undesired information in the IPv6 header, because this option is only an optional indication for intermediate devices that do not in any case wish to inspect such information. Security devices may simply ignore this indication and verify every extension header in the chain.

6. IANA Considerations

IANA is requested to assign the IPv6 Option Type TBD1 for the Offset Indicating Option and record it in the IPv6 Destination Options and Hop-by-Hop Options registry.

In accordance with Section 4.2 of [RFC2460], this option type has the two most significant bits set to 00 (skip if unrecognized) and the third-highest-order bit set to 1 (option data may change en-route). This is in case a future IPv6 extension header type may be defined whose size may change en-route, requiring the Offset value to be updated.

Note to RFC Editor: please replace TBD1 with the value assigned by IANA and delete this note.

7. Acknowledgements

Valuable comments on this draft were made by Thomas Narten.

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.

8.2. Informative References

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/

Co-existence Security Considerations", RFC 4942,
September 2007.

Authors' Addresses

Dacheng Zhang
Huawei Technologies Co., Ltd
Huawei Building, No.3 Xinxi Rd.,
Shang-Di Information Industry Base, Hai-Dian District, Beijing
P.R. China

Email: zhangdacheng@huawei.com

Sheng Jiang
Huawei Technologies Co., Ltd
Huawei Building, No.3 Xinxi Rd.,
Shang-Di Information Industry Base, Hai-Dian District, Beijing
P.R. China

Email: jiangsheng@huawei.com

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

Email: brian.e.carpenter@gmail.com