

roll
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2017

P. Thubert, Ed.
Cisco
C. Bormann
Uni Bremen TZI
L. Toutain
IMT-TELECOM Bretagne
R. Cragie
ARM
October 27, 2016

6LoWPAN Routing Header
draft-ietf-roll-routing-dispatch-05

Abstract

This specification introduces a new 6LoWPAN dispatch type for use in 6LoWPAN Route-Over topologies, that initially covers the needs of RPL (RFC6550) data packets compression. Using this dispatch type, this specification defines a method to compress RPL Option (RFC6553) information and Routing Header type 3 (RFC6554), an efficient IP-in-IP technique and is extensible for more applications.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 30, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Terminology	6
3. Using the Page Dispatch	6
3.1. New Routing Header Dispatch (6LoRH)	6
3.2. Placement Of 6LoRH headers	7
3.2.1. Relative To Non-6LoRH Headers	7
3.2.2. Relative To Other 6LoRH Headers	7
4. 6LoWPAN Routing Header General Format	8
4.1. Elective Format	9
4.2. Critical Format	9
4.3. Compressing Addresses	10
4.3.1. Coalescence	10
4.3.2. DODAG Root Address Determination	11
5. The SRH 6LoRH Header	12
5.1. Encoding	12
5.2. SRH-6LoRH General Operation	13
5.2.1. Uncompressed SRH Operation	13
5.2.2. 6LoRH-Compressed SRH Operation	14
5.2.3. Inner LOWPAN_IPHC Compression	14
5.3. The Design Point of Popping Entries	15
5.4. Compression Reference for SRH-6LoRH header entries	16
5.5. Popping Headers	17
5.6. Forwarding	17
6. The RPL Packet Information 6LoRH	18
6.1. Compressing the RPLInstanceID	19
6.2. Compressing the SenderRank	20
6.3. The Overall RPI-6LoRH encoding	20
7. The IP-in-IP 6LoRH Header	23
8. Management Considerations	24
9. Security Considerations	25
10. IANA Considerations	26
10.1. Reserving Space in 6LoWPAN Dispatch Page 1	26
10.2. New Critical 6LoWPAN Routing Header Type Registry	26
10.3. New Elective 6LoWPAN Routing Header Type Registry	26
11. Acknowledgments	27
12. References	27
12.1. Normative References	27
12.2. Informative References	28
Appendix A. Examples	29

A.1. Examples Compressing The RPI	29
A.2. Example Of Downward Packet In Non-Storing Mode	31
A.3. Example of SRH-6LoRH life-cycle	33
Authors' Addresses	35

1. Introduction

The design of Low Power and Lossy Networks (LLNs) is generally focused on saving energy, a very constrained resource in most cases. The other constraints, such as the memory capacity and the duty cycling of the LLN devices, derive from that primary concern. Energy is often available from primary batteries that are expected to last for years, or is scavenged from the environment in very limited quantities. Any protocol that is intended for use in LLNs must be designed with the primary concern of saving energy as a strict requirement.

Controlling the amount of data transmission is one possible venue to save energy. In a number of LLN standards, the frame size is limited to much smaller values than the guaranteed IPv6 maximum transmission unit (MTU) of 1280 bytes. In particular, an LLN that relies on the classical Physical Layer (PHY) of IEEE 802.15.4 [IEEE802154] is limited to 127 bytes per frame. The need to compress IPv6 packets over IEEE 802.15.4 led to the "6LoWPAN Header Compression" [RFC6282] work (6LoWPAN_HC).

Innovative Route-over techniques have been and are still being developed for routing inside a LLN. In a general fashion, such techniques require additional information in the packet to provide loop prevention and to indicate information such as flow identification, source routing information, etc.

For reasons such as security and the capability to send ICMPv6 errors (see "Internet Control Message Protocol (ICMPv6)" [RFC4443]) back to the source, an original packet must not be tampered with, and any information that must be inserted in or removed from an IPv6 packet must be placed in an extra IP-in-IP encapsulation.

This is the case when the additional routing information is inserted by a router on the path of a packet, for instance the root of a mesh, as opposed to the source node, with the non-storing mode of the "IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550] (RPL).

This is also the case when some routing information must be removed from a packet that flows outside the LLN.

"When to use RFC 6553, RFC 6554 and IPv6-in-IPv6"
[I-D.ietf-roll-useofrplinfo] details different cases where IPv6

headers defined in the "RPL Option for Carrying RPL Information in Data-Plane Datagrams" [RFC6553] and the "Routing Header for Source Routes with RPL" [RFC6554], and IPv6-in-IPv6 encapsulation, are inserted or removed from packets in a LLN environments operating RPL.

When using RFC 6282 [RFC6282] the outer IP header of an IP-in-IP encapsulation may be compressed down to 2 octets in stateless compression and down to 3 octets in stateful compression when context information must be added.

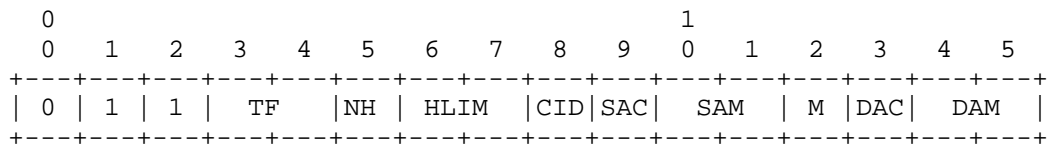


Figure 1: LOWPAN_IPHC base Encoding (RFC6282).

The Stateless Compression of an IPv6 addresses can only happen if the IPv6 address can be deduced from the MAC addresses, meaning that the IP end point is also the MAC-layer endpoint. This is generally not the case in a RPL network which is generally a multi-hop route-over (i.e., operated at Layer-3) network. A better compression, which does not involve variable compressions depending on the hop in the mesh, can be achieved based on the fact that the outer encapsulation is usually between the source (or destination) of the inner packet and the root. Also, the inner IP header can only be compressed by RFC 6282 [RFC6282] if all the fields preceding it are also compressed. This specification makes the inner IP header the first header to be compressed by RFC 6282 [RFC6282], and keeps the inner packet encoded the same way whether it is encapsulated or not, thus preserving existing implementations.

As an example, RPL [RFC6550] is designed to optimize the routing operations in constrained LLNs. As part of this optimization, RPL requires the addition of RPL Packet Information (RPI) in every packet, as defined in Section 11.2 of RFC 6550 [RFC6550].

The "RPL Option for Carrying RPL Information in Data-Plane Datagrams" [RFC6553] specification indicates how the RPI can be placed in a RPL Option (RPL-OPT) that is placed in an IPv6 Hop-by-Hop header.

This representation demands a total of 8 bytes, while in most cases the actual RPI payload requires only 19 bits. Since the Hop-by-Hop header must not flow outside of the RPL domain, it must be inserted in packets entering the domain and be removed from packets that leave the domain. In both cases, this operation implies an IP-in-IP encapsulation.

Additionally, in the case of the Non-Storing Mode of Operation (MOP), RPL requires a Source Routing Header (SRH) in all packets that are routed down a RPL graph. for that purpose, the "IPv6 Routing Header for Source Routes with RPL" [RFC6554] specification defines the type 3 Routing Header for IPv6 (RH3).

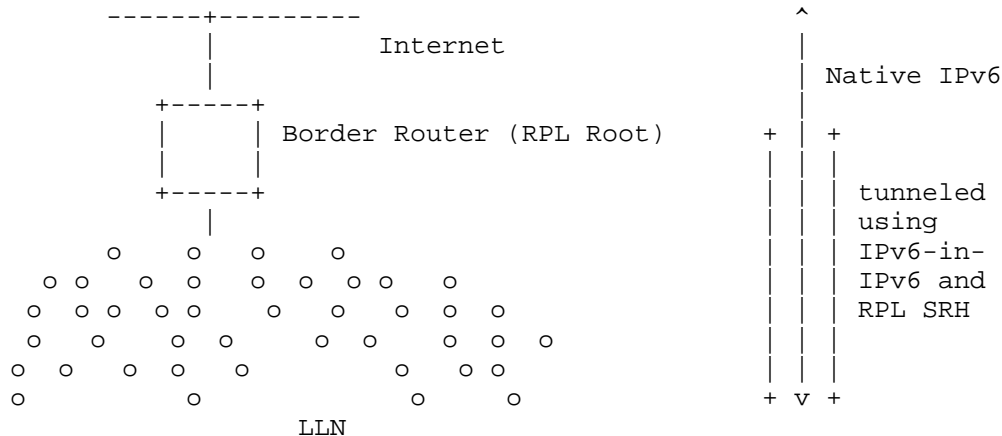


Figure 2: IP-in-IP Encapsulation within the LLN.

With Non-Storing RPL, even if the source is a node in the same LLN, the packet must first reach up the graph to the root so that the root can insert the SRH to go down the graph. In any fashion, whether the packet was originated in a node in the LLN or outside the LLN, and regardless of whether the packet stays within the LLN or not, as long as the source of the packet is not the root itself, the source-routing operation also implies an IP-in-IP encapsulation at the root in order to insert the SRH.

"The 6TiSCH Architecture" [I-D.ietf-6tisch-architecture] specifies the operation of IPv6 over the "TimeSlotted Channel Hopping" [RFC7554] (TSCH) mode of operation of IEEE 802.15.4. The architecture requires the use of both RPL and the 6lo adaptation layer over IEEE 802.15.4. Because it inherits the constraints on frame size from the MAC layer, 6TiSCH cannot afford to allocate 8 bytes per packet on the RPI. Hence the requirement for 6LoWPAN header compression of the RPI.

An extensible compression technique is required that simplifies IP-in-IP encapsulation when it is needed, and optimally compresses existing routing artifacts found in RPL LLNs.

This specification extends the 6lo adaptation layer framework (RFC 4944 [RFC4944] and RFC 6282 [RFC6282]) so as to carry routing information for route-over networks based on RPL. The specification includes the formats necessary for RPL and is extensible for additional formats.

2. Terminology

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

The Terminology used in this document is consistent with and incorporates that described in Terminology in Low power And Lossy Networks [RFC7102] and RPL [RFC6550].

The terms Route-over and Mesh-under are defined in RFC 6775 [RFC6775].

Other terms in use in LLNs are found in "Terminology for Constrained-Node Networks" [RFC7228].

The term "byte" is used in its now customary sense as a synonym for "octet".

3. Using the Page Dispatch

The 6LoWPAN Paging Dispatch [I-D.ietf-6lo-paging-dispatch] specification extends the 6lo adaptation layer framework (RFC 4944 [RFC4944] and RFC 6282 [RFC6282]) by introducing a concept of "context" in the 6LoWPAN parser, a context being identified by a Page number. The specification defines 16 Pages.

This draft operates within Page 1, which is indicated by a Dispatch Value of binary 11110001.

3.1. New Routing Header Dispatch (6LoRH)

This specification introduces a new 6LoWPAN Routing Header (6LoRH) to carry IPv6 routing information. The 6LoRH may contain source routing information such as a compressed form of SRH, as well as other sorts of routing information such as the RPI and IP-in-IP encapsulation.

The 6LoRH is expressed in a 6loWPAN packet as a Type-Length-Value (TLV) field, which is extensible for future use.

It is expected that a router that does not recognize the 6LoRH general format detailed in Section 4 will drop the packet when a 6LoRH is present.

This specification uses the bit pattern 10xxxxxx in Page 1 for the new 6LoRH Dispatch. Section 4 describes how RPL artifacts in data packets can be compressed as 6LoRH headers.

3.2. Placement Of 6LoRH headers

3.2.1. Relative To Non-6LoRH Headers

In a zone of a packet where Page 1 is active (that is, once the Page 1 Paging Dispatch is parsed, and until another Paging Dispatch is parsed as described in the 6LoWPAN Paging Dispatch specification [I-D.ietf-6lo-paging-dispatch]), the parsing of the packet MUST follow this specification if the 6LoRH Bit Pattern (see Section 3.1) is found.

With this specification, the 6LoRH Dispatch is only defined in Page 1, so it MUST be placed in the packet in a zone where the Page 1 context is active.

Because a 6LoRH header requires a Page 1 context, it MUST always be placed after any Fragmentation Header and/or Mesh Header as defined in RFC 4944 [RFC4944].

A 6LoRH header MUST always be placed before the LOWPAN_IPHC as defined in RFC 6282 [RFC6282]. It is designed in such a fashion that placing or removing a header that is encoded with 6LoRH does not modify the part of the packet that is encoded with LOWPAN_IPHC, whether there is an IP-in-IP encapsulation or not. For instance, the final destination of the packet is always the one in the LOWPAN_IPHC whether there is a Routing Header or not.

3.2.2. Relative To Other 6LoRH Headers

The "Internet Protocol, Version 6 (IPv6) Specification" [RFC2460] defines chains of headers that are introduced by an IPv6 header and terminated by either another IPv6 header (IP-in-IP) or an Upper Layer Protocol (ULP) header. When an outer header is stripped from the packet, the whole chain goes with it. When one or more header(s) are inserted by an intermediate router, that router normally chains the headers and encapsulates the result in IP-in-IP.

With this specification, the chains of headers MUST be compressed in the same order as they appear in the uncompressed form of the packet. This means that if there is more than one nested IP-in-IP

encapsulations, the first IP-in-IP encapsulation, with all its chain of headers, is encoded first in the compressed form.

In the compressed form of a packet that has a Source Route or a Hop-by-Hop (HbH) Options Header [RFC2460] after the inner IPv6 header (e.g., if there is no IP-in-IP encapsulation), these headers are placed in the 6LoRH form before the 6LOWPAN_IPHC that represents the IPv6 header (see Section 3.2.1). If this packet gets encapsulated and some other SRH or HbH Options Headers are added as part of the encapsulation, placing the 6LoRH headers next to one another may present an ambiguity on which header belong to which chain in the uncompressed form.

In order to disambiguate the headers that follow the inner IPv6 header in the uncompressed form from the headers that follow the outer IP-in-IP header, it is REQUIRED that the compressed IP-in-IP header is placed last in the encoded chain. This means that the 6LoRH headers that are found after the last compressed IP-in-IP header are to be inserted after the IPv6 header that is encoded with the 6LOWPAN_IPHC when decompressing the packet.

With regards to the relative placement of the SRH and the RPI in the compressed form, it is a design point for this specification that the SRH entries are consumed as the packet progresses down the LLN (see Section 5.3). In order to make this operation simpler in the compressed form, it is REQUIRED that in the compressed form, the addresses along the source route path are encoded in the order of the path, and that the compressed SRH are placed before the compressed RPI.

4. 6LoWPAN Routing Header General Format

The 6LoRH uses the Dispatch Value Bit Pattern of 10xxxxxx in Page 1.

The Dispatch Value Bit Pattern is split in two forms of 6LoRH:

- Elective (6LoRHE) that may skipped if not understood

- Critical (6LoRHC) that may not be ignored

For each form, a Type field is used to encode the type of 6LoRH.

Note that there is a different registry for the Type field of each form of 6LoRH.

This means that a value for the Type that is defined for one form of 6LoRH may be redefined in the future for the other form.

4.1. Elective Format

The 6LoRHE uses the Dispatch Value Bit Pattern of 101xxxxx. A 6LoRHE may be ignored and skipped in parsing. If it is ignored, the 6LoRHE is forwarded with no change inside the LLN.

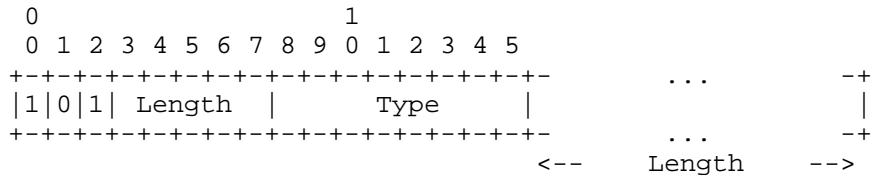


Figure 3: Elective 6LoWPAN Routing Header.

Length: Length of the 6LoRHE expressed in bytes, excluding the first 2 bytes. This enables a node to skip a 6LoRHE header that it does not support and/or cannot parse, for instance if the Type is not recognized.

Type: Type of the 6LoRHE

4.2. Critical Format

The 6LoRHC uses the Dispatch Value Bit Pattern of 100xxxxx.

A node which does not support the 6LoRHC Type MUST silently discard the packet.

Note: the situation where a node receives a message with a Critical 6LoWPAN Routing Header that it does not understand should not occur and is an administrative error, see Section 8.

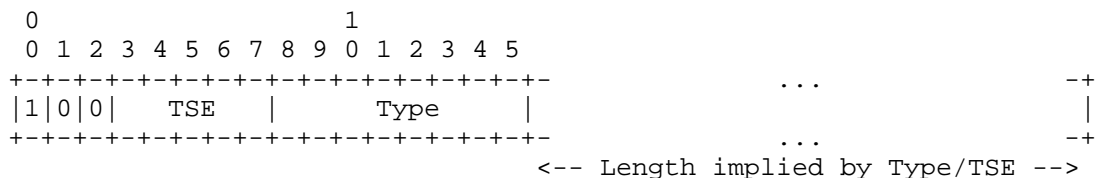


Figure 4: Critical 6LoWPAN Routing Header.

TSE: Type Specific Extension. The meaning depends on the Type, which must be known in all of the nodes. The interpretation of the TSE depends on the Type field that follows. For instance, it may be used to transport control bits, the number of elements in an array, or the length of the remainder of the 6LoRHC expressed in a unit other than bytes.

4.3.2. DODAG Root Address Determination

Stateful Address compression requires that some state is installed in the devices to store the compression information that is elided from the packet. That state is stored in an abstract context table and some form of index is found in the packet to obtain the compression information from the context table.

With RFC 6282 [RFC6282], the state is provided to the stack by the "6LoWPAN Neighbor Discovery Protocol (NDP)" [RFC6775]. NDP exchanges the context through 6LoWPAN Context Option in Router Advertisement (RA) messages. In the compressed form of the packet, the context can be signaled in a Context Identifier Extension.

With this specification, the compression information is provided to the stack by RPL, and RPL exchanges it through the DODAGID field in the DAG Information Object (DIO) messages, as described in more detail below. In the compressed form of the packet, the context can be signaled in by the RPLInstanceID in the RPI.

With RPL [RFC6550], the address of the DODAG root is known from the DODAGID field of the DIO messages. For a Global Instance, the RPLInstanceID that is present in the RPI is enough information to identify the DODAG that this node participates to and its associated root. But for a Local Instance, the address of the root MUST be explicit, either in some device configuration or signaled in the packet, as the source or the destination address, respectively.

When implicit, the address of the DODAG root MUST be determined as follows:

If the whole network is a single DODAG then the root can be well-known and does not need to be signaled in the packets. But since RPL does not expose that property, it can only be known by a configuration applied to all nodes.

Else, the router that encapsulates the packet and compresses it with this specification MUST also place an RPI in the packet as prescribed by RPL to enable the identification of the DODAG. The RPI must be present even in the case when the router also places an SRH header in the packet.

It is expected that the RPL implementation maintains an abstract context table, indexed by Global RPLInstanceID, that provides the address of the root of the DODAG that this nodes participates to for that particular RPL Instance.

5. The SRH 6LoRH Header

5.1. Encoding

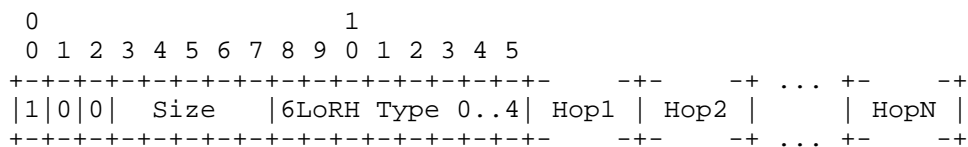
A Source Routing Header 6LoRH (SRH-6LoRH) header provides a compressed form for the SRH, as defined in RFC 6554 [RFC6554] for use by RPL routers.

One or more SRH-6LoRH header(s) MAY be placed in a 6LoWPAN packet.

If a non-RPL router receives a packet with a SRH-6LoRH header, there was a routing or a configuration error (see Section 8).

The desired reaction for the non-RPL router is to drop the packet as opposed to skip the header and forward the packet.

The Dispatch Value Bit Pattern for the SRH-6LoRH header indicates Critical. Routers that understand the 6LoRH general format detailed in Section 4 cannot ignore a 6LoRH header of this type, and will drop the packet if it is unknown to them.



Where $N = \text{Size} + 1$

Figure 6: The SRH-6LoRH.

The 6LoRH Type of a SRH-6LoRH header indicates the compression level used for that header.

The fields following the 6LoRH Type are compressed addresses indicating the consecutive hops, and are ordered from the first to the last hop.

All the addresses in a given SRH-6LoRH header MUST be compressed in an identical fashion, so the Length of the compressed form is the same for all.

In order to get different degrees of compression, multiple consecutive SRH-6LoRH headers MUST be used.

Type 0 means that the address is compressed down to one byte, whereas Type 4 means that the address is provided in full in the SRH-6LoRH

with no compression. The complete list of Types of SRH-6LoRH and the corresponding compression level are provided in Figure 7:

6LoRH Type	Length of compressed IPv6 address (bytes)
0	1
1	2
2	4
3	8
4	16

Figure 7: The SRH-6LoRH Types.

In the case of a SRH-6LoRH header, the TSE field is used as a Size, which encodes the number of hops minus 1; so a Size of 0 means one hop, and the maximum that can be encoded is 32 hops. (If more than 32 hops need to be expressed, a sequence of SRH-6LoRH elements can be employed.) It results that the Length in bytes of a SRH-6LoRH header is:

$$2 + \text{Length_of_compressed_IPv6_address} * (\text{Size} + 1)$$

5.2. SRH-6LoRH General Operation

5.2.1. Uncompressed SRH Operation

In the non-compressed form, when the root generates or forwards a packet in non-Storing Mode, it needs to include a Source Routing Header [RFC6554] to signal a strict source-route path to a final destination down the DODAG.

All the hops along the path, but the first one, are encoded in order in the SRH. The last entry in the SRH is the final destination and the destination in the IPv6 header is the first hop along the source-route path. The intermediate hops perform a swap and the Segment-Left field indicates the active entry in the Routing Header [RFC2460].

The current destination of the packet, which is the termination of the current segment, is indicated at all times by the destination address of the IPv6 header.

5.2.2. 6LoRH-Compressed SRH Operation

The handling of the SRH-6LoRH is different: there is no swap, and a forwarding router that corresponds to the first entry in the first SRH-6LoRH upon reception of a packet effectively consumes that entry when forwarding. This means that the size of a compressed source-routed packet decreases as the packet progresses along its path and that the routing information is lost along the way. This also means that an SRH encoded with 6LoRH is not recoverable and cannot be protected.

When compressed with this specification, all the remaining hops **MUST** be encoded in order in one or more consecutive SRH-6LoRH headers. Whether or not there is a SRH-6LoRH header present, the address of the final destination is indicated in the LOWPAN_IPHC at all times along the path. Examples of this are provided in Appendix A.

The current destination (termination of the current segment) for a compressed source-routed packet is indicated in the first entry of the first SRH-6LoRH. In strict source-routing, that entry **MUST** match an address of the router that receives the packet.

The last entry in the last SRH-6LoRH is the last router on the way to the final destination in the LLN. This router can be the final destination if it is found desirable to carry a whole IP-in-IP encapsulation all the way. Else, it is the RPL parent of the final destination, or a router acting at 6LR [RFC6775] for the destination host, and advertising the host as an external route to RPL.

If the SRH-6LoRH header is contained in an IP-in-IP encapsulation, the last router removes the whole chain of headers. Otherwise, it removes the SRH-6LoRH header only.

5.2.3. Inner LOWPAN_IPHC Compression

6LoWPAN ND [RFC6282] is designed to support more than one IPv6 address per node and per Interface Identifier (IID), an IID being typically derived from a MAC address to optimize the LOWPAN_IPHC compression.

Link local addresses are compressed with stateless address compression (S/DAC=0). The other addresses are derived from different prefixes and they can be compressed with stateful address compression based on a context (S/DAC=1).

But stateless compression is only defined for the specific link-local prefix as opposed to the prefix in an encapsulating header. And with

stateful compression, the compression reference is found in a context, as opposed to an encapsulating header.

It results that in the case of an IP-in-IP encapsulation, it is possible to compress an inner source (respectively destination) IP address in a LOWPAN_IPHC based on the encapsulating IP header only if stateful (context-based) compression is used. The compression will operate only if the IID in the source (respectively the destination) IP address in the outer and inner headers match, which usually means that they refer to the same node. This is encoded as S/DAC = 1 and S/AM=11. It must be noted that the outer destination address that is used to compress the inner destination address is the last entry in the last SRH-6LoRH header.

5.3. The Design Point of Popping Entries

In order to save energy and to optimize the chances of transmission success on lossy media, it is a design point for this specification that the entries in the SRH that have been used are removed from the packet. This creates a discrepancy from the art of IPv6 where Routing Header are mutable but recoverable.

With this specification, the packet can be expanded at any hop into a valid IPv6 packet, including a SRH, and compressed back. But the packet as decompressed along the way will not carry all the consumed addresses that packet would have if it had been forwarded in the uncompressed form.

It is noted that:

The value of keeping the whole RH in an IPv6 header is for the receiver to reverse it to use the symmetrical path on the way back.

It is generally not a good idea to reverse a routing header. The RH may have been used to stay away from the shortest path for some reason that is only valid on the way in (segment routing).

There is no use of reversing a RH in the present RPL specifications.

P2P RPL reverses a path that was learned reactively, as a part of the protocol operation, which is probably a cleaner way than a reversed echo on the data path.

Reversing a header is discouraged by RFC 2460 [RFC2460] for RH0 unless it is authenticated, which requires an Authentication

Header (AH). There is no definition of an AH operation for SRH, and there is no indication that the need exists in LLNs.

It is noted that AH does not protect the RH on the way. AH is a validation at the receiver with the sole value of enabling the receiver to reversing it.

A RPL domain is usually protected by L2 security and that secures both RPL itself and the RH in the packets, at every hop. This is a better security than that provided by AH.

In summary, the benefit of saving energy and lowering the chances of loss by sending smaller frames over the LLN are seen as overwhelming compared to the value of possibly reversing the header.

5.4. Compression Reference for SRH-6LoRH header entries

In order to optimize the compression of IP addresses present in the SRH headers, this specification requires that the 6LoWPAN layer identifies an address that is used as reference for the compression.

With this specification, the Compression Reference for the first address found in an SRH header is the source of the IPv6 packet, and then the reference for each subsequent entry is the address of its predecessor once it is uncompressed.

With RPL [RFC6550], an SRH header may only be present in Non-Storing mode, and it may only be placed in the packet by the root of the DODAG, which must be the source of the resulting IPv6 packet [RFC2460]. In this case, the address used as Compression Reference is the address of the root.

The Compression Reference MUST be determined as follows:

The reference address may be obtained by configuration. The configuration may indicate either the address in full, or the identifier of a 6LoWPAN Context that carries the address [RFC6775], for instance one of the 16 Context Identifiers used in LOWPAN_IPHC [RFC6282].

Else, and if there is no IP-in-IP encapsulation, the source address in the IPv6 header that is compressed with LOWPAN_IPHC is the reference for the compression.

Else, and if the IP-in-IP compression specified in this document is used and the Encapsulator Address is provided, then the Encapsulator Address is the reference.

Else, meaning that the IP-in-IP compression specified in this document is used and the encapsulator is implicitly the root, the address of the root is the reference.

5.5. Popping Headers

Upon reception, the router checks whether the address in the first entry of the first SRH-6LoRH one of its own addresses. In that case, router MUST consume that entry before forwarding, which is an action of popping from a stack, where the stack is effectively the sequence of entries in consecutive SRH-6LoRH headers.

Popping an entry of an SRH-6LoRH header is a recursive action performed as follows:

If the Size of the SRH-6LoRH header is 1 or more, indicating that there are at least 2 entries in the header, the router removes the first entry and decrements the Size (by 1).

Else (meaning that this is the last entry in the SRH-6LoRH header), and if there is no next SRH-6LoRH header after this then the SRH-6LoRH is removed.

Else, if there is a next SRH-6LoRH of a Type with a larger or equal value, meaning a same or lesser compression yielding same or larger compressed forms, then the SRH-6LoRH is removed.

Else, the first entry of the next SRH-6LoRH is popped from the next SRH-6LoRH and coalesced with the first entry of this SRH-6LoRH.

At the end of the process, if there is no more SRH-6LoRH in the packet, then the processing node is the last router along the source route path.

An example of this operation is provided in Appendix A.3.

5.6. Forwarding

When receiving a packet with a SRH-6LoRH, a router determines the IPv6 address of the current segment endpoint.

If strict source routing is enforced and this router is not the segment endpoint for the packet then this router MUST drop the packet.

If this router is the current segment endpoint, then the router pops its address as described in Section 5.5 and continues processing the packet.

If there is still a SRH-6LoRH, then the router determines the new segment endpoint and routes the packet towards that endpoint.

Otherwise the router uses the destination in the inner IP header to forward or accept the packet.

The segment endpoint of a packet **MUST** be determined as follows:

The router first determines the Compression Reference as discussed in Section 4.3.1.

The router then coalesces the Compression Reference with the first entry of the first SRH-6LoRH header as discussed in Section 5.4. If the type of the SRH-6LoRH header is type 4 then the coalescence is a full override.

Since the Compression Reference is an uncompressed address, the coalesced IPv6 address is also expressed in the full 128bits.

6. The RPL Packet Information 6LoRH

RPL [RFC6550], Section 11.2, specifies the RPL Packet Information (RPI) as a set of fields that are placed by RPL routers in IP packets to identify the RPL Instance, detect anomalies and trigger corrective actions.

In particular, the SenderRank, which is the scalar metric computed by a specialized Objective Function such as described in RFC 6552 [RFC6552], indicates the Rank of the sender and is modified at each hop. The SenderRank field is used to validate that the packet progresses in the expected direction, either upwards or downwards, along the DODAG.

RPL defines the "RPL Option for Carrying RPL Information in Data-Plane Datagrams" [RFC6553] to transport the RPI, which is carried in an IPv6 Hop-by-Hop Options Header [RFC2460], typically consuming eight bytes per packet.

With RFC 6553 [RFC6553], the RPL option is encoded as six octets, which must be placed in a Hop-by-Hop header that consumes two additional octets for a total of eight octets. To limit the header's range to just the RPL domain, the Hop-by-Hop header must be added to (or removed from) packets that cross the border of the RPL domain.

The 8-byte overhead is detrimental to LLN operation, in particular with regards to bandwidth and battery constraints. These bytes may cause a containing frame to grow above maximum frame size, leading to Layer 2 or 6LoWPAN [RFC4944] fragmentation, which in turn leads to

even more energy expenditure and issues discussed in "LLN Fragment Forwarding and Recovery" [I-D.thubert-6lo-forwarding-fragments].

An additional overhead comes from the need, in certain cases, to add an IP-in-IP encapsulation to carry the Hop-by-Hop header. This is needed when the router that inserts the Hop-by-Hop header is not the source of the packet, so that an error can be returned to the router. This is also the case when a packet originated by a RPL node must be stripped from the Hop-by-Hop header to be routed outside the RPL domain.

For that reason, this specification defines an IP-in-IP-6LoRH header in Section 7, but it must be noted that removal of a 6LoRH header does not require manipulation of the packet in the LOWPAN_IPHC, and thus, if the source address in the LOWPAN_IPHC is the node that inserted the IP-in-IP-6LoRH header then this situation alone does not mandate an IP-in-IP-6LoRH header.

Note: it was found that some implementations omit the RPI for packets going down the RPL graph in Non-Storing Mode, even though RPL indicates that the RPI should be placed in the packet. With this specification, the RPI is important to indicate the RPLInstanceID so the RPI should not be omitted.

As a result, a RPL packet may bear only an RPI-6LoRH header and no IP-in-IP-6LoRH header. In that case, the source and destination of the packet are specified by the LOWPAN_IPHC.

As with RFC 6553 [RFC6553], the fields in the RPI include an 'O', an 'R', and an 'F' bit, an 8-bit RPLInstanceID (with some internal structure), and a 16-bit SenderRank.

The remainder of this section defines the RPI-6LoRH header, which is a Critical 6LoWPAN Routing Header that is designed to transport the RPI in 6LoWPAN LLNs.

6.1. Compressing the RPLInstanceID

RPL Instances are discussed in Section 5 of the RPL specification [RFC6550]. A number of simple use cases do not require more than one RPL Instance, and in such cases, the RPL Instance is expected to be the Global Instance 0. A global RPLInstanceID is encoded in a RPLInstanceID field as follows:

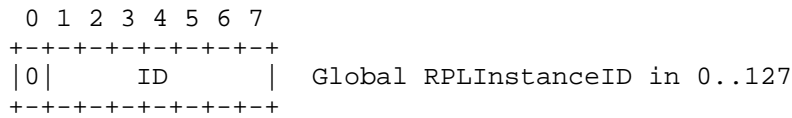


Figure 8: RPLInstanceID Field Format for Global Instances.

For the particular case of the Global Instance 0, the RPLInstanceID field is all zeros. This specification allows to elide a RPLInstanceID field that is all zeros, and defines a I flag that, when set, signals that the field is elided.

6.2. Compressing the SenderRank

The SenderRank is the result of the DAGRank operation on the rank of the sender; here the DAGRank operation is defined in Section 3.5.1 of the RPL specification [RFC6550] as:

$$\text{DAGRank}(\text{rank}) = \text{floor}(\text{rank}/\text{MinHopRankIncrease})$$

If MinHopRankIncrease is set to a multiple of 256, the least significant 8 bits of the SenderRank will be all zeroes; by eliding those, the SenderRank can be compressed into a single byte. This idea is used in RFC 6550 [RFC6550] by defining DEFAULT_MIN_HOP_RANK_INCREASE as 256 and in RFC 6552 [RFC6552] that defaults MinHopRankIncrease to DEFAULT_MIN_HOP_RANK_INCREASE.

This specification allows to encode the SenderRank as either one or two bytes, and defines a K flag that, when set, signals that a single byte is used.

6.3. The Overall RPI-6LoRH encoding

The RPI-6LoRH header provides a compressed form for the RPL RPI. Routers that need to forward a packet with a RPI-6LoRH header are expected to be RPL routers that support this specification.

If a non-RPL router receives a packet with a RPI-6LoRH header, there was a routing or a configuration error (see Section 8).

The desired reaction for the non-RPL router is to drop the packet as opposed to skip the header and forward the packet, which could end up forming loops by reinjecting the packet in the wrong RPL Instance.

The Dispatch Value Bit Pattern for the SRH-6LoRH header indicates Critical. Routers that understand the 6LoRH general format detailed in Section 4 cannot ignore a 6LoRH header of this type, and will drop the packet if it is unknown to them.

Since the RPI-6LoRH header is a critical header, the TSE field does not need to be a length expressed in bytes. In that case the field is fully reused for control bits that encode the O, R and F flags from the RPI, as well as the I and K flags that indicate the compression format.

The Type for the RPI-6LoRH is 5.

The RPI-6LoRH header is immediately followed by the RPLInstanceID field, unless that field is fully elided, and then the SenderRank, which is either compressed into one byte or fully in-lined as two bytes. The I and K flags in the RPI-6LoRH header indicate whether the RPLInstanceID is elided and/or the SenderRank is compressed. Depending on these bits, the Length of the RPI-6LoRH may vary as described hereafter.

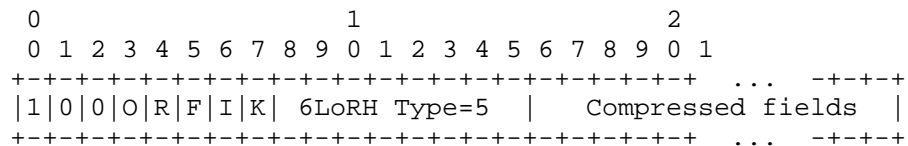


Figure 9: The Generic RPI-6LoRH Format.

O, R, and F bits: The O, R, and F bits are defined in section 11.2 of RFC 6550 [RFC6550].

I flag: If it is set, the RPLInstanceID is elided and the RPLInstanceID is the Global RPLInstanceID 0. If it is not set, the octet immediately following the type field contains the RPLInstanceID as specified in section 5.1 of RFC 6550 [RFC6550].

K flag: If it is set, the SenderRank is compressed into one octet, with the least significant octet elided. If it is not set, the SenderRank, is fully inlined as two octets.

In Figure 10, the RPLInstanceID is the Global RPLInstanceID 0, and the MinHopRankIncrease is a multiple of 256 so the least significant byte is all zeros and can be elided:

```

      0               1               2
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-----+-----+-----+-----+-----+-----+
|1|0|0|0|R|F|1|1| 6LoRH Type=5 | SenderRank |
+-----+-----+-----+-----+-----+
      I=1, K=1

```

Figure 10: The most compressed RPI-6LoRH.

In Figure 11, the RPLInstanceID is the Global RPLInstanceID 0, but both bytes of the SenderRank are significant so it can not be compressed:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|0|0|0|R|F|1|0| 6LoRH Type=5 | SenderRank |
+-----+-----+-----+-----+-----+-----+
      I=1, K=0

```

Figure 11: Eliding the RPLInstanceID.

In Figure 12, the RPLInstanceID is not the Global RPLInstanceID 0, and the MinHopRankIncrease is a multiple of 256:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|0|0|0|R|F|0|1| 6LoRH Type=5 | RPLInstanceID | SenderRank |
+-----+-----+-----+-----+-----+-----+-----+
      I=0, K=1

```

Figure 12: Compressing SenderRank.

In Figure 13, the RPLInstanceID is not the Global RPLInstanceID 0, and both bytes of the SenderRank are significant:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|0|0|0|R|F|0|0| 6LoRH Type=5 | RPLInstanceID | Sender-...
+-----+-----+-----+-----+-----+-----+-----+
...-Rank |
+-----+-----+-----+
      I=0, K=0

```

Figure 13: Least compressed form of RPI-6LoRH.

7. The IP-in-IP 6LoRH Header

The IP-in-IP 6LoRH (IP-in-IP-6LoRH) header is an Elective 6LoWPAN Routing Header that provides a compressed form for the encapsulating IPv6 Header in the case of an IP-in-IP encapsulation.

An IP-in-IP encapsulation is used to insert a field such as a Routing Header or an RPI at a router that is not the source of the packet. In order to send an error back regarding the inserted field, the address of the router that performs the insertion must be provided.

The encapsulation can also enable the last router prior to Destination to remove a field such as the RPI, but this can be done in the compressed form by removing the RPI-6LoRH, so an IP-in-IP-6LoRH encapsulation is not required for that sole purpose.

The Dispatch Value Bit Pattern for the SRH-6LoRH header indicates Elective. This field is not critical for routing since it does not indicate the destination of the packet, which is either encoded in a SRH-6LoRH header or in the inner IP header. A 6LoRH header of this type can be skipped if not understood (per Section 4), and the 6LoRH header indicates the Length in bytes.

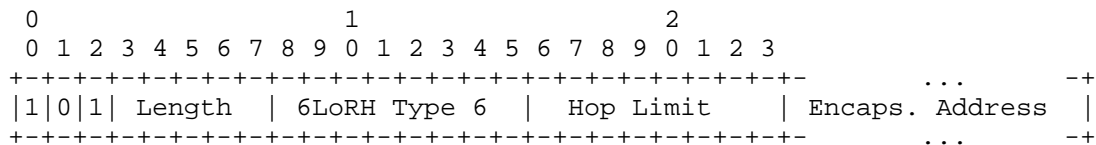


Figure 14: The IP-in-IP-6LoRH.

The Length of an IP-in-IP-6LoRH header is expressed in bytes and MUST be at least 1, to indicate a Hop Limit (HL), that is decremented at each hop. When the HL reaches 0, the packet is dropped per RFC 2460 [RFC2460].

If the Length of an IP-in-IP-6LoRH header is exactly 1, then the Encapsulator Address is elided, which means that the Encapsulator is a well-known router, for instance the root in a RPL graph.

The most efficient compression of an IP-in-IP encapsulation that can be achieved with this specification is obtained when an endpoint of the packet is the root of the RPL DODAG associated to the RPL Instance that is used to forward the packet, and the root address is known implicitly as opposed to signaled explicitly in the data packets.

If the Length of an IP-in-IP-6LoRH header is greater than 1, then an Encapsulator Address is placed in a compressed form after the Hop Limit field. The value of the Length indicates which compression is performed on the Encapsulator Address. For instance, a Length of 3 indicates that the Encapsulator Address is compressed to 2 bytes. The reference for the compression is the address of the root of the DODAG. The way the address of the root is determined is discussed in Section 4.3.2.

With RPL, the destination address in the IP-in-IP header is implicitly the root in the RPL graph for packets going upwards, and, in storing mode, it is the destination address in the LOWPAN_IPHC for packets going downwards. In non-storing mode, there is no implicit value for packets going downwards.

If the implicit value is correct, the destination IP address of the IP-in-IP encapsulation can be elided. Else, the destination IP address of the IP-in-IP header is transported in a SRH-6LoRH header as the first entry of the first of these headers.

If the final destination of the packet is a leaf that does not support this specification, then the chain of 6LoRH headers must be stripped by the RPL/6LR router to which the leaf is attached. In that example, the destination IP address of the IP-in-IP header cannot be elided.

In the special case where a 6LoRH header is used to route 6LoWPAN fragments, the destination address is not accessible in the LOWPAN_IPHC on all fragments and can be elided only for the first fragment and for packets going upwards.

8. Management Considerations

Though it is possible to decompress a packet at any hop, this specification is optimized to enable that a packet is forwarded in its compressed form all the way, and it makes sense to deploy homogeneous networks, where all nodes, or no node at all, use the compression technique detailed therein.

This specification aims at a simple implementation running in constrained nodes, so it does indeed expect an homogeneous network and as a consequence it does not provide a method to determine the level of support by the next hops at forwarding time.

Should an extension to this specification provide such a method, forwarding nodes could compress or uncompress the RPL artifacts appropriately and enable a backward compatibility between nodes that support this specification and nodes that do not.

It results that this specification does not attempt to enable such backwards compatibility. It does not require extraneous code to exchange and handle error messages to correct automatically mismatch situations, either.

When a packet is expected to carry a 6LoRH header but it does not, the node that discovers the issue is expected to send an ICMPv6 error message to the root, at an adapted rate limitation and with a Type 4 indicating a "Parameter Problem", and a Code 0 indicating an "erroneous header field encountered", embedding the relevant portion of the received packet and pointing at the offset therein where the 6LoRH header was expected.

When a packet is received with a 6LoRH header that is not recognized, the node that discovers the issue is expected to send an ICMPv6 error message, to the root, at an adapted rate limitation and with a Type 4 indicating a "Parameter Problem", and a Code 1 indicating an "unrecognized Next Header type", embedding the relevant portion of the received packet and pointing at the offset therein where the 6LoRH header was expected.

In both cases, the node SHOULD NOT place a 6LoRH header defined in this specification in the resulting message, and should either omit the RPI or place it uncompressed after the IPv6 header.

In both cases also, an alternate management method may be preferred in order to notify the network administrator that there is a configuration error.

Keeping the network homogeneous is either a deployment issue, by deploying only devices with a same capability, or a management issue, by configuring all devices to either use, or not use, a certain level of this compression technique and its future additions.

In particular, the situation where a node receives a message with a Critical 6LoWPAN Routing Header that it does not understand is an administrative error whereby the wrong device is placed in a network, or the device is mis-configured.

When a mismatch situation is detected, it is expected that the device raises some management alert, indicating the issue, e.g., that it has to drop a packet with a Critical 6LoRH.

9. Security Considerations

The security considerations of RFC 4944 [RFC4944], RFC 6282 [RFC6282], and RFC 6553 [RFC6553] apply.

Using a compressed format as opposed to the full in-line format is logically equivalent and is believed to not create an opening for a new threat when compared to RFC 6550 [RFC6550], RFC 6553 [RFC6553] and RFC 6554 [RFC6554], noting that, even though intermediate hops are removed from the SRH header as they are consumed, a node may still identify that the rest of the source routed path includes a loop or not (see Security section of RFC 6554). It must be noted that if the attacker is not part of the loop, then there is always a node at the beginning of the loop that can detect it and remove it.

10. IANA Considerations

10.1. Reserving Space in 6LoWPAN Dispatch Page 1

This specification reserves Dispatch Value Bit Patterns within the 6LoWPAN Dispatch Page 1 as follows:

101xxxxx: for Elective 6LoWPAN Routing Headers

100xxxxx: for Critical 6LoWPAN Routing Headers.

Additionally this document creates two IANA registries, one for the Critical 6LoWPAN Routing Header Type and one for the Elective 6LoWPAN Routing Header Type, each with 32 possible values from 0 to 31, as described below.

Future assignments in these registries are to be coordinated via IANA under the policy of "RFC Required" (per RFC 5226 [RFC5226]) to enable any type of RFC to obtain a value in the registry.

10.2. New Critical 6LoWPAN Routing Header Type Registry

This document creates an IANA registry for the Critical 6LoWPAN Routing Header Type, and assigns the following values:

0..4: SRH-6LoRH [RFCthis]

5: RPI-6LoRH [RFCthis]

10.3. New Elective 6LoWPAN Routing Header Type Registry

This document creates an IANA registry for the Elective 6LoWPAN Routing Header Type, and assigns the following value:

6: IP-in-IP-6LoRH [RFCthis]

11. Acknowledgments

The authors wish to thank Tom Phinney, Thomas Watteyne, Tengfei Chang, Martin Turon, James Woodyatt, Samita Chakrabarti, Jonathan Hui, Gabriel Montenegro and Ralph Droms for constructive reviews to the design in the 6lo Working Group. The overall discussion involved participants to the 6MAN, 6TiSCH and ROLL WGs, thank you all. Special thanks to the chairs of the ROLL WG, Michael Richardson and Ines Robles, Brian Haberman, Internet Area A-D, and Alvaro Retana and Adrian Farrel, Routing Area A-Ds, for driving this complex effort across Working Groups and Areas.

12. References

12.1. Normative References

- [I-D.ietf-6lo-paging-dispatch]
Thubert, P. and R. Cragie, "6LoWPAN Paging Dispatch", draft-ietf-6lo-paging-dispatch-05 (work in progress), October 2016.
- [IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<http://www.rfc-editor.org/info/rfc4944>>.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<http://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <<http://www.rfc-editor.org/info/rfc6552>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.

12.2. Informative References

- [I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-10 (work in progress), June 2016.
- [I-D.ietf-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "When to use RFC 6553, 6554 and IPv6-in-IPv6", draft-ietf-roll-useofrplinfo-09 (work in progress), October 2016.

- [I-D.thubert-6lo-forwarding-fragments]
Thubert, P. and J. Hui, "LLN Fragment Forwarding and Recovery", draft-thubert-6lo-forwarding-fragments-03 (work in progress), October 2016.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <<http://www.rfc-editor.org/info/rfc7102>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<http://www.rfc-editor.org/info/rfc7228>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<http://www.rfc-editor.org/info/rfc7554>>.

Appendix A. Examples

A.1. Examples Compressing The RPI

The example in Figure 15 illustrates the 6LoRH compression of a classical packet in Storing Mode in all directions, as well as in non-Storing mode for a packet going up the DODAG following the default route to the root. In this particular example, a fragmentation process takes place per RFC 4944 [RFC4944], and the fragment headers must be placed in Page 0 before switching to Page 1:

```

+-- ... -+- ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...
|Frag type|Frag hdr |11110001| RPI- |IP-in-IP| LOWPAN_IPHC | ...
|RFC 4944|RFC 4944 | Page 1 | 6LoRH | 6LoRH |
+-- ... -+- ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...
                                                    <- RFC 6282 ->
                                                    No RPL artifact

+-- ... -+- ... -+-+ ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...
|Frag type|Frag hdr |
|RFC 4944|RFC 4944 | Payload (cont)
+-- ... -+- ... -+-+ ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...

+-- ... -+- ... -+-+ ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...
|Frag type|Frag hdr |
|RFC 4944|RFC 4944 | Payload (cont)
+-- ... -+- ... -+-+ ... -+-+ ... -+- ... +-+ ... -+-+ ... -+-+ ... -+-+ ...

```

Figure 15: Example Compressed Packet with RPI.

In Storing Mode, if the packet stays within the RPL domain, then it is possible to save the IP-in-IP encapsulation, in which case only the RPI is compressed with a 6LoRH, as illustrated in Figure 16 in the case of a non-fragmented ICMP packet:

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 11110001 | RPI-6LoRH | NH = 0      | NH = 58    | ICMP message ... |
| Page 1   | type 5    | 6LOWPAN_IPHC | (ICMP)     | (no compression) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
                                     <-   RFC 6282   ->
                                     No RPL artifact

```

Figure 16: Example ICMP Packet with RPI in Storing Mode.

The format in Figure 16 is logically equivalent to the non-compressed format illustrated in Figure 17:

```
+-----+--- ... +-----+--- +-----+-----+-----+-----+-----+-----+...
| IPv6 Header | Hop-by-Hop | RPI in      | ICMP message ...
| NH = 58     | Header    | RPL Option  |
+-----+--- ... +-----+--- +-----+-----+-----+-----+-----+-----+...
```

Figure 17: Uncompressed ICMP Packet with RPI.

For a UDP packet, the transport header can be compressed with 6LoWPAN HC [RFC6282] as illustrated in Figure 18:

```

+--+ ... -+--+...-+--+ ... -+--+--+ ... -+--+ ... -+--+--+...
|11110001| RPI-   | NH=1           |11110CPP| Compressed | UDP
|Page 1  | 6LoRH | LOWPAN_IPHC | UDP     | UDP header | Payload
+--+ ... -+--+...-+--+ ... -+--+--+ ... -+--+ ... -+--+--+...
                        <-          RFC 6282          ->
                        No RPL artifact

```

Figure 18: Uncompressed ICMP Packet with RPI.

If the packet is received from the Internet in Storing Mode, then the root is supposed to encapsulate the packet to insert the RPI. The resulting format would be as represented in Figure 19:

```

+--+ ... -+--+...-+--+ ... -+--+--+ ... -+--+ ... -+--+--+ ... -+--+--+...
|11110001| RPI-   | IP-in-IP | NH=1           |11110CPP| Compressed | UDP
|Page 1  | 6LoRH | 6LoRH     | LOWPAN_IPHC | UDP     | UDP header | Payld
+--+ ... -+--+...-+--+ ... -+--+--+ ... -+--+ ... -+--+--+ ... -+--+--+...
                        <-          RFC 6282          ->
                        No RPL artifact

```

Figure 19: RPI inserted by the root in Storing Mode.

A.2. Example Of Downward Packet In Non-Storing Mode

The example illustrated in Figure 20 is a classical packet in non-Storing mode for a packet going down the DODAG following a source routed path from the root. Say that we have 4 forwarding hops to reach a destination. In the non-compressed form, when the root generates the packet, the last 3 hops are encoded in a Routing Header type 3 (SRH) and the first hop is the destination of the packet. The intermediate hops perform a swap and the hop count indicates the current active hop as defined in RFC 2460 [RFC2460] and RFC 6554 [RFC6554].

When compressed with this specification, the 4 hops are encoded in SRH-6LoRH when the root generates the packet, and the final destination is left in the LOWPAN_IPHC. There is no swap, and the forwarding node that corresponds to the first entry effectively consumes it when forwarding, which means that the size of the encoded packet decreases and that the hop information is lost.

If the last hop in a SRH-6LoRH is not the final destination then it removes the SRH-6LoRH before forwarding.

In the particular example illustrated in Figure 20, all addresses in the DODAG are assigned from a same /112 prefix and the last 2 octets encoding an identifier such as a IEEE 802.15.4 short address. In

that case, all addresses can be compressed to 2 octets, using the root address as reference. There will be one SRH_6LoRH header, with, in this example, 3 compressed addresses:

```
+--+ ... -++-+ ... +-+- ... -++-+ ... +-+-+--+ ... +-+-+ ... -+ ... +-...
|11110001|SRH-6LoRH| RPI-  | IP-in-IP | NH=1      |11110CPP| UDP | UDP
|Page 1  |Type1 S=2| 6LoRH | 6LoRH   |LOWPAN_IPHC| UDP   | hdr | Payld
+--+ ... -++-+ ... +-+- ... -++-+ ... +-+-+--+ ... +-+-+ ... -+ ... +-...
<-8bytes->                <-          RFC 6282          ->
                               No RPL artifact
```

Figure 20: Example Compressed Packet with SRH.

One may note that the RPI is provided. This is because the address of the root that is the source of the IP-in-IP header is elided and inferred from the RPLInstanceID in the RPI. Once found from a local context, that address is used as Compression Reference to expand addresses in the SRH-6LoRH.

With the RPL specifications available at the time of writing this draft, the root is the only node that may incorporate a SRH in an IP packet. When the root forwards a packet that it did not generate, it has to encapsulate the packet with IP-in-IP.

But if the root generates the packet towards a node in its DODAG, then it should avoid the extra IP-in-IP as illustrated in Figure 21:

```
+-- ... -+++++ ... +-+-+--+ ... -+++++-----+--+ ... -+++++-----...
|11110001| SRH-6LoRH | NH=1      | 11110CPP | Compressed | UDP
|Page 1  | Type1 S=3 | LOWPAN_IPHC| LOWPAN-NHC| UDP header | Payload
+-- ... -+++++ ... +-+-+--+ ... -+++++-----+--+ ... -+++++-----...
                               <-          RFC 6282          ->
```

Figure 21: compressed SRH 4*2bytes entries sourced by root.

Note: the RPI is not represented though RPL [RFC6550] generally expects it. In this particular case, since the Compression Reference for the SRH-6LoRH is the source address in the LOWPAN_IPHC, and the routing is strict along the source route path, the RPI does not appear to be absolutely necessary.

In Figure 21, all the nodes along the source route path share a same /112 prefix. This is typical of IPv6 addresses derived from an IEEE802.15.4 short address, as long as all the nodes share a same PAN-ID. In that case, a type-1 SRH-6LoRH header can be used for encoding. The IPv6 address of the root is taken as reference, and only the last 2 octets of the address of the intermediate hops is

encoded. The Size of 3 indicates 4 hops, resulting in a SRH-6LoRH of 10 bytes.

A.3. Example of SRH-6LoRH life-cycle

This section illustrates the operation specified in Section 5.6 of forwarding a packet with a compressed SRH along an A->B->C->D source route path. The operation of popping addresses is exemplified at each hop.

Packet as received by node A

```
-----
Type 3 SRH-6LoRH Size = 0   AAAA AAAA AAAA AAAA
Type 1 SRH-6LoRH Size = 0                                     BBBB
Type 2 SRH-6LoRH Size = 1                                     CCCC CCCC
                                                                DDDD DDDD
```

Step 1 popping BBBB the first entry of the next SRH-6LoRH

Step 2 next is if larger value (2 vs. 1) the SRH-6LoRH is removed

```
Type 3 SRH-6LoRH Size = 0   AAAA AAAA AAAA AAAA
Type 2 SRH-6LoRH Size = 1                                     CCCC CCCC
                                                                DDDD DDDD
```

Step 3: recursion ended, coalescing BBBB with the first entry

```
Type 3 SRH-6LoRH Size = 0   AAAA AAAA AAAA BBBB
```

Step 4: routing based on next segment endpoint to B

Figure 22: Processing at Node A.

Packet as received by node B

```
-----
Type 3 SRH-6LoRH Size = 0   AAAA AAAA AAAA BBBB
Type 2 SRH-6LoRH Size = 1               CCCC CCCC
                                      DDDD DDDD
```

Step 1 popping CCCC CCCC, the first entry of the next SRH-6LoRH
 Step 2 removing the first entry and decrementing the Size (by 1)

```
Type 3 SRH-6LoRH Size = 0   AAAA AAAA AAAA BBBB
Type 2 SRH-6LoRH Size = 0               DDDD DDDD
```

Step 3: recursion ended, coalescing CCCC CCCC with the first entry
 Type 3 SRH-6LoRH Size = 0 AAAA AAAA CCCC CCCC

Step 4: routing based on next segment endpoint to C

Figure 23: Processing at Node B.

Packet as received by node C

```
-----
Type 3 SRH-6LoRH Size = 0   AAAA AAAA CCCC CCCC
Type 2 SRH-6LoRH Size = 0               DDDD DDDD
```

Step 1 popping DDDD DDDD, the first entry of the next SRH-6LoRH
 Step 2 the SRH-6LoRH is removed

```
Type 3 SRH-6LoRH Size = 0   AAAA AAAA CCCC CCCC
```

Step 3: recursion ended, coalescing DDDD DDDDD with the first entry
 Type 3 SRH-6LoRH Size = 0 AAAA AAAA DDDD DDDD

Step 4: routing based on next segment endpoint to D

Figure 24: Processing at Node C.

Packet as received by node D

Type 3 SRH-6LoRH Size = 0 AAAA AAAA DDDD DDDD

Step 1 the SRH-6LoRH is removed.

Step 2 no more header, routing based on inner IP header.

Figure 25: Processing at Node D.

Authors' Addresses

Pascal Thubert (editor)
Cisco Systems
Building D - Regus
45 Allee des Ormes
BP1200
MOUGINS - Sophia Antipolis 06254
France

Phone: +33 4 97 23 26 34
Email: pthubert@cisco.com

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Laurent Toutain
Institut MINES TELECOM; TELECOM Bretagne
2 rue de la Chataigneraie
CS 17607
Cesson-Sevigne Cedex 35576
France

Email: Laurent.Toutain@telecom-bretagne.eu

Robert Cragie
ARM Ltd.
110 Fulbourn Road
Cambridge CB1 9NJ
UK

Email: robert.cragie@gridmerge.com

ROLL Working Group
Internet-Draft
Updates: 6553, 6550, 8138 (if approved)
Intended status: Standards Track
Expires: July 19, 2021

M. Robles
UTN-FRM/Aalto
M. Richardson
SSW
P. Thubert
Cisco
January 15, 2021

Using RPI Option Type, Routing Header for Source Routes and IPv6-in-IPv6
encapsulation in the RPL Data Plane
draft-ietf-roll-useofrplinfo-44

Abstract

This document looks at different data flows through LLN (Low-Power and Lossy Networks) where RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) is used to establish routing. The document enumerates the cases where RFC6553 (RPI Option Type), RFC6554 (Routing Header for Source Routes) and IPv6-in-IPv6 encapsulation is required in data plane. This analysis provides the basis on which to design efficient compression of these headers. This document updates RFC6553 adding a change to the RPI Option Type. Additionally, this document updates RFC6550 defining a flag in the DIO Configuration option to indicate about this change and updates RFC8138 as well to consider the new Option Type when the RPL Option is decompressed.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 19, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Overview	4
2. Terminology and Requirements Language	5
3. RPL Overview	6
4. Updates to RFC6550, RFC6553 and RFC8138	7
4.1. Updates to RFC6550	7
4.1.1. Advertising External Routes with Non-Storing Mode Signaling.	7
4.1.2. Configuration Options and Mode of Operation	8
4.1.3. Indicating the new RPI in the DODAG Configuration option Flag.	9
4.2. Updates to RFC6553: Indicating the new RPI Option Type. .	10
4.3. Updates to RFC8138: Indicating the way to decompress with the new RPI Option Type.	13
5. Sample/reference topology	14
6. Use cases	16
7. Storing mode	19
7.1. Storing Mode: Interaction between Leaf and Root	20
7.1.1. SM: Example of Flow from RAL to Root	21
7.1.2. SM: Example of Flow from Root to RAL	22
7.1.3. SM: Example of Flow from Root to RUL	22
7.1.4. SM: Example of Flow from RUL to Root	24
7.2. SM: Interaction between Leaf and Internet.	25
7.2.1. SM: Example of Flow from RAL to Internet	25
7.2.2. SM: Example of Flow from Internet to RAL	27
7.2.3. SM: Example of Flow from RUL to Internet	28
7.2.4. SM: Example of Flow from Internet to RUL.	29
7.3. SM: Interaction between Leaf and Leaf	30
7.3.1. SM: Example of Flow from RAL to RAL	30
7.3.2. SM: Example of Flow from RAL to RUL	31

7.3.3.	SM: Example of Flow from RUL to RAL	33
7.3.4.	SM: Example of Flow from RUL to RUL	34
8.	Non Storing mode	35
8.1.	Non-Storing Mode: Interaction between Leaf and Root . . .	37
8.1.1.	Non-SM: Example of Flow from RAL to root	37
8.1.2.	Non-SM: Example of Flow from root to RAL	38
8.1.3.	Non-SM: Example of Flow from root to RUL	39
8.1.4.	Non-SM: Example of Flow from RUL to root	40
8.2.	Non-Storing Mode: Interaction between Leaf and Internet .	41
8.2.1.	Non-SM: Example of Flow from RAL to Internet	41
8.2.2.	Non-SM: Example of Flow from Internet to RAL	43
8.2.3.	Non-SM: Example of Flow from RUL to Internet	44
8.2.4.	Non-SM: Example of Flow from Internet to RUL	45
8.3.	Non-SM: Interaction between leaves	46
8.3.1.	Non-SM: Example of Flow from RAL to RAL	46
8.3.2.	Non-SM: Example of Flow from RAL to RUL	49
8.3.3.	Non-SM: Example of Flow from RUL to RAL	51
8.3.4.	Non-SM: Example of Flow from RUL to RUL	52
9.	Operational Considerations of supporting RUL-leaves	53
10.	Operational considerations of introducing 0x23	54
11.	IANA Considerations	54
11.1.	Option Type in RPL Option	54
11.2.	Change to the DODAG Configuration Options Flags registry	55
11.3.	Change MOP value 7 to Reserved	55
12.	Security Considerations	56
13.	Acknowledgments	59
14.	References	59
14.1.	Normative References	60
14.2.	Informative References	61
	Authors' Addresses	63

1. Introduction

RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [RFC6550] is a routing protocol for constrained networks. [RFC6553] defines the RPL Option carried within the IPv6 Hop-by-Hop Header to carry the RPLInstanceID and quickly identify inconsistencies (loops) in the routing topology. The RPL Option is commonly referred to as the RPL Packet Information (RPI) though the RPI is the routing information that is defined in [RFC6550] and transported in the RPL Option. RFC6554 [RFC6554] defines the "RPL Source Route Header" (RH3), an IPv6 Extension Header to deliver datagrams within a RPL routing domain, particularly in non-storing mode.

These various items are referred to as RPL artifacts, and they are seen on all of the data-plane traffic that occurs in RPL routed networks; they do not in general appear on the RPL control plane

traffic at all which is mostly Hop-by-Hop traffic (one exception being DAO messages in non-storing mode).

It has become clear from attempts to do multi-vendor interoperability, and from a desire to compress as many of the above artifacts as possible that not all implementers agree when artifacts are necessary, or when they can be safely omitted, or removed.

The ROLL WG analyzed how [RFC2460] rules apply to storing and non-storing use of RPL. The result was 24 data plane use cases. They are exhaustively outlined here in order to be completely unambiguous. During the processing of this document, new rules were published as [RFC8200], and this document was updated to reflect the normative changes in that document.

This document updates [RFC6553], changing the value of the Option Type of the RPL Option to make [RFC8200] routers ignore this option when not recognized.

A Routing Header Dispatch for 6LoWPAN (6LoRH) ([RFC8138]) defines a mechanism for compressing RPL Option information and Routing Header type 3 (RH3) [RFC6554], as well as an efficient IPv6-in-IPv6 technique.

Most of the use cases described herein require the use of IPv6-in-IPv6 packet encapsulation. When encapsulating and decapsulating packets, [RFC6040] MUST be applied to map the setting of the explicit congestion notification (ECN) field between inner and outer headers. Additionally, [I-D.ietf-intarea-tunnels] is recommended reading to explain the relationship of IP tunnels to existing protocol layers and the challenges in supporting IP tunneling.

Non-constrained uses of RPL are not in scope of this document, and applicability statements for those uses may provide different advice, E.g. [I-D.ietf-anima-autonomic-control-plane].

1.1. Overview

The rest of the document is organized as follows: Section 2 describes the used terminology. Section 3 provides a RPL Overview. Section 4 describes the updates to RFC6553, RFC6550 and RFC 8138. Section 5 provides the reference topology used for the uses cases. Section 6 describes the use cases included. Section 7 describes the storing mode cases and section 8 the non-storing mode cases. Section 9 describes the operational considerations of supporting RPL-unaware-leaves. Section 10 depicts operational considerations for the proposed change on RPL Option Type, section 11 the IANA considerations and then section 12 describes the security aspects.

2. Terminology and Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Terminology defined in [RFC7102] applies to this document: LLN, RPL, RPL domain and ROLL.

Consumed: A Routing Header is consumed when the Segments Left field is zero, which indicates that the destination in the IPv6 header is the final destination of the packet and that the hops in the Routing Header have been traversed.

RPL Leaf: An IPv6 host that is attached to a RPL router and obtains connectivity through a RPL Destination Oriented Directed Acyclic Graph (DODAG). As an IPv6 node, a RPL Leaf is expected to ignore a consumed Routing Header and as an IPv6 host, it is expected to ignore a Hop-by-Hop header. It results that a RPL Leaf can correctly receive a packet with RPL artifacts. On the other hand, a RPL Leaf is not expected to generate RPL artifacts or to support IP-in-IP encapsulation. For simplification, this document uses the standalone term leaf to mean a RPL leaf.

RPL Packet Information (RPI): The information defined abstractly in [RFC6550] to be placed in IP packets. The term is commonly used, including in this document, to refer to the RPL Option [RFC6553] that transports that abstract information in an IPv6 Hop-by-Hop Header. [RFC8138] provides an alternate (more compressed) formatting for the same abstract information.

RPL-aware-node (RAN): A device which implements RPL. Please note that the device can be found inside the LLN or outside LLN.

RPL-Aware-Leaf(RAL): A RPL-aware-node that is also a RPL Leaf.

RPL-unaware-node: A device which does not implement RPL, thus the device is not-RPL-aware. Please note that the device can be found inside the LLN.

RPL-Unaware-Leaf(RUL): A RPL-unaware-node that is also a RPL Leaf.

6LoWPAN Node (6LN): [RFC6775] defines it as: "A 6LoWPAN node is any host or router participating in a LoWPAN. This term is used when referring to situations in which either a host or router can play the role described.". In this document, a 6LN acts as a leaf.

6LoWPAN Router (6LR): [RFC6775] defines it as:" An intermediate router in the LoWPAN that is able to send and receive Router Advertisements (RAs) and Router Solicitations (RSs) as well as forward and route IPv6 packets. 6LoWPAN routers are present only in route-over topologies."

6LoWPAN Border Router (6LBR): [RFC6775] defines it as:"A border router located at the junction of separate 6LoWPAN networks or between a 6LoWPAN network and another IP network. There may be one or more 6LBRs at the 6LoWPAN network boundary. A 6LBR is the responsible authority for IPv6 prefix propagation for the 6LoWPAN network it is serving. An isolated LoWPAN also contains a 6LBR in the network, which provides the prefix(es) for the isolated network."

Flag Day: A Flag Day is caused when a network is reconfigured in a way that nodes running the older configuration can not communicate with nodes running the new configuration. For instance, when the ARPANET changed from IP version 3 to IP version 4 on January 1, 1983 ([RFC0801]). In the context of this document, a switch from RPI Option Type (0x63) and Option Type (0x23) presents as a disruptive changeover. In order to reduce the amount of time for such a changeover, Section 4.1.3 provides a mechanism to allow nodes to be incrementally upgraded.

Non-Storing Mode (Non-SM): RPL mode of operation in which the RPL-aware-nodes send information to the root about their parents. Thus, the root knows the topology. Because the root knows the topology, the intermediate 6LRs do not maintain routing state and source routing is needed.

Storing Mode (SM): RPL mode of operation in which RPL-aware-nodes (6LRs) maintain routing state (of the children) so that source routing is not needed.

Note: Due to lack of space in some figures (tables) we refer to IPv6-in-IPv6 as IP6-IP6.

3. RPL Overview

RPL defines the RPL Control messages (control plane), a new ICMPv6 [RFC4443] message with Type 155. DIS (DODAG Information Solicitation), DIO (DODAG Information Object) and DAO (Destination Advertisement Object) messages are all RPL Control messages but with different Code values. A RPL Stack is shown in Figure 1.

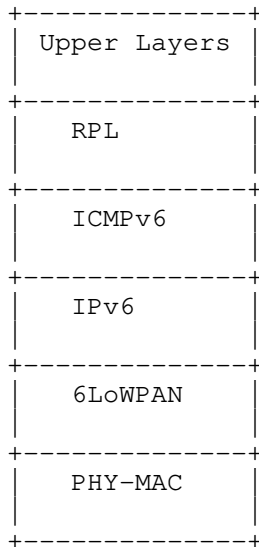


Figure 1: RPL Stack.

RPL supports two modes of Downward internal traffic: in storing mode (SM), it is fully stateful; in non-storing mode (Non-SM), it is fully source routed. A RPL Instance is either fully storing or fully non-storing, i.e. a RPL Instance with a combination of a fully storing and non-storing nodes is not supported with the current specifications at the time of writing this document. External routes are advertised with non-storing-mode messaging even in a storing mode network, see Section 4.1.1

4. Updates to RFC6550, RFC6553 and RFC8138

4.1. Updates to RFC6550

4.1.1. Advertising External Routes with Non-Storing Mode Signaling.

Section 6.7.8. of [RFC6550] introduces the 'E' flag that is set to indicate that the 6LR that generates the DAO redistributes external targets into the RPL network. An external Target is a Target that has been learned through an alternate protocol, for instance a route to a prefix that is outside the RPL domain but reachable via a 6LR. Being outside of the RPL domain, a node that is reached via an external target cannot be guaranteed to ignore the RPL artifacts and cannot be expected to process the [RFC8138] compression correctly. This means that the RPL artifacts should be contained in an IP-in-IP encapsulation that is removed by the 6LR, and that any remaining

compression should be expanded by the 6LR before it forwards a packet outside the RPL domain.

This specification updates [RFC6550] to RECOMMEND that external targets are advertised using Non-Storing Mode DAO messaging even in a Storing-Mode network. This way, external routes are not advertised within the DODAG and all packets to an external target reach the Root like normal Non-Storing Mode traffic. The Non-Storing Mode DAO informs the Root of the address of the 6LR that injects the external route, and the root uses IP-in-IP encapsulation to that 6LR, which terminates the IP-in-IP tunnel and forwards the original packet outside the RPL domain free of RPL artifacts.

In the other direction, for traffic coming from an external target into the LLN, the parent (6LR) that injects the traffic always encapsulates to the root. This whole operation is transparent to intermediate routers that only see traffic between the 6LR and the Root, and only the Root and the 6LRs that inject external routes in the network need to be upgraded to add this function to the network.

A RUL is a special case of external target when the target is actually a host and it is known to support a consumed Routing Header and to ignore a Hop-by-Hop header as prescribed by [RFC8200]. The target may have been learned through an external routing protocol or may have been registered to the 6LR using [RFC8505].

In order to enable IP-in-IP all the way to a 6LN, it is beneficial that the 6LN supports decapsulating IP-in-IP, but that is not assumed by [RFC8504]. If the 6LN is a RUL, the Root that encapsulates a packet SHOULD terminate the tunnel at a parent 6LR unless it is aware that the RUL supports IP-in-IP decapsulation.

A node that is reachable over an external route is not expected to support [RFC8138]. Whether a decapsulation took place or not and even when the 6LR is delivering the packet to a RUL, the 6LR that injected an external route MUST uncompress the packet before forwarding over that external route.

4.1.2. Configuration Options and Mode of Operation

Section 6.7.6 of RFC6550 describes the DODAG Configuration Option as containing a series of Flags in the first octet of the payload.

Anticipating future work to revise RPL relating to how the LLN and DODAG are configured, this document renames the DODAG Configuration Option Flags registry so that it applies to Mode of Operation (MOP) values zero (0) to six (6) only, leaving the flags unassigned for MOP value seven (7). The MOP is described in RFC6550 section 6.3.1.

In addition, this document reserves MOP value 7 for future expansion.

See Sections 11.2 and 11.3.

4.1.3. Indicating the new RPI in the DODAG Configuration option Flag.

In order to avoid a Flag Day caused by lack of interoperation between new RPI Option Type (0x23) and old RPI Option Type (0x63) nodes, this section defines a flag in the DIO Configuration option, to indicate when the new RPI Option Type can be safely used. This means, the flag is going to indicate the value of Option Type that the network will be using for the RPL Option. Thus, when a node joins to a network it will know which value to use. With this, RPL-capable nodes know if it is safe to use 0x23 when creating a new RPL Option. A node that forwards a packet with an RPI MUST NOT modify the Option Type of the RPL Option.

This is done using a DODAG Configuration option flag which will signal "RPI 0x23 enable" and propagate through the network. Section 6.3.1. of [RFC6550] defines a 3-bit Mode of Operation (MOP) in the DIO Base Object. The flag is defined only for MOP value between 0 to 6.

For a MOP value of 7, a node MUST use the RPI 0x23 option.

As stated in [RFC6550] the DODAG Configuration option is present in DIO messages. The DODAG Configuration option distributes configuration information. It is generally static, and does not change within the DODAG. This information is configured at the DODAG root and distributed throughout the DODAG with the DODAG Configuration option. Nodes other than the DODAG root do not modify this information when propagating the DODAG Configuration option.

Currently, the DODAG Configuration option in [RFC6550] states: "the unused bits MUST be initialized to zero by the sender and MUST be ignored by the receiver". If the flag is received with a value zero (which is the default), then new nodes will remain in RFC6553 Compatible Mode; originating traffic with the old-RPI Option Type (0x63) value. If the flag is received with a value of 1, then the value for the RPL Option MUST be set to 0x23.

Bit number three of the flag field in the DODAG Configuration option is to be used as shown in Figure 2 (which is the same as Figure 39 in Section 11 and is shown here for convenience):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 2: DODAG Configuration option Flag to indicate the RPI-flag-day.

In the case of reboot, the node (6LN or 6LR) does not remember the RPI Option Type (i.e., whether or not the flag is set), so the node will not trigger DIO messages until a DIO message is received indicating the RPI value to be used. The node will use the value 0x23 if the network supports this feature.

4.2. Updates to RFC6553: Indicating the new RPI Option Type.

This modification is required in order to be able to send, for example, IPv6 packets from a RPL-Aware-Leaf to a RPL-unaware node through Internet (see Section 7.2.1), without requiring IPv6-in-IPv6 encapsulation.

[RFC6553] (Section 6, Page 7) states as shown in Figure 3, that in the Option Type field of the RPL Option, the two high order bits must be set to '01' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node must discard the packet if it doesn't recognize the Option Type, and the third bit indicates that the Option Data may change in route. The remaining bits serve as the Option Type.

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x63	01	1	00011	RPL Option	[RFC6553]

Figure 3: Option Type in RPL Option.

This document illustrates that it is not always possible to know for sure at the source that a packet will only travel within the RPL domain or may leave it.

At the time [RFC6553] was published, leaking a Hop-by-Hop header in the outer IPv6 header chain could potentially impact core routers in the internet. So at that time, it was decided to encapsulate any

packet with a RPL Option using IPv6-in-IPv6 in all cases where it was unclear whether the packet would remain within the RPL domain. In the exception case where a packet would still leak, the Option Type would ensure that the first router in the Internet that does not recognize the option would drop the packet and protect the rest of the network.

Even with [RFC8138], where the IPv6-in-IPv6 header is compressed, this approach yields extra bytes in a packet; this means consuming more energy, more bandwidth, incurring higher chances of loss and possibly causing a fragmentation at the 6LoWPAN level. This impacts the daily operation of constrained devices for a case that generally does not happen and would not heavily impact the core anyway.

While intention was and remains that the Hop-by-Hop header with a RPL Option should be confined within the RPL domain, this specification modifies this behavior in order to reduce the dependency on IPv6-in-IPv6 and protect the constrained devices. Section 4 of [RFC8200] clarifies the behaviour of routers in the Internet as follows: "it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so".

When unclear about the travel of a packet, it becomes preferable for a source not to encapsulate, accepting the fact that the packet may leave the RPL domain on its way to its destination. In that event, the packet should reach its destination and should not be discarded by the first node that does not recognize the RPL Option. But with the current value of the Option Type, if a node in the Internet is configured to process the Hop-by-Hop header, and if such node encounters an option with the first two bits set to 01 and conforms to [RFC8200], it will drop the packet. Host systems should do the same, irrespective of the configuration.

Thus, this document updates the Option Type of the RPL Option [RFC6553], naming it RPI Option Type for simplicity, to (Figure 4): the two high order bits MUST be set to '00' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node MUST skip over this option and continue processing the header ([RFC8200] Section 4.2) if it doesn't recognize the Option Type, and the third bit continues to be set to indicate that the Option Data may change en route. The rightmost five bits remain at 0x3(00011). This ensures that a packet that leaves the RPL domain of an LLN (or that leaves the LLN entirely) will not be discarded when it contains the RPL Option.

With the new Option Type, if an IPv6 (intermediate) node (RPL-not-capable) receives a packet with a RPL Option, it should ignore the

Hop-by-Hop RPL Option (skip over this option and continue processing the header). This is relevant, as it was mentioned previously, in the case that there is a flow from RAL to Internet (see Section 7.2.1).

This is a significant update to [RFC6553].

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x23	00	1	00011	RPL Option	[RFCXXXX] (*)

Figure 4: Revised Option Type in RPL Option. (*) represents this document

Without the signaling described below, this change would otherwise create a lack of interoperation (flag day) for existing networks which are currently using 0x63 as the RPI Option Type value. A move to 0x23 will not be understood by those networks. It is suggested that RPL implementations accept both 0x63 and 0x23 when processing the header.

When forwarding packets, implementations SHOULD use the same value of RPI Type as was received. This is required because the RPI Option Type does not change en route ([RFC8200] - Section 4.2). It allows the network to be incrementally upgraded and allows the DODAG root to know which parts of the network have been upgraded.

When originating new packets, implementations should have an option to determine which value to originate with, this option is controlled by the DIO Configuration option (Section Section 4.1.3).

The change of RPI Option Type from 0x63 to 0x23, makes all [RFC8200] Section 4.2 compliant nodes tolerant of the RPL artifacts. There is no longer a need to remove the artifacts when sending traffic to the Internet. This change clarifies when to use IPv6-in-IPv6 headers, and how to address them: The Hop-by-Hop Options header containing the RPI MUST always be added when 6LRs originate packets (without IPv6-in-IPv6 headers), and IPv6-in-IPv6 headers MUST always be added when a 6LR finds that it needs to insert a Hop-by-Hop Options header containing the RPL Option. The IPv6-in-IPv6 header is to be addressed to the RPL root when on the way up, and to the end-host when on the way down.

In the non-storing case, dealing with not-RPL aware leaf nodes is much easier as the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node.

The 6LBR can recognize not-RPL aware leaf nodes because it will receive a DAO about that node from the 6LR immediately above that not-RPL aware node.

The non-storing mode case does not require the type change from 0x63 to 0x23, as the root can always create the right packet. The type change does not adversely affect the non-storing case. (see Section 4.1.3)

4.3. Updates to RFC8138: Indicating the way to decompress with the new RPI Option Type.

This modification is required in order to be able to decompress the RPL Option with the new Option Type of 0x23.

RPI-6LoRH header provides a compressed form for the RPL RPI; see [RFC8138], Section 6. A node that is decompressing this header MUST decompress using the RPI Option Type that is currently active: that is, a choice between 0x23 (new) and 0x63 (old). The node will know which to use based upon the presence of the flag in the DODAG Configuration option defined in Section 4.1.3. E.g. If the network is in 0x23 mode (by DIO option), then it should be decompressed to 0x23.

[RFC8138] section 7 documents how to compress the IPv6-in-IPv6 header.

There are potential significant advantages to having a single code path that always processes IPv6-in-IPv6 headers with no conditional branches.

In Storing Mode, the scenarios where the flow goes from RAL to RUL and RUL to RUL include compression of the IPv6-in-IPv6 and RPI headers. The use of the IPv6-in-IPv6 header is MANDATORY in this case, and it SHOULD be compressed with [RFC8138] section 7. Figure 5 illustrates the case in Storing mode where the packet is received from the Internet, then the root encapsulates the packet to insert the RPI. In that example, the leaf is not known to support RFC 8138, and the packet is encapsulated to the 6LR that is the parent and last hop to the final destination.

```

+-+ ... +-+ ... +-+ ... +-+ +-+ +-+ +-+ ... +-+ +-+ ... -+++ ... +-...
|11110001|SRH-6LoRH| RPI- |IP-in-IP| NH=1 |11110001| UDP | UDP
|Page 1 |Type1 S=0| 6LoRH |6LoRH | LOWPAN_IPHC | UDP | hdr | Payld
+-+ ... +-+ ... +-+ ... +-+ +-+ +-+ +-+ ... +-+ +-+ ... -+ ... +-...
                <-4bytes->                <- RFC 6282 ->
                                           No RPL artifact

```

Figure 5: RPI Inserted by the Root in Storing Mode

In Figure 5, the source of the IPv6-in-IPv6 encapsulation is the Root, so it is elided in the IP-in-IP 6LoRH. The destination is the parent 6LR of the destination of the inner packet so it cannot be elided. It is placed as the single entry in an SRH-6LoRH as the first 6LoRH. There is a single entry so the SRH-6LoRH Size is 0. In that example, the type is 1 so the 6LR address is compressed to 2 bytes. It results that the total length of the SRH-6LoRH is 4 bytes. Follows the RPI-6LoRH and then the IP-in-IP 6LoRH. When the IP-in-IP 6LoRH is removed, all the router headers that precede it are also removed. The Paging Dispatch [RFC8025] may also be removed if there was no previous Page change to a Page other than 0 or 1, since the LOWPAN_IPHC is encoded in the same fashion in the default Page 0 and in Page 1. The resulting packet to the destination is the inner packet compressed with [RFC6282].

5. Sample/reference topology

A RPL network in general is composed of a 6LBR, a Backbone Router (6BBR), a 6LR and a 6LN as a leaf logically organized in a DODAG structure.

Figure 6 shows the reference RPL Topology for this document. The letters above the nodes are there so that they may be referenced in subsequent sections. In the figure, 6LR represents a full router node. The 6LN is a RPL aware router, or host (as a leaf). Additionally, for simplification purposes, it is supposed that the 6LBR has direct access to Internet and is the root of the DODAG, thus the 6BBR is not present in the figure.

The 6LN leaves (RAL) marked as (F, H and I) are RPL nodes with no children hosts.

The leaves marked as RUL (G and J) are devices that do not speak RPL at all (not-RPL-aware), but use Router-Advertisements, 6LowPAN DAR/DAC and 6LoWPAN ND only to participate in the network [RFC8505]. In the document these leaves (G and J) are also referred to as a RUL.

The 6LBR ("A") in the figure is the root of the Global DODAG.

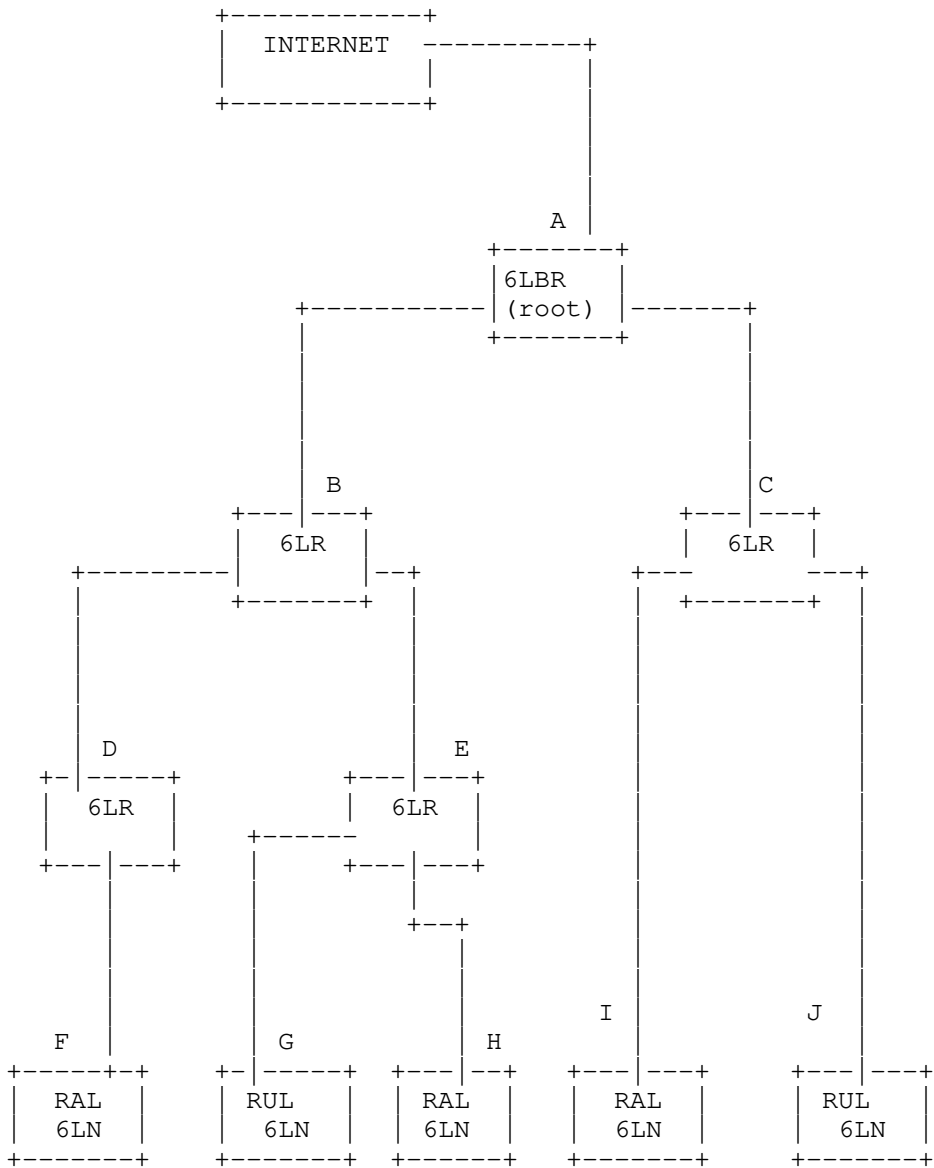


Figure 6: A reference RPL Topology.

6. Use cases

In the data plane a combination of RFC6553, RFC6554 and IPv6-in-IPv6 encapsulation are going to be analyzed for a number of representative traffic flows.

The use cases describe the communication in the following cases: - Between RPL-aware-nodes with the root (6LBR) - Between RPL-aware-nodes with the Internet - Between RUL nodes within the LLN (e.g. see Section 7.1.4) - Inside of the LLN when the final destination address resides outside of the LLN (e.g. see Section 7.2.3).

The use cases are as follows:

Interaction between Leaf and Root:

RAL to root

root to RAL

RUL to root

root to RUL

Interaction between Leaf and Internet:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

Interaction between leaves:

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

This document is consistent with the rule that a Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed. This is a fundamental precept of the IPv6 architecture as outlined in [RFC8200].

As the rank information in the RPI artifact is changed at each hop, it will typically be zero when it arrives at the DODAG root. The DODAG root MUST force it to zero when passing the packet out to the Internet. The Internet will therefore not see any SenderRank information.

Despite being legal to leave the RPI artifact in place, an intermediate router that needs to add an extension header (e.g. RH3 or RPL Option) MUST still encapsulate the packet in an (additional) outer IP header. The new header is placed after this new outer IP header.

A corollary is that an intermediate router can remove an RH3 or RPL Option only if it is placed in an encapsulating IPv6 Header that is addressed TO this intermediate router. When doing the above, the whole encapsulating header must be removed. (A replacement may be added). This sometimes can result in outer IP headers being addressed to the next hop router using link-local address.

Both the RPL Option and the RH3 headers may be modified in very specific ways by routers on the path of the packet without the need to add and remove an encapsulating header. Both headers were designed with this modification in mind, and both the RPL RH3 and the RPL Option are marked mutable but recoverable: so an IPsec AH security header can be applied across these headers, but it can not secure the values which mutate.

The RPI MUST be present in every single RPL data packet.

Prior to [RFC8138], there was significant interest in creating an exception to this rule and removing the RPI for downward flows in non-storing mode. This exception covered a very small number of cases, and caused significant interoperability challenges while adding significant interest in the code and tests. The ability to compress the RPI down to three bytes or less removes much of the pressure to optimize this any further [I-D.ietf-anima-autonomic-control-plane].

Throughout the following subsections, the examples are described in more details in the first subsections, and more concisely in the later ones.

The uses cases are delineated based on the following IPV6 and RPL mandates:

The RPI has to be in every packet that traverses the LLN.

- Because of the above requirement, packets from the Internet have to be encapsulated.
- A Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed.
- Extension headers may not be added or removed except by the sender or the receiver.
- RPI and RH3 headers may be modified by routers on the path of the packet without the need to add and remove an encapsulating header.
- an RH3 or RPL Option can only be removed by an intermediate router if it is placed in an encapsulating IPv6 Header, which is addressed to the intermediate router.
- Non-storing mode requires downstream encapsulation by root for RH3.

The uses cases are delineated based on the following assumptions:

This document assumes that the LLN is using the no-drop RPI Option Type (0x23).

- Each IPv6 node (including Internet routers) obeys [RFC8200], so that 0x23 RPI Option Type can be safely inserted.
- All 6LRs obey [RFC8200].
- The RPI is ignored at the IPv6 dst node (RUL).
- In the uses cases, we assume that the RAL supports IP-in-IP encapsulation.
- In the uses cases, we don't assume that the RUL supports IP-in-IP encapsulation.
- For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags.
- The description for RALs applies to RAN in general.
- Non-constrained uses of RPL are not in scope of this document.
- Compression is based on [RFC8138].

- The flow label [RFC6437] is not needed in RPL.

7. Storing mode

In storing mode (SM) (fully stateful), the sender can determine if the destination is inside the LLN by looking if the destination address is matched by the DIO's Prefix Information Option (PIO) option.

The following table (Figure 7) itemizes which headers are needed in each of the following scenarios. It indicates whether an IPv6-in-IPv6 header must be added and what destination it must be addressed to: (1) the final destination (the RAL node that is the target (tgt)), (2) the "root", or (3) the 6LR parent of a RUL.

In cases where no IPv6-in-IPv6 header is needed, the column states "No", and the destination is N/A (Not Applicable). If the IPv6-in-IPv6 header is needed, the column shows "must".

In all cases, the RPI is needed, since it identifies inconsistencies (loops) in the routing topology. In general, the RH3 is not needed because it is not used in storing mode. However, there is one scenario (from the root to the RUL in SM) where the RH3 can be used to point at the RUL (Figure 11).

The leaf can be a router 6LR or a host, both indicated as 6LN. The root refers to the 6LBR (see Figure 6).

Interaction between	Use Case	IPv6-in-IPv6	IPv6-in-IPv6 dst
Leaf - Root	RAL to root	No	N/A
	root to RAL	No	N/A
	root to RUL	must	6LR
	RUL to root	must	root
Leaf - Internet	RAL to Int	may	root
	Int to RAL	must	RAL (tgt)
	RUL to Int	must	root
	Int to RUL	must	6LR
Leaf - Leaf	RAL to RAL	No	N/A
	RAL to RUL	No (up)	N/A
		must (down)	6LR
	RUL to RAL	must (up)	root
		must (down)	RAL
	RUL to RUL	must (up)	root
		must (down)	6LR

Figure 7: Table of IPv6-in-IPv6 encapsulation in Storing mode.

7.1. Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in storing mode (SM) between,

RAL to root

root to RAL

RUL to root

root to RUL

7.1.1. SM: Example of Flow from RAL to Root

In storing mode, RFC 6553 (RPI) is used to send RPL Information instanceID and rank information.

In this case the flow comprises:

RAL (6LN) --> 6LR_i --> root(6LBR)

For example, a communication flow could be: Node F (6LN) --> Node D (6LR_i) --> Node B (6LR_i) --> Node A root(6LBR)

The RAL (Node F) inserts the RPI, and sends the packet to 6LR (Node D) which decrements the rank in the RPI and sends the packet up. When the packet arrives at 6LBR (Node A), the RPI is removed and the packet is processed.

No IPv6-in-IPv6 header is required.

The RPI can be removed by the 6LBR because the packet is addressed to the 6LBR. The RAL must know that it is communicating with the 6LBR to make use of this scenario. The RAL can know the address of the 6LBR because it knows the address of the root via the DODAGID in the DIO messages.

The Figure 8 summarizes what headers are needed for this use case.

Header	RAL src	6LR_i	6LBR dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 8: SM: Summary of the use of headers from RAL to root

7.1.2. SM: Example of Flow from Root to RAL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RAL (6LN)

For example, a communication flow could be: Node A root(6LBR) --> Node B (6LR_i) --> Node D (6LR_i) --> Node F (6LN)

In this case the 6LBR inserts RPI and sends the packet down, the 6LR is going to increment the rank in RPI (it examines the RPLInstanceID to identify the right forwarding table), the packet is processed in the RAL and the RPI removed.

No IPv6-in-IPv6 header is required.

The Figure 9 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	RAL dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 9: SM: Summary of the use of headers from root to RAL

7.1.3. SM: Example of Flow from Root to RUL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Node A (6LBR) --> Node B (6LR_i) --> Node E (6LR_n) --> Node G (RUL)

6LR_i (Node B) represents the intermediate routers from the source (6LBR) to the destination (RUL), $1 \leq i \leq n$, where n is the total

number of routers (6LR) that the packet goes through from the 6LBR (Node A) to the RUL (Node G).

The 6LBR will encapsulate the packet in an IPv6-in-IPv6 header, and prepend an RPI. The IPv6-in-IPv6 header is addressed to the 6LR parent of the RUL (6LR_n). The 6LR parent of the RUL removes the header and sends the packet to the RUL.

The Figure 10 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	6LR_n	RUL dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 10: SM: Summary of the use of headers from root to RUL

IP-in-IP encapsulation may be avoided for Root to RUL communication. In SM, it can be replaced by a loose RH3 header that indicates the RUL, in which case the packet is routed to the 6LR as a normal SM operation, then the 6LR forwards to the RUL based on the RH3, and the RUL ignores both the consumed RH3 and the RPI, as in Non-Storing Mode.

The Figure 11 summarizes what headers are needed for this scenario.

Header	6LBR src	6LR _i <i>i</i> =(1,...,n-1)	6LR _n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI	RPI RH3 (consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	RH3	--	RPI, RH3 (both ignored)

Figure 11: SM: Summary of the use of headers from root to RUL without encapsulation

7.1.4. SM: Example of Flow from RUL to Root

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR)

For example, a communication flow could be: Node G (RUL) --> Node E (6LR₁) --> Node B (6LR_i) --> Node A root (6LBR)

6LR_i represents the intermediate routers from the source (RUL) to the destination (6LBR), $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RUL to the 6LBR.

When the packet arrives from the RUL (Node G) to 6LR₁ (Node E), the 6LR₁ will encapsulate the packet in an IPv6-in-IPv6 header with an RPI. The IPv6-in-IPv6 header is addressed to the root (Node A). The root removes the header and processes the packet.

The Figure 12 shows the table that summarizes what headers are needed for this use case where the IPv6-in-IPv6 header is addressed to the root (Node A).

Header	RUL src node	6LR_1	6LR_i	6LBR dst
Added headers	--	IP6-IP6 RPI	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	---	IP6-IP6 RPI
Untouched headers	--	--	IP6-IP6	--

Figure 12: SM: Summary of the use of headers from RUL to root.

7.2. SM: Interaction between Leaf and Internet.

In this section is described the communication flow in storing mode (SM) between,

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

7.2.1. SM: Example of Flow from RAL to Internet

In this case the flow comprises:

RAL (6LN) --> 6LR_i --> root (6LBR) --> Internet

For example, the communication flow could be: Node F (RAL) --> Node D (6LR_i) --> Node B (6LR_i) --> Node A root(6LBR) --> Internet

6LR_i represents the intermediate routers from the source (RAL) to the root (6LBR), $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RAL to the 6LBR.

RPL information from RFC 6553 may go out to Internet as it will be ignored by nodes which have not been configured to be RPI aware. No IPv6-in-IPv6 header is required.

On the other hand, the RAL may insert the RPI encapsulated in a IPv6-in-IPv6 header to the root. Thus, the root removes the RPI and send the packet to the Internet.

Note: In this use case, it is used a node as a leaf, but this use case can be also applicable to any RPL-aware-node type (e.g. 6LR)

The Figure 13 summarizes what headers are needed for this use case when there is no encapsulation. Note that the RPI is modified by 6LBR to set the SenderRank to zero in case that it is not already zero. The Figure 14 summarizes what headers are needed when encapsulation to the root takes place.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

Figure 13: SM: Summary of the use of headers from RAL to Internet with no encapsulation

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 14: SM: Summary of the use of headers from RAL to Internet with encapsulation to the root (6LBR).

7.2.2. SM: Example of Flow from Internet to RAL

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RAL (6LN)

For example, a communication flow could be: Internet --> Node A
root(6LBR) --> Node B (6LR_1) --> Node D (6LR_n) --> Node F (RAL)

When the packet arrives from Internet to 6LBR the RPI is added in a outer IPv6-in-IPv6 header (with the IPv6-in-IPv6 destination address set to the RAL) and sent to 6LR, which modifies the rank in the RPI. When the packet arrives at the RAL, the packet is decapsulated, which removes the RPI before the packet is processed.

The Figure 15 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	RAL dst
Added headers	--	IP6-IP6 (RPI)	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	--	IP6-IP6 (RPI)
Untouched headers	--	--	--	--

Figure 15: SM: Summary of the use of headers from Internet to RAL.

7.2.3. SM: Example of Flow from RUL to Internet

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_i --> root (6LBR) --> Internet

For example, a communication flow could be: Node G (RUL) --> Node E (6LR_1) --> Node B (6LR_i) --> Node A root (6LBR) --> Internet

The node 6LR_1 (i=1) will add an IPv6-in-IPv6(RPI) header addressed to the root such that the root can remove the RPI before passing upwards. In the intermediate 6LR, the rank in the RPI is modified.

The originating node will ideally leave the IPv6 flow label as zero so that the packet can be better compressed through the LLN. The 6LBR will set the flow label of the packet to a non-zero value when sending to the Internet, for details check [RFC6437].

The Figure 16 shows the table that summarizes what headers are needed for this use case.

Header	IPv6 src node (RUL)	6LR_1	6LR_i [i=2,...,n]	6LBR	Internet dst
Added headers	--	IP6-IP6 (RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6 (RPI)	--
Untouched headers	--	--	--	--	--

Figure 16: SM: Summary of the use of headers from RUL to Internet.

7.2.4. SM: Example of Flow from Internet to RUL.

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Internet --> Node A
root(6LBR) --> Node B (6LR_i) --> Node E (6LR_n) --> Node G (RUL)

The 6LBR will have to add an RPI within an IPv6-in-IPv6 header. The IPv6-in-IPv6 is addressed to the 6LR parent of the RUL.

Further details about this are mentioned in [I-D.ietf-roll-unaware-leaves], which specifies RPL routing for a 6LN acting as a plain host and not being aware of RPL.

The 6LBR may set the flow label on the inner IPv6-in-IPv6 header to zero in order to aid in compression [RFC8138][RFC6437].

The Figure 17 shows the table that summarizes what headers are needed for this use case.

Header	Inter- net src	6LBR	6LR_i [i=1,...,n-1]	6LR_n	RUL dst
Inserted headers	--	IP6-IP6 (RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6 (RPI)	--
Untouched headers	--	--	--	--	--

Figure 17: SM: Summary of the use of headers from Internet to RUL.

7.3. SM: Interaction between Leaf and Leaf

In this section is described the communication flow in storing mode (SM) between,

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

7.3.1. SM: Example of Flow from RAL to RAL

In [RFC6550] RPL allows a simple one-hop optimization for both storing and non-storing networks. A node may send a packet destined to a one-hop neighbor directly to that node. See section 9 in [RFC6550].

When the nodes are not directly connected, then in storing mode, the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> RAL dst (6LN)

For example, a communication flow could be: Node F (RAL src) --> Node D (6LR_ia) --> Node B (6LR_x) --> Node E (6LR_id) --> Node H (RAL dst)

6LR_ia (Node D) represents the intermediate routers from source to the common parent (6LR_x) (Node B), $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL (Node F) to the common parent 6LR_x (Node B).

6LR_id (Node E) represents the intermediate routers from the common parent (6LR_x) (Node B) to destination RAL (Node H), $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination RAL (Node H).

It is assumed that the two nodes are in the same RPL domain (that they share the same DODAG root). At the common parent (Node B), the direction flag ('O' flag) of the RPI is changed (from decreasing ranks to increasing ranks).

While the 6LR nodes will update the RPI, no node needs to add or remove the RPI, so no IPv6-in-IPv6 headers are necessary.

The Figure 18 summarizes what headers are needed for this use case.

Header	RAL src	6LR_ia	6LR_x (common parent)	6LR_id	RAL dst
Added headers	RPI	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Removed headers	--	--	--	--	RPI
Untouched headers	--	--	--	--	--

Figure 18: SM: Summary of the Use of Headers from RAL to RAL

7.3.2. SM: Example of Flow from RAL to RUL

In this case the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LBR - The root-) -->
6LR_id --> RUL (IPv6 dst node)

For example, a communication flow could be: Node F (RAL) --> Node D
--> Node B --> Node A --> Node B --> Node E --> Node G (RUL)

6LR_ia represents the intermediate routers from source (RAL) to the common parent (the Root), $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL to the Root.

6LR_id (Node E) represents the intermediate routers from the Root (Node B) to destination RUL (Node G). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the Root down to the destination RUL.

In this case, the packet from the RAL goes to 6LBR because the route to the RUL is not injected into the RPL-SM. Thus, the RAL inserts an RPI (RPI1) addressed to the root (6LBR). The root does not remove the RPI1 (the root cannot remove an RPI if there is no encapsulation). The root inserts an IPv6-IPv6 encapsulation with an RPI2 and sends it to the 6LR parent of the RUL, which removes the encapsulation and RPI2 before passing the packet to the RUL.

The Figure 19 summarizes what headers are needed for this use case.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	RPI1	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 19: SM: Summary of the Use of Headers from RAL to RUL

7.3.3. SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_ia --> 6LBR --> 6LR_id --> RAL dst (6LN)

For example, a communication flow could be: Node G (RUL) --> Node E --> Node B --> Node A --> Node B --> Node D --> Node F (RAL)

6LR_ia (Node E) represents the intermediate routers from source (RUL) (Node G) to the root (Node A). In this case, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from source to the root.

6LR_id represents the intermediate routers from the root (Node A) to destination RAL (Node F). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the root to the destination RAL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI1) encapsulated in a IPv6-in-IPv6 header to the root. The root removes the outer header including the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the destination RAL (Node F).

The Figure 20 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--	--
Modified headers	--	--	RPI1	--	RPI2	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)
Untouched headers	--	--	--	--	--	--

Figure 20: SM: Summary of the use of headers from RUL to RAL.

7.3.4. SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node)--> 6LR_1--> 6LR_ia --> 6LBR --> 6LR_id --> RUL
(IPv6 dst node)

For example, a communication flow could be: Node G (RUL src)--> Node E --> Node B --> Node A (root) --> Node C --> Node J (RUL dst)

Internal nodes 6LR_ia (e.g: Node E or Node B) is the intermediate router from the RUL source (Node G) to the root (6LBR) (Node A). In this case, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RUL to the root. 6LR_1 refers when $ia=1$.

6LR_id (Node C) represents the intermediate routers from the root (Node A) to the destination RUL dst node (Node J). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the root to destination RUL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI), encapsulated in an IPv6-in-IPv6 header directed to the root. The root removes the outer header including

the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the 6LR father of the RUL.

The Figure 21 shows the table that summarizes what headers are needed for this use case.

Header	RUL src	6LR_1	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst
Added Headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 21: SM: Summary of the use of headers from RUL to RUL

8. Non Storing mode

In Non Storing Mode (Non-SM) (fully source routed), the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node. Thus, there is no need for all nodes to know about the existence of RPL-unaware nodes. Only the 6LBR needs to act if compensation is necessary for not-RPL aware receivers.

The table (Figure 22) summarizes what headers are needed in the following scenarios, and indicates when the RPI, RH3 and IPv6-in-IPv6 header are to be inserted. The last column depicts the target destination of the IPv6-in-IPv6 header: 6LN (indicated by "RAL"), 6LR (parent of a RUL) or the root. In cases where no IPv6-in-IPv6 header is needed, the column indicates "No". There is no expectation on RPL that RPI can be omitted, because it is needed for routing, quality of service and compression. This specification expects that an RPI is always present. The term "may(up)" means that the IPv6-in-IPv6 header may be necessary in the upwards direction. The term "must(up)" means that the IPv6-in-IPv6 header must be present in the upwards direction. The term "must(down)" means that the IPv6-in-IPv6 header must be present in the downward direction.

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 6). In the table (Figure 22) the (1) indicates a 6tisch case [RFC8180], where the RPI may still be needed for the RPLInstanceID to be available for priority/channel selection at each hop.

Interaction between	Use Case	RPI	RH3	IPv6-in-IPv6	IP-in-IP dst
Leaf - Root	RAL to root	Yes	No	No	No
	root to RAL	Yes	Yes	No	No
	root to RUL	Yes (1)	Yes	No	6LR
	RUL to root	Yes	No	must	root
Leaf - Internet	RAL to Int	Yes	No	may (up)	root
	Int to RAL	Yes	Yes	must	RAL
	RUL to Int	Yes	No	must	root
	Int to RUL	Yes	Yes	must	6LR
Leaf - Leaf	RAL to RAL	Yes	Yes	may (up)	root
				must (down)	RAL
	RAL to RUL	Yes	Yes	may (up)	root
				must (down)	6LR
	RUL to RAL	Yes	Yes	must (up)	root
				must (down)	RAL
	RUL to RUL	Yes	Yes	must (up)	root
				must (down)	6LR

Figure 22: Table that shows headers needed in Non-Storing mode: RPI, RH3, IPv6-in-IPv6 encapsulation.

8.1. Non-Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RAL to root

root to RAL

RUL to root

root to RUL

8.1.1. Non-SM: Example of Flow from RAL to root

In non-storing mode the leaf node uses default routing to send traffic to the root. The RPI must be included since it contains the rank information, which is used to avoid/detect loops.

RAL (6LN) --> 6LR_i --> root(6LBR)

For example, a communication flow could be: Node F --> Node D --> Node B --> Node A (root)

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RAL) to destination (6LBR).

This situation is the same case as storing mode.

The Figure 23 summarizes what headers are needed for this use case.

Header	RAL src	6LR_i	6LBR dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 23: Non-SM: Summary of the use of headers from RAL to root

8.1.2. Non-SM: Example of Flow from root to RAL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RAL (6LN)

For example, a communication flow could be: Node A (root) --> Node B --> Node D --> Node F

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (6LBR) to destination (RAL).

The 6LBR inserts an RH3, and an RPI. No IPv6-in-IPv6 header is necessary as the traffic originates with a RPL aware node, the 6LBR. The destination is known to be RPL-aware because the root knows the whole topology in non-storing mode.

The Figure 24 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	RAL dst
Added headers	RPI, RH3	--	--
Modified headers	--	RPI, RH3	--
Removed headers	--	--	RPI, RH3
Untouched headers	--	--	--

Figure 24: Non-SM: Summary of the use of headers from root to RAL

8.1.3. Non-SM: Example of Flow from root to RUL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Node A (root) --> Node B --> Node E --> Node G (RUL)

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (6LBR) to destination (RUL).

In the 6LBR, the RH3 is added; it is then modified at each intermediate 6LR (6LR_1 and so on), and it is fully consumed in the last 6LR (6LR_n) but is left in place. When the RPI is added, the RUL, which does not understand the RPI, will ignore it (per [RFC8200]); thus, encapsulation is not necessary.

The Figure 25 depicts the table that summarizes what headers are needed for this use case.

Header	6LBR src	6LR _i $i=(1,\dots,n-1)$	6LR _n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI, RH3	RPI, RH3 (consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI, RH3 (both ignored)

Figure 25: Non-SM: Summary of the use of headers from root to RUL

8.1.4. Non-SM: Example of Flow from RUL to root

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR) dst

For example, a communication flow could be: Node G --> Node E --> Node B --> Node A (root)

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RUL) to destination (6LBR). For example, 6LR₁ ($i=1$) is the router that receives the packets from the RUL.

In this case, the RPI is added by the first 6LR (6LR₁) (Node E), encapsulated in an IPv6-in-IPv6 header, and modified in the subsequent 6LRs in the flow. The RPI and the entire packet are consumed by the root.

The Figure 26 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_i	6LBR dst
Added headers	--	IPv6-in-IPv6 (RPI)	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	--	IPv6-in-IPv6 (RPI)
Untouched headers	--	--	--	--

Figure 26: Non-SM: Summary of the use of headers from RUL to root

8.2. Non-Storing Mode: Interaction between Leaf and Internet

This section will describe the communication flow in Non Storing Mode (Non-SM) between:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

8.2.1. Non-SM: Example of Flow from RAL to Internet

In this case the flow comprises:

RAL (6LN) src --> 6LR_i --> root (6LBR) --> Internet dst

For example, a communication flow could be: Node F (RAL) --> Node D --> Node B --> Node A --> Internet. Having the RAL information about the RPL domain, the packet may be encapsulated to the root when the destination is not in the RPL domain of the RAL.

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RAL) to 6LBR.

In this case, the encapsulation from the RAL to the root is optional. The simplest case is when the RPI gets to the Internet (as the Figure 27 shows it), knowing that the Internet is going to ignore it.

The IPv6 flow label should be set to zero to aid in compression [RFC8138], and the 6LBR will set it to a non-zero value when sending towards the Internet [RFC6437].

The Figure 27 summarizes what headers are needed for this use case when no encapsulation is used. The Figure 28 summarizes what headers are needed for this use case when encapsulation to the root is used.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

Figure 27: Non-SM: Summary of the use of headers from RAL to Internet with no encapsulation

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	IPv6-in-IPv6 (RPI)	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI)	--
Untouched headers	--	--	--	--

Figure 28: Non-SM: Summary of the use of headers from RAL to Internet with encapsulation to the root

8.2.2. Non-SM: Example of Flow from Internet to RAL

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RAL dst (6LN)

For example, a communication flow could be: Internet --> Node A (root) --> Node B --> Node D --> Node F (RAL)

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from 6LBR to destination (RAL).

The 6LBR must add an RH3 header. As the 6LBR will know the path and address of the target node, it can address the IPv6-in-IPv6 header to that node. The 6LBR will zero the flow label upon entry in order to aid compression [RFC8138].

The Figure 29 summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR _i	RAL dst
Added headers	--	IPv6-in-IPv6 (RH3, RPI)	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3, RPI)	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3, RPI)
Untouched headers	--	--	--	--

Figure 29: Non-SM: Summary of the use of headers from Internet to RAL

8.2.3. Non-SM: Example of Flow from RUL to Internet

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR) --> Internet
dst

For example, a communication flow could be: Node G --> Node E -->
Node B --> Node A --> Internet

6LR_i represents the intermediate routers from source to destination,
1 ≤ i ≤ n, where n is the total number of routers (6LRs) that the
packet goes through from the source (RUL) to the 6LBR, e.g., 6LR₁
(i=1).

In this case the flow label is recommended to be zero in the RUL. As
the RUL parent adds RPL headers in the RUL packet, the first 6LR
(6LR₁) will add an RPI inside a new IPv6-in-IPv6 header. The IPv6-
in-IPv6 header will be addressed to the root. This case is identical
to the storing-mode case (see Section 7.2.3).

The Figure 30 shows the table that summarizes what headers are needed
for this use case.

Header	RUL src node	6LR_1	6LR_i [i=2,...,n]	6LBR	Internet dst
Added headers	--	IP6-IP6 (RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6 (RPI)	--
Untouched headers	--	--	--	--	--

Figure 30: Non-SM: Summary of the use of headers from RUL to Internet

8.2.4. Non-SM: Example of Flow from Internet to RUL

In this case the flow comprises:

Internet src --> root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Internet --> Node A (root) --> Node B --> Node E --> Node G

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from 6LBR to RUL.

The 6LBR must add an RH3 header inside an IPv6-in-IPv6 header. The 6LBR will know the path, and will recognize that the final node is not a RPL capable node as it will have received the connectivity DAO from the nearest 6LR. The 6LBR can therefore make the IPv6-in-IPv6 header destination be the last 6LR. The 6LBR will set to zero the flow label upon entry in order to aid compression [RFC8138].

The Figure 31 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	6LR_n	RUL dst
Added headers	--	IP6-IP6 (RH3,RPI)	--	--	--
Modified headers	--	--	IP6-IP6 (RH3,RPI)	--	--
Removed headers	--	--	--	IP6-IP6 (RH3,RPI)	--
Untouched headers	--	--	--	--	--

Figure 31: Non-SM: Summary of the use of headers from Internet to RUL.

8.3. Non-SM: Interaction between leaves

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

8.3.1. Non-SM: Example of Flow from RAL to RAL

In this case the flow comprises:

RAL src --> 6LR_ia --> root (6LBR) --> 6LR_id --> RAL dst

For example, a communication flow could be: Node F (RAL src)--> Node D --> Node B --> Node A (root) --> Node B --> Node E --> Node H (RAL dst)

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL to the root.

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

This case involves only nodes in same RPL domain. The originating node will add an RPI to the original packet, and send the packet upwards.

The originating node may put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root, so that the 6LBR can remove that header. If it does not, then the RPI1 is forwarded down from the root in the inner header to no avail.

The 6LBR will need to insert an RH3 header, which requires that it add an IPv6-in-IPv6 header. It removes the RPI(RPI1), as it was contained in an IPv6-in-IPv6 header addressed to it. Otherwise, there may be an RPI buried inside the inner IP header, which should get ignored. The root inserts an RPI (RPI2) alongside the RH3.

Networks that use the RPL P2P extension [RFC6997] are essentially non-storing DODAGs and fall into this scenario or scenario Section 8.1.2, with the originating node acting as 6LBR.

The Figure 32 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place.

The Figure 33 shows the table that summarizes what headers are needed for this use case when there is no encapsulation to the root. Note that in the Modified headers row, going up in each 6LR_id only the RPI1 is changed. Going down, in each 6LR_id the IPv6 header is swapped with the RH3 so both are changed alongside with the RPI2.

Header	RAL src	6LR_ia	6LBR	6LR_id	RAL dst
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3-> RAL, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3,RPI2)	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--

Figure 32: Non-SM: Summary of the Use of Headers from RAL to RAL with encapsulation to the root.

Header	RAL	6LR_ia	6LBR	6LR_id	RAL
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	RPI1	RPI1	RPI1 (Ignored)

Figure 33: Non-SM: Summary of the Use of Headers from RAL to RAL without encapsulation to the root.

8.3.2. Non-SM: Example of Flow from RAL to RUL

In this case the flow comprises:

RAL --> 6LR_ia --> root (6LBR) --> 6LR_id --> RUL (IPv6 dst node)

For example, a communication flow could be: Node F (RAL) --> Node D --> Node B --> Node A (root) --> Node B --> Node E --> Node G (RUL)

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LRs).

As in the previous case, the RAL (6LN) may insert an RPI (RPI1) header which must be in an IPv6-in-IPv6 header addressed to the root so that the 6LBR can remove this RPI. The 6LBR will then insert an RH3 inside a new IPv6-in-IPv6 header addressed to the last 6LR_id ($6LR_id = m$) alongside the insertion of RPI2.

If the originating node does not put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root. Then, the RPI1 is forwarded down from the root in the inner header to no avail.

The Figure 34 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place. The Figure 35 shows the table that summarizes what headers are needed for this use case when no encapsulation to the root takes place.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--

Figure 34: Non-SM: Summary of the use of headers from RAL to RUL with encapsulation to the root.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst node
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 35: Non-SM: Summary of the use of headers from RAL to RUL without encapsulation to the root.

8.3.3. Non-SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id
--> RAL dst (6LN)

For example, a communication flow could be: Node G (RUL) --> Node E
--> Node B --> Node A (root) --> Node B --> Node E --> Node H (RAL)

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

In this scenario the RPI (RPI1) is added by the first 6LR (6LR_1) inside an IPv6-in-IPv6 header addressed to the root. The 6LBR will remove this RPI, and add its own IPv6-in-IPv6 header containing an RH3 header and an RPI (RPI2).

The Figure 36 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--	--

Figure 36: Non-SM: Summary of the use of headers from RUL to RAL.

8.3.4. Non-SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id
--> RUL (IPv6 dst node)

For example, a communication flow could be: Node G --> Node E -->
Node B --> Node A (root) --> Node C --> Node J

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

This scenario is the combination of the previous two cases.

The Figure 37 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 37: Non-SM: Summary of the use of headers from RUL to RUL

9. Operational Considerations of supporting RUL-leaves

Roughly half of the situations described in this document involve leaf ("host") nodes that do not speak RPL. These nodes fall into two further categories: ones that drop a packet that have RPI or RH3 headers, and ones that continue to process a packet that has RPI and/or RH3 headers.

[RFC8200] provides for new rules that suggest that nodes that have not been configured (explicitly) to examine Hop-by-Hop headers, should ignore those headers, and continue processing the packet. Despite this, and despite the switch from 0x63 to 0x23, there may be nodes that are pre-RFC8200, or simply intolerant. Those nodes will drop packets that continue to have RPL artifacts in them. In general, such nodes can not be easily supported in RPL LLNs.

There are some specific cases where it is possible to remove the RPL artifacts prior to forwarding the packet to the leaf host. The critical thing is that the artifacts have been inserted by the RPL root inside an IPv6-in-IPv6 header, and that the header has been addressed to the 6LR immediately prior to the leaf node. In that case, in the process of removing the IPv6-in-IPv6 header, the artifacts can also be removed.

The above case occurs whenever traffic originates from the outside the LLN (the "Internet" cases above), and non-storing mode is used. In non-storing mode, the RPL root knows the exact topology (as it must create the RH3 header) and therefore knows which 6LR is prior to the leaf. For example, in Figure 6, Node E is the 6LR prior to leaf Node G, or Node C is the 6LR prior to leaf Node J.

Traffic originating from the RPL root (such as when the data collection system is co-located on the RPL root), does not require an IPv6-in-IPv6 header (in storing or non-storing mode), as the packet is originating at the root, and the root can insert the RPI and RH3 headers directly into the packet, as it is formed. Such a packet is slightly smaller, but only can be sent to nodes (whether RPL aware or not), that will tolerate the RPL artifacts.

An operator that finds itself with a high amount of traffic from the RPL root to RPL-not-aware-leaves, will have to do IPv6-in-IPv6 encapsulation if the leaf is not tolerant of the RPL artifacts. Such an operator could otherwise omit this unnecessary header if it was certain of the properties of the leaf.

As storing mode can not know the final path of the traffic, intolerant (that drop packets with RPL artifacts) leaf nodes can not be supported.

10. Operational considerations of introducing 0x23

This section describes the operational considerations of introducing the new RPI Option Type of 0x23.

During bootstrapping the node gets the DIO with the information of RPI Option Type, indicating the new RPI in the DODAG Configuration option Flag. The DODAG root is in charge to configure the current network to the new value, through DIO messages and when all the nodes are set with the new value. The DODAG should change to a new DODAG version. In case of rebooting, the node does not remember the RPI Option Type. Thus, the DIO is sent with a flag indicating the new RPI Option Type.

The DODAG Configuration option is contained in a RPL DIO message, which contains a unique DTSN counter. The leaf nodes respond to this message with DAO messages containing the same DTSN. This is a normal part of RPL routing; the RPL root therefore knows when the updated DODAG Configuration option has been seen by all nodes.

Before the migration happens, all the RPL-aware nodes should support both values . The migration procedure is triggered when the DIO is sent with the flag indicating the new RPI Option Type. Namely, it remains at 0x63 until it is sure that the network is capable of 0x23, then it abruptly changes to 0x23. The 0x23 RPI Option allows to send packets to not-RPL nodes. The not-RPL nodes should ignore the option and continue processing the packets.

As mentioned previously, indicating the new RPI in the DODAG Configuration option flag is a way to avoid the flag day (abrupt changeover) in a network using 0x63 as the RPI Option Type value. It is suggested that RPL implementations accept both 0x63 and 0x23 RPI Option type values when processing the header to enable interoperability.

11. IANA Considerations

11.1. Option Type in RPL Option

This document updates the registration made in [RFC6553] Destination Options and Hop-by-Hop Options registry from 0x63 to 0x23 as shown in Figure 38.

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x23	00	1	00011	RPL Option	[RFCXXXX] (*)
0x63	01	1	00011	RPL Option (DEPRECATED)	[RFC6553] [RFCXXXX] (*)

Figure 38: Option Type in RPL Option. (*) represents this document
DODAG Configuration option is updated as follows (Figure 39):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 39: DODAG Configuration option Flag to indicate the RPI-flag-day.

11.2. Change to the DODAG Configuration Options Flags registry

This document requests IANA to change the name of the "DODAG Configuration Option Flags" registry to "DODAG Configuration Option Flags for MOP 0..6".

This document requests to be mentioned as a reference for this change.

11.3. Change MOP value 7 to Reserved

This document requests the changing the registration status of value 7 in the Mode of Operation registry from Unassigned to Reserved. This change is in support of future work.

This document requests to be mentioned as a reference for this entry in the registry.

12. Security Considerations

The security considerations covered in [RFC6553] and [RFC6554] apply when the packets are in the RPL Domain.

The IPv6-in-IPv6 mechanism described in this document is much more limited than the general mechanism described in [RFC2473]. The willingness of each node in the LLN to decapsulate packets and forward them could be exploited by nodes to disguise the origin of an attack.

While a typical LLN may be a very poor origin for attack traffic (as the networks tend to be very slow, and the nodes often have very low duty cycles), given enough nodes, LLNs could still have a significant impact, particularly if the attack is targeting another LLN. Additionally, some uses of RPL involve large backbone ISP scale equipment [I-D.ietf-anima-autonomic-control-plane], which may be equipped with multiple 100Gb/s interfaces.

Blocking or careful filtering of IPv6-in-IPv6 traffic entering the LLN as described above will make sure that any attack that is mounted must originate from compromised nodes within the LLN. The use of BCP38 [BCP38] filtering at the RPL root on egress traffic will both alert the operator to the existence of the attack, as well as drop the attack traffic. As the RPL network is typically numbered from a single prefix, which is itself assigned by RPL, BCP38 filtering involves a single prefix comparison and should be trivial to automatically configure.

There are some scenarios where IPv6-in-IPv6 traffic should be allowed to pass through the RPL root, such as the IPv6-in-IPv6 mediated communications between a new Pledge and the Join Registrar/Coordinator (JRC) when using [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-6tisch-dtsecurity-zerotouch-join]. This is the case for the RPL root to do careful filtering: it occurs only when the Join Coordinator is not co-located inside the RPL root.

With the above precautions, an attack using IPv6-in-IPv6 tunnels can only be by a node within the LLN on another node within the LLN. Such an attack could, of course, be done directly. An attack of this kind is meaningful only if the source addresses are either fake or if the point is to amplify return traffic. Such an attack, could also be done without the use of IPv6-in-IPv6 headers using forged source addresses. If the attack requires bi-directional communication, then IPv6-in-IPv6 provides no advantages.

Whenever IPv6-in-IPv6 headers are being proposed, there is a concern about creating security issues. In the Security Considerations

section of [RFC2473], it was suggested that tunnel entry and exit points can be secured by securing the IPv6 path between them. This recommendation is not practical for RPL networks. [RFC5406] goes into some detail on what additional details would be needed in order to "Use IPsec". Use of ESP would prevent [RFC8138] compression (compression must occur before encryption), and [RFC8138] compression is lossy in a way that prevents use of AH. These are minor issues. The major issue is how to establish trust enough such that IKEv2 could be used. This would require a system of certificates to be present in every single node, including any Internet nodes that might need to communicate with the LLN. Thus, using IPsec requires a global PKI in the general case.

More significantly, the use of IPsec tunnels to protect the IPv6-in-IPv6 headers would in the general case scale with the square of the number of nodes. This is a lot of resource for a constrained nodes on a constrained network. In the end, the IPsec tunnels would be providing only BCP38-like origin authentication! That is, IPsec provides a transitive guarantee to the tunnel exit point that the tunnel entry point did BCP38 on traffic going in. Just doing origin filtering per BCP 38 at the entry and exit of the LLN provides a similar level of security without all the scaling and trust problems related to IPv6 tunnels as discussed in RFC 2473. IPsec is not recommended.

An LLN with hostile nodes within it would not be protected against impersonation with the LLN by entry/exit filtering.

The RH3 header usage described here can be abused in equivalent ways. An external attacker may form a packet with an RH3 that is not fully consumed and encapsulate it to hide the RH3 from intermediate nodes and disguise the origin of traffic. As such, the attacker's RH3 header will not be seen by the network until it reaches the destination, which will decapsulate it. As indicated in section 4.2 of [RFC6554], RPL routers are responsible for ensuring that an SRH is only used between RPL routers. As such, if there is an RH3 that is not fully consumed in the encapsulated packet, the node that decapsulates it MUST ensure that the outer packet was originated in the RPL domain and drop the packet otherwise.

Also, as indicated by section 2 of [RFC6554], RPL Border Routers "do not allow datagrams carrying an SRH header to enter or exit a RPL routing domain". This sentence must be understood as concerning non-fully-consumed packets. A consumed (inert) RH3 header could be present in a packet that flows from one LLN, crosses the Internet, and enters another LLN. As per the discussion in this document, such headers do not need to be removed. However, there is no case described in this document where an RH3 is inserted in a non-storing

network on traffic that is leaving the LLN, but this document should not preclude such a future innovation.

In short, a packet that crosses the border of the RPL domain MAY carry and RH3, and if so, that RH3 MUST be fully consumed.

The RPI, if permitted to enter the LLN, could be used by an attacker to change the priority of a packet by selecting a different RPLInstanceID, perhaps one with a higher energy cost, for instance. It could also be that not all nodes are reachable in an LLN using the default RPLInstanceID, but a change of RPLInstanceID would permit an attacker to bypass such filtering. Like the RH3, an RPI is to be inserted by the RPL root on traffic entering the LLN by first inserting an IPv6-in-IPv6 header. The attacker's RPI therefore will not be seen by the network. Upon reaching the destination node the RPI has no further meaning and is just skipped; the presence of a second RPI will have no meaning to the end node as the packet has already been identified as being at it's final destination.

For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags. This is done in order to avoid: 1) The leaf is an external router that passes a packet that it did not generate and that carries an unrelated RPI and 2) The leaf is an attacker or presents misconfiguration and tries to inject traffic in a protected instance. Also, this applies in the case where the leaf is aware of the RPL instance and passes a correct RPI; the 6LR needs a configuration that allows that leaf to inject in that instance.

The RH3 and RPIs could be abused by an attacker inside of the network to route packets on non-obvious ways, perhaps eluding observation. This usage appears consistent with a normal operation of [RFC6997] and can not be restricted at all. This is a feature, not a bug.

[RFC7416] deals with many other threats to LLNs not directly related to the use of IPv6-in-IPv6 headers, and this document does not change that analysis.

Nodes within the LLN can use the IPv6-in-IPv6 mechanism to mount an attack on another part of the LLN, while disguising the origin of the attack. The mechanism can even be abused to make it appear that the attack is coming from outside the LLN, and unless countered, this could be used to mount a Distributed Denial Of Service attack upon nodes elsewhere in the Internet. See [DDOS-KREBS] for an example of such attacks already seen in the real world.

If an attack comes from inside of LLN, it can be alleviated with SAVI (Source Address Validation Improvement) using [RFC8505] with

[I-D.ietf-6lo-ap-nd]. The attacker will not be able to source traffic with an address that is not registered, and the registration process checks for topological correctness. Notice that there is an L2 authentication in most of the cases. If an attack comes from outside LLN IPv6-in- IPv6 can be used to hide inner routing headers, but by construction, the RH3 can typically only address nodes within the LLN. That is, an RH3 with a CmprI less than 8 , should be considered an attack (see RFC6554, section 3).

Nodes outside of the LLN will need to pass IPv6-in-IPv6 traffic through the RPL root to perform this attack. To counter, the RPL root SHOULD either restrict ingress of IPv6-in-IPv6 packets (the simpler solution), or it SHOULD walk the IP header extension chain until it can inspect the upper-layer-payload as described in [RFC7045]. In particular, the RPL root SHOULD do [BCP38] processing on the source addresses of all IP headers that it examines in both directions.

Note: there are some situations where a prefix will spread across multiple LLNs via mechanisms such as the one described in [I-D.ietf-6lo-backbone-router]. In this case the BCP38 filtering needs to take this into account, either by exchanging detailed routing information on each LLN, or by moving the BCP38 filtering further towards the Internet, so that the details of the multiple LLNs do not matter.

13. Acknowledgments

This work is done thanks to the grant given by the StandICT.eu project.

A special BIG thanks to C. M. Heard for the help with the Section 4. Much of the redaction in that section is based on his comments.

Additionally, the authors would like to acknowledge the review, feedback, and comments of (alphabetical order): Dominique Barthel, Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, Rahul Jadhav, Benjamin Kaduk, Matthias Kovatsch, Gustavo Mercado, Subramanian Moonesamy, Marcela Orbiscay, Charlie Perkins, Cristian Perez, Alvaro Retana, Peter van der Stok, Xavier Vilajosana, Eric Vyncke and Thomas Watteyne.

14. References

14.1. Normative References

- [BCP38] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/bcp38>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<https://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/info/rfc7045>>.

- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<https://www.rfc-editor.org/info/rfc8025>>.
- [RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <<https://www.rfc-editor.org/info/rfc8138>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

14.2. Informative References

- [DDOS-KREBS]
Goodin, D., "Record-breaking DDoS reportedly delivered by >145k hacked cameras", September 2016, <<http://arstechnica.com/security/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/>>.
- [I-D.ietf-6lo-ap-nd]
Thubert, P., Sarikaya, B., Sethi, M., and R. Struik, "Address Protected Neighbor Discovery for Low-power and Lossy Networks", draft-ietf-6lo-ap-nd-23 (work in progress), April 2020.
- [I-D.ietf-6lo-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-20 (work in progress), March 2020.
- [I-D.ietf-6tisch-dtsecurity-zerotouch-join]
Richardson, M., "6tisch Zero-Touch Secure Join protocol", draft-ietf-6tisch-dtsecurity-zerotouch-join-04 (work in progress), July 2019.
- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-30 (work in progress), October 2020.

- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M.,
and K. Watsen, "Bootstrapping Remote Secure Key
Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
keyinfra-45 (work in progress), November 2020.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet
Architecture", draft-ietf-intarea-tunnels-10 (work in
progress), September 2019.
- [I-D.ietf-roll-unaware-leaves]
Thubert, P. and M. Richardson, "Routing for RPL Leaves",
draft-ietf-roll-unaware-leaves-29 (work in progress),
January 2021.
- [RFC0801] Postel, J., "NCP/TCP transition plan", RFC 801,
DOI 10.17487/RFC0801, November 1981,
<<https://www.rfc-editor.org/info/rfc801>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
(IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in
IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,
December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet
Control Message Protocol (ICMPv6) for the Internet
Protocol Version 6 (IPv6) Specification", STD 89,
RFC 4443, DOI 10.17487/RFC4443, March 2006,
<<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5406] Bellovin, S., "Guidelines for Specifying the Use of IPsec
Version 2", BCP 146, RFC 5406, DOI 10.17487/RFC5406,
February 2009, <<https://www.rfc-editor.org/info/rfc5406>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
"IPv6 Flow Label Specification", RFC 6437,
DOI 10.17487/RFC6437, November 2011,
<<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C.
Bormann, "Neighbor Discovery Optimization for IPv6 over
Low-Power Wireless Personal Area Networks (6LoWPANs)",
RFC 6775, DOI 10.17487/RFC6775, November 2012,
<<https://www.rfc-editor.org/info/rfc6775>>.

- [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks", RFC 6997, DOI 10.17487/RFC6997, August 2013, <<https://www.rfc-editor.org/info/rfc6997>>.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <<https://www.rfc-editor.org/info/rfc7102>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.
- [RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <<https://www.rfc-editor.org/info/rfc8504>>.
- [RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <<https://www.rfc-editor.org/info/rfc8505>>.

Authors' Addresses

Maria Ines Robles
Universidad Tecno. Nac.(UTN)-FRM, Argentina/ Aalto University Finland

Email: mariainesrobles@gmail.com

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: <http://www.sandelman.ca/mcr/>

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

ROLL WG
INTERNET-DRAFT
Intended Status: Informational
Expires: December 27, 2016

R. Jadhav
R. Sahoo
Z. Cao
Huawei Tech
H. Deng
China Mobile
June 27, 2016

No-Path DAO Problem Statement
draft-jadhav-roll-no-path-dao-ps-01

Abstract

This document describes the problems associated with the use of No-Path DAO messaging in RPL.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Current No-Path DAO messaging	3
1.2. Cases when No-Path DAO may be used	4
1.3. Why No-Path DAO is important?	5
1.4. Terminology	5
2. Problems with current No-Path DAO messaging	5
2.1. Lost NP-DAO due to Link break to the previous parent	5
2.2. Invalidate routes to dependent nodes of the switching node	6
2.3. Route downtime caused by asynchronous operation of NPDAO and DAO	6
3. Requirements for the No-Path DAO Optimization	6
3.1. Req#1: Tolerant to the link failures to the previous parents.	7
3.2. Req#2: Support of removal of entries to the dependent nodes of the switching node.	7
3.3. Req#3: No disruption of downstream reachability to the node while sending NP-DAO.	7
4. Existing Solution	7
4.1. NP-DAO can be generated by the parent node who detects link failure to the child	7
4.2. NP-DAO can be generated once the link is restored to the previous parent.	8
5. Security Considerations	9
6. IANA Considerations	9
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

RPL [RFC6550] specifies a proactive distance-vector based routing scheme. The specification has an optional messaging in the form of DAO messages using which the 6LBR can learn route towards any of the nodes. In storing mode, DAO messages would result in routing entries been created on all intermediate hops from the node's parent all the way towards the 6LBR.

[RFC6550] also allows use of No-Path DAO (NPDAO) messaging to invalidate a routing path and thus releasing of any resources utilized on that path. A No-Path DAO is a DAO message with route lifetime of zero, signaling route invalidation for the given target.

This document studies the problems associated with the current use of No-Path DAO messaging, which creates route inefficiency and inconsistency. This document also discusses the requirements for an optimized No-Path DAO messaging scheme.

1.1. Current No-Path DAO messaging

[RFC6550] introduced No-Path DAO messaging in the storing mode so that the node switching its current parent can inform its parents and ancestors to invalidate the existing route. Subsequently parents or ancestors would release any resources (such as the routing entry) it maintains on behalf of that child node. The No-Path DAO message always traverses the RPL tree in upward direction, originating at the target node itself.

For the rest of this document consider the following topology:

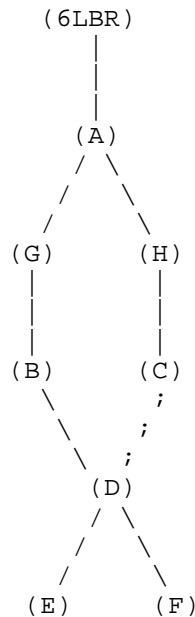


Figure 1: Sample Topology

Node (D) is connected via preferred parent (B). (D) has an alternate path via (C) towards the BR. Node (A) is the common ancestor for (D) for paths through (B)-(G) and (C)-(H). When (D) switches from (B) to (C), [RFC6550] suggests sending No-Path DAO to (B) and regular DAO to (C).

1.2. Cases when No-Path DAO may be used

There are following cases in which a node switches its parent and may employ No-Path DAO messaging:

Case I: Current parent becomes unavailable because of transient or permanent link or parent node failure.

Case II: The node finds a better parent node i.e. the metrics of another parent is better than its current parent.

Case III: The node switches to a new parent whom it "thinks" has a better metric but does not in reality.

The usual steps of operation when the node switches the parent is that the node sends a No-Path DAO message via its current parent to invalidate its current route and subsequently it tries to establish a new routing path by sending a new DAO via its new parent.

1.3. Why No-Path DAO is important?

Nodes in LLNs may be resource constrained. There is limited memory available and routing entry records are the one of the primary elements occupying dynamic memory in the nodes. Route invalidation helps 6LR nodes to decide which entries could be discarded to better achieve resource utilization in case of contention. Thus it becomes necessary to have efficient route invalidation mechanism. Also note that a single parent switch may result in a "sub-tree" switching from one parent to another. Thus the route invalidation needs to be done on behalf of the sub-tree and not the switching node alone. In the above example, when Node (D) switches parent, the route invalidation needs to be done for (D), (E) and (F). Thus without efficient route invalidation, a 6LR may have to hold a lot of unwanted route entries.

1.4. Terminology

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

Common Ancestor node: 6LR node which is the first common node on the old and new path for the child node.

Current parent: Parent 6LR node before switching to the new path

New parent: Parent 6LR node after switching to the new path

NPDAO: No-Path DAO. A DAO message which has target with lifetime 0.

Reverse NPDAO: A No-Path DAO message which traverses downstream in the network.

Regular DAO: A DAO message with non-zero lifetime.

This document also uses terminology described in [RFC6550].

2. Problems with current No-Path DAO messaging

We will confront the following problems when using the current NP-DAO messaging.

2.1. Lost NP-DAO due to Link break to the previous parent

When the node switches its parent, the NPDAO is to be sent via its previous parent and a regular DAO via its new parent. In cases where the node switches its parent because of transient or permanent parent link/node failure then the NPDAO message is bound to fail. [RFC6550]

assumes communication link with the previous parent for No-Path DAO messaging.

[RFC6550] mentions use of route lifetime to remove unwanted routes in case the routes could not be refreshed. But route lifetimes in case of LLNs could be substantially high and thus the route entries would be stuck for long.

2.2. Invalidate routes to dependent nodes of the switching node

No-path DAO is sent by the node who has switched the parent but it does not work for the dependent child nodes below it. The specification does not specify how route invalidation will work for sub-children, resulting in stale routing entries on behalf of the sub-children on the previous route. The only way for 6LR to invalidate the route entries for dependent nodes would be to use route lifetime expiry which could be substantially high for LLNs. In the example topology, when Node (D) switches its parent, Node (D) generates an NPDAO on its behalf. Post switching, Node (D) transmits a DIO with incremented DTSN so that child nodes, node (E) and (F), generate DAOs to trigger route update on the new path for themselves. There is no NPDAO generated by these child nodes through the previous path resulting in stale entries on nodes (B) and (G) for nodes (E) and (F).

2.3. Route downtime caused by asynchronous operation of NPDAO and DAO

A switching node may generate both an NPDAO and DAO via two different paths at almost the same time. There is a possibility that an NPDAO generated may invalidate the previous route and the regular DAO sent via the new path gets lost on the way. This may result in route downtime thus impacting downward traffic for the switching node. In the example topology, consider Node (D) switches from parent (B) to (C) because the metrics of the path via (C) are better. Note that the previous path via (B) may still be available (albeit at relatively bad metrics). An NPDAO sent from previous route may invalidate the existing route whereas there is no way to determine whether the new DAO has successfully updated the route entries on the new path.

An implementation technique to avoid this problem is to further delay the route invalidation by a fixed time interval after receiving an NPDAO, considering the time taken for the new path to be established. Coming up with such a time interval is tricky since the new route may also not be available and it may subsequently require more parent switches to establish a new path.

3. Requirements for the No-Path DAO Optimization

We identify the following requirements for the NP-DAO optimization.

3.1. Req#1: Tolerant to the link failures to the previous parents.

When the switching node send the NP-DAO message to the previous parent, it is normal that the link to the previous parent is prone to failure. Therefore, it is required that the NP-DAO message MUST be tolerant to the link failure during the switching.

3.2. Req#2: Support of removal of entries to the dependent nodes of the switching node.

While switching the parent node and sending NP-DAO message, it is required that the routing entries to the dependent nodes of the switching node will be updated accordingly on the previous parents and other relevant upstream nodes.

3.3. Req#3: No disruption of downstream reachability to the node while sending NP-DAO.

While sending the NP-DAO and DAO messages, it is possible that the NP-DAO successfully invalidates the previous path, while the newly sent DAO gets lost (new path not set up successfully). This will result into downstream unreachability to the current switching node. Therefore, it is desirable that the NP-DAO is synchronized with the DAO to avoid the risk of routing downtime.

4. Existing Solution

4.1. NP-DAO can be generated by the parent node who detects link failure to the child

RPL [RFC6550] states mechanisms which could be utilized to clear DAO states in a sub-DODAG. [RFC6550] Section 11.2.2.3 states "With DAO inconsistency loop recovery, a packet can be used to recursively explore and clean up the obsolete DAO states along a sub-DODAG."

Thus in the sample topology in Figure 1, when Node (B) detects link failure to (D), (B) has an option of generating an NP-DAO on behalf of Node (D) and its sub-children, (E) and (F).

This section explains why generation of an NP-DAO in such cases may not function as desired. Primarily the DAO state information in the form of Path Sequence plays a major role here. Every target is associated with a Path Sequence number which relates to the latest state of the target. [RFC6550] Section 7.1 explains the semantics of Path Sequence number. The target node increments the Path Sequence number every time it generates a new DAO. The router nodes en-route

utilize this Path Sequence number to decide the freshness of target information. If a non-target node has to generate an NP-DAO then it could use following two possibilities with Path Sequence number:

Let the Path Sequence number of old regular DAO that flowed through (B) be x . The subsequent regular DAO generated by Node (D) will have sequence number $x+1$.

i. Node (B) uses the previous Path Sequence number from the regular DAO i.e. NP-DAO(pathseq= x)

ii. Node (B) increments the Path Sequence number i.e. NP-DAO(pathseq= $x+1$)

In case i, the NP-DAO(pathseq= x) will be dropped by all the intermediate nodes since the semantics of Path Sequence number dictates that any DAO with an older Path Sequence number be dropped.

In case ii, there is a risk that the NP-DAO(pathseq= $x+1$) traverses up the DODAG and invalidates all the routes till the root and then the regular DAO(pathseq= $x+1$) from the target traverses upwards. In this case the regular DAO(pathseq= $x+1$) will be dropped from common ancestor node to the root. This will result in route downtime.

Another problem with this scheme is its dependence on the upstream neighbor to detect that the downstream neighbor is unavailable. There are two possibilities by which such a detection might be put to work:

i. There is P2P traffic from the previous sub-DODAG to any of nodes in the sub-tree which has switched the path. In the above example, lets consider that Node (G) has P2P traffic for either of nodes (D), (E), or (F). In this case, Node (B) will detect forwarding error while forwarding the packets from Node (B) to (D). But dependence on P2P traffic may not be an optimal way to solve this problem considering the reactive approach of the scheme. The P2P traffic pattern might be sparse and thus such a detection might kick-in too late.

ii. The other case is where Node (B) explicitly employs some mechanism to probe directly attached downstream child nodes. Such kind of schemes are seldom used.

4.2. NP-DAO can be generated once the link is restored to the previous parent.

This scheme solves a specific scenario of transient links. The child node can detect that the connection to previous parent is restored and then transmit an NP-DAO to the previous parent to invalidate the

route. This scheme is stateful, thus requires more memory and solves a specific scenario.

5. Security Considerations

This draft is a problem statement, and therefore, does not introduce any new security risks.

6. IANA Considerations

Not applicable to this document.

7. Acknowledgements

We would like to thank Cenk Gundogan, Simon Duquennoy and Pascal Thubert for their review and insightful comments.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012.
- [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012.

8.2. Informative References

- [RFC5548] Dohler, M., Ed., Watteyne, T., Ed., Winter, T., Ed., and D. Barthel, Ed., "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

Authors' Addresses

Rahul Arvind Jadhav
Huawei Tech,
Kundalahalli Village,

Bangalore, India

EMail: rahul.jadhav@huawei.com

Rabi Narayan Sahoo
Huawei Tech,
Kundalahalli Village,
Bangalore, India

EMail: rabinarayans@huawei.com

Zhen Cao
Huawei Tech,
Beijing, China

EMail: zhen.cao@huawei.com

Hui Deng
China Mobile,
Beijing, China

EMail: denghui@chinamobile.com

roll
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2017

P. van der Stok, Ed.
consultant
October 25, 2016

A YANG model for Multicast Protocol for Low power and lossy Networks
(MPL)
draft-vanderstok-roll-mpl-yang-02

Abstract

This document defines a YANG data model for management of Multicast Protocol for Low power and lossy Networks (MPL) implementations. The data model includes configuration data and state data.

Note

Discussion and suggestions for improvement are requested, and should be sent to roll@ietf.org.

Status of This Memo

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

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

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

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

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.1.1. Tree Diagrams	3
2. MPL model	3
3. yang-mpl modules	5
3.1. yang-mpl-domain module	5
3.2. yang-mpl-ops module	8
3.3. yang-mpl-seeds module	12
3.4. yang-mpl-statistics module	15
4. IANA Considerations	19
5. Acknowledgements	19
6. Changelog	19
7. References	19
7.1. Normative References	19
7.2. Informative References	20
Author's Address	20

1. Introduction

This document defines a YANG [RFC6020] data model for management of Multicast Protocol for Low power and lossy Networks (MPL) [RFC7731] implementations. The data model covers configuration of per-interface MPL parameters. It also provides information about which Multicast addresses are operationally used, and the seeds for which packets are forwarded

1.1. Terminology

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

The following terms are defined in [RFC6241] and are not redefined here:

- o client
- o configuration data
- o server

- o state data

The following terms are defined in [RFC6020] and are not redefined here:

- o data model
- o data node

The terminology for describing YANG data models is found in [RFC6020].

Terms like domain, seed, I, k, c are defined in [RFC7731].

1.1.1. Tree Diagrams

A simplified graphical representation of the data model is used in the YANG modules specified in this document. The meaning of the symbols in these diagrams is as follows:

Brackets "[" and "]" enclose list keys.

Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).

Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

Ellipsis ("...") stands for contents of subtrees that are not shown.

2. MPL model

This document defines the YANG module "ietf-yang-mpl", which specifies a data model for MPL servers. The model consists of the following parts: (1) a "mpl-domain" part that describes the MPL-domains and associated Multicast addresses and the interfaces on which the Multicast addresses are enabled, (2) a "mpl-op" part that describes the parameters settings per seed, (3) a "mpl-seeds" part that describes the MPL buffer contents and the Trickle timer values, and (4) a "mpl-statistics" part that describes the number of lost and correctly forwarded messages. The data model, divided in four modules, has the following structure for MPL configuration per node:

```
module: ietf-yang-mpl-domain
  +--rw (single)?
    +--:(mpl-domain)
      |   +--rw mpl-domain
      |   |   +--rw domains* [domainID]
      |   |   |   +--rw domainID      uint16
      |   |   |   +--rw MClst*        inet:ipv6-address
      |   |   +--rw addresses* [MCaddress]
      |   |   |   +--rw MCaddress      inet:ipv6-address
      |   |   |   +--rw interfaces*   string
      |   +--:(mpl-single)
      |   |   +--rw mpl-single
      |   |   |   +--rw MCaddresses*   inet:ipv6-address

module: ietf-yang-mpl-ops
  +--rw mpl-ops
    +--rw SE_LIFETIME?          uint16
    +--rw PROACTIVE_FORWARDING? boolean
    +--rw SEED_SET_ENTRY_LIFETIME? uint64
    +--rw mpl-parameter* [domainID]
      +--rw domainID              uint16
      +--rw DATA_MESSAGE_IMIN?   uint16
      +--rw DATA_MESSAGE_IMAX?   uint16
      +--rw DATA_MESSAGE_K?      uint16
      +--rw DATA_MESSAGE_TIMER_EXPIRATIONS? uint16
      +--rw CONTROL_MESSAGE_IMIN? uint16
      +--rw CONTROL_MESSAGE_IMAX? uint16
      +--rw CONTROL_MESSAGE_K?    uint16
      +--rw CONTROL_MESSAGE_TIMER_EXPIRATIONS? uint16

module: ietf-yang-mpl-seeds
  +--ro mpl-seeds* [seedID domainID]
    +--ro seedID          uint64
    +--ro domainID        uint16
    +--ro local?          boolean
    +--ro generate-seqno? uint8
    +--ro life-time?      uint64
    +--ro min-seqno?      uint8
    +--ro data-number?     uint8
    +--ro control-number?  uint8
    +--ro buffered-messages* [seqno]
      +--ro seqno      uint8
      +--ro I?         uint8
      +--ro c?         uint8
      +--ro e?         uint8
      +--ro t?         uint8

module: ietf-yang-mpl-statistics
```

```

+---ro mpl-statistics* [seedID domainID]
  +---ro seedID                               uint64
  +---ro domainID                             uint16
  +---ro c-too-high?                          uint64
  +---ro nr-forwarded?                       uint64
  +---ro nr-of-messages-received?            uint64
  +---ro nr-of-copies-received?              uint64
  +---ro nr-of-messages-forwarded?          uint64
  +---ro nr-of-copies-forwarded?            uint64
  +---ro nr-of-refused?                      uint64
  +---ro nr-of-missed?                       uint64
  +---ro nr-of-notreceived?                  uint64
  +---ro nr-of-inconsistent-data?            uint64
  +---ro nr-of-consistent-data?              uint64
  +---ro nr-of-consistent-control?          uint64
  +---ro nr-of-inconsistent-control?         uint64

```

3. yang-mpl modules

This section describes four yang modules. The model is based on the MPL specification published in [RFC7731] and the specification of [RFC6206]. The identification of the interfaces follows the specification of ietf-interfaces of [RFC7223].

The data model allows to set values to the parameters of the MPL algorithm. This approach requires an active manager process to set the values without use of DHCP as described in: [RFC7774].

The names of the four modules are: yang-mpl-domain, yang-mpl-ops, yang-mpl-seeds, and yang-mpl-statistics, described in subsections with the same name.

3.1. yang-mpl-domain module

This modules describes (1) the MPL domains and the associated multicast addresses, and (2) the interfaces and the multicast addresses for which they are enabled.

The model features a choice such that for constrained devices with only one "single" interface and only one "single" domain, the model specifies a list of MC addresses for which the single interface is enabled.

<CODE BEGINS>file "ietf-yang-mpl-domain@2016-10-25.yang"

```
module ietf-yang-mpl-domain {  
    yang-version 1;  
    namespace  
        "urn:ietf:params:xml:ns:yang:ietf-yang-mpl-domain";  
    prefix mpl;  
    import ietf-inet-types{  
        prefix inet;  
    }  
    organization  
        "IETF ROLL (Routing Over Low power and lossy networks) Working Group";  
    contact  
        "WG Web:   http://tools.ietf.org/wg/roll/  
        WG List:  mailto:roll@ietf.org  
  
        WG Chair: Peter van der Stok  
                  mailto:consultancy@vanderstok.org  
  
        WG Chair: Ines Robles  
                  mailto:maria.ines.robles@ericsson.com  
  
        Editor:   Peter van der Stok  
                  mailto:consultancy@vanderstok.org";  
    description  
        "This module contains information about the state of the MPL domain.  
  
        Copyright (c) 2016 IETF Trust and the persons identified as  
        authors of the code. All rights reserved.  
  
        Redistribution and use in source and binary forms, with or  
        without modification, is permitted pursuant to, and subject  
        to the license terms contained in, the Simplified BSD License  
        set forth in Section 4.c of the IETF Trust's Legal Provisions  
        Relating to IETF Documents  
        (http://trustee.ietf.org/license-info).  
  
        This version of this YANG module is part of RFC XXXX; see  
        the RFC itself for full legal notices.";  
    revision "2016-10-25" {  
        description "Initial revision.";  
        reference
```

```
"I-D:draft-vanderstok-roll-mpl-yang: A YANG model for Multicast Protocol
for Low power and lossy Networks (MPL)";
}

choice single {
  description
    "A choice between single domain/interface and multiple
    domains and interfaces.";
  container mpl-domain {
    description
      "The entries describe the MPL domains, the associated
      Multicast addresses and interfaces.";

    list domains {
      key domainID;
      description
        "The entries describe a given domain identified with domainID and the a
        ssociated Multicast addresses.";

      leaf domainID {
        type uint16;
        description
          "Entry uniquely identifies the domain in the
          forwarder.";
      }

      leaf-list Mclist{
        type inet:ipv6-address;
        description
          "List of associated IPv6 Addresses.";
      }
    } // domains list

    list addresses {
      key MCaddress;
      description
        "The entries describe the interfaces enabled with the specified MC ad
        dress.";

      leaf MCaddress {
        type inet:ipv6-address;
        description
          "MC address belonging to a MPL domain.";
      }

      leaf-list interfaces {
        type string;
        description
          "List of names of interfaces enabled for this Multicast address. In
          terface name is defined in [RFC6206].";
      }
    } // addresses list
  }
}
```

```

    } // container mpl-domain
    container mpl-single {
        description
            "For small devices list of MC addresses for single
            interface and domain.";
        leaf-list MCAddresses{
            type inet:ipv6-address;
            description
                "list of MC addresses belonging to one single domain and interfa
ce.";
        }
    } // container mpl-simple
} // choice simple
} //module ietf-yang-mpl-domain

```

<CODE ENDS>

3.2. yang-mpl-ops module

This module models the operational aspects of MPL. Per domain MPL specifies four parameters I_MAX, I_MIN, K, and TIMER_EXPIRATIONS for data and control messages. The value of the MPL intervals are expressed in TUNIT. The entry SE_LIFETIME taken over from [RFC7774] fixes TUNIT to milliseconds. For very constrained devices with only one domain there can be only one instance of mpl-parameter list.

<CODE BEGINS>file "ietf-yang-mpl-ops@2016-10-25.yang"

```

module ietf-yang-mpl-ops {

    yang-version 1;

    namespace
        "urn:ietf:params:xml:ns:yang:ietf-yang-mpl-ops";

    prefix mpl;

    organization
        "IETF ROLL (Routing over Low power and lossy networks) Working Group";

    contact
        "WG Web:  http://tools.ietf.org/wg/roll/
        WG List:  mailto:roll@ietf.org

        WG Chair: Peter van der Stok
                  mailto:consultancy@vanderstok.org

```

WG Chair: Ines Robles
mailto:maria.ines.robles@ericsson.com

Editor: Peter van der Stok
mailto:consultancy@vanderstok.org";

description

"This module contains information about the operation of the MPL protocol.

Copyright (c) 2016 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or

without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.";

```
revision "2016-10-25" {  
  description "Initial revision.";  
  reference  
    "I-D:draft-vanderstok-roll-mpl-yang: A YANG model for Multicast Protocol  
for Low power and lossy Networks (MPL)";  
}
```

```
container mpl-ops {  
  description  
    "Parameter settings for each MPL server and for each individual domain o  
f the server.";
```

```
  leaf SE_LIFETIME {  
    type uint16;  
    description  
      "lifetime in milliseconds/(mpl timer units),  
      equivalent to SEED_SET_ENTRY_LIFETIME/TUNIT as  
      specified in RFC7774.";  
  }
```

```
  leaf PROACTIVE_FORWARDING {  
    type boolean;  
    description  
      "The boolean value indicates whether the MPL forwarder  
schedules MPL data message transmission after  
receiving them for the first time.";
```

```
}

leaf SEED_SET_ENTRY_LIFETIME {
    type uint64;
    description
        "The value indicates the minimum lifetime for an entry
        in the Seed set expressed in seconds. Default value
        is 30 minutes.";
}

list mpl-parameter{
    key domainID;
    description
        "Each domain has a set of MPL forwarding parameters
        which regulate the forwarding operation.";

    leaf domainID{
        type uint16;
        description
            "Each domainID must be present in  mpl-parameter list.";
    }

    leaf DATA_MESSAGE_IMIN{
        type uint16;
        description
            "The minimum Trickle timer interval, as defined in
            [RFC6206], for MPL Data Message transmissions.";
    }

    leaf DATA_MESSAGE_IMAX{
        type uint16;
        description
            "The maximum Trickle timer interval, as defined in
            [RFC6206], for MPL Data Message transmissions.";
    }

    leaf DATA_MESSAGE_K{
        type uint16;
        default 1;
        description
            "The redundancy constant, as defined in [RFC6206], for
            MPL Data Message transmissions.";
    }

    leaf DATA_MESSAGE_TIMER_EXPIRATIONS{
        type uint16;
        default 3;
        description
```



```
        "The number of Trickle timer expirations that occur
        before terminating the Trickle algorithm's
        retransmission of a given MPL Data Message.";
    }

    leaf CONTROL_MESSAGE_IMIN{
    type uint16;
    description
        "The minimum Trickle timer interval, as defined
        in [RFC6206], for MPL Control Message
        transmissions.";
    }

    leaf CONTROL_MESSAGE_IMAX{
    type uint16;
    description
        "The maximum Trickle timer interval, as defined
        in [RFC6206], for MPL Control Message
        transmissions.";
    }

    leaf CONTROL_MESSAGE_K{
    type uint16;
    default 1;
    description
        "The redundancy constant, as defined in [RFC6206],
        for MPL Control Message transmissions.";
    }

    leaf CONTROL_MESSAGE_TIMER_EXPIRATIONS{
    type uint16;
    default 10;
    description
        "The number of Trickle time expirations that occur
        before terminating the Trickle algorithm
        for MPL Control Message transmissions.";
    }

    } // list MPL-parameter
} // container MPL-ops
} // module ietf-yang-mpl-ops

<CODE ENDS>
```

3.3. yang-mpl-seeds module

This module specifies the current values of the operation of the MPL forwarder. The values are acquired by the client and set by the server. The module specifies a set of message buffers, with a buffer per seed and domain. In constrained devices there will be only one domain, but probably multiple seeds.

The message buffer contains a set of messages where each message is uniquely identified by its sequence number and seed. The associated I, c, e, and t values indicate the progress of MPL with respect to this message, as specified in [RFC7731]. A forwarder sends and receives multiple copies of a message. When a forwarder has sent (received) a copy of a message, the forwarder has sent (received) that message.

For forwarders which are seeds, local has value true and seqno is the sequence number of the next message to send.

```
<CODE BEGINS>file "ietf-yang-mpl-seeds@2016-10-25.yang"

module ietf-yang-mpl-seeds {

  yang-version 1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-mpl-seeds";

  prefix mpl;

  organization
    "IETF ROLL (Routing over Low power and lossy networks) Working Group";

  contact
    "WG Web:  http://tools.ietf.org/wg/roll/
    WG List:  mailto:roll@ietf.org

    WG Chair: Peter van der Stok
               mailto:consultancy@vanderstok.org

    WG Chair: Ines Robles
               mailto:maria.ines.robles@ericsson.com

    Editor:   Peter van der Stok
               mailto:consultancy@vanderstok.org";

  description
```

"This module contains information about the operation of the MPL protocol.

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or

without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision "2016-10-25" {  
    description "Initial revision.";   
    reference  
        "I-D:draft-vanderstok-roll-mpl-yang: A YANG model for Multicast Protocol  
for Low power and lossy Networks (MPL)";  
}
```

```
list mpl-seeds{  
    key "seedID domainID";  
    config false;  
    description  
        "List describes all seeds that are active in the server. Seed informatio  
n contains the message buffer contents and the operational values of I, c, seque  
nce number and the life-times per message.";
```

```
    leaf seedID{  
        type uint64;  
        description  
            "value uniquely identifies the MPL Seed within a MPL  
domain.";  
    }
```

```
    leaf domainID{  
        type uint16;  
        description  
            "together with seedID uniquely identifies buffer set.";  
    }
```

```
    leaf local {  
        type boolean;  
        description  
            "When local == TRUE, seed is located in this forwarder.  
            WHEN local == false, seed is located in different  
forwarder.";
```

```
    }

    leaf generate-seqno {
      type uint8;
      description
        "Sequence number of next message to be generated by this local seed.";
    }

    leaf life-time {
      type uint64;
      description
        " Minimum remaining lifetime of the seed entry in
         SE_LIFETIME units.";
    }

    leaf min-seqno{
      type uint8;
      description
        "Lower bound sequence number in the buffer of the seed.";
    }

    leaf data-number{
      type uint8;
      description
        "Number of currently buffered data messages.";
    }

    leaf control-number{
      type uint8;
      description
        "Number of currently buffered control messages.";
    }

    list buffered-messages{
      key seqno;
      description
        " status of trickle intervals of the buffered message identified by se
qno. and seed/domain";

      leaf seqno{
        type uint8;
        description
          "Sequence number of message.";
      }

      leaf I{
        type uint8;
        description
          "Current Trickle timer interval size in SE-LIFETIME units.";
      }
    }
  }
}
```

```
    }

    leaf c{
      type uint8;
      description
        "number of times that copy of this message has been
         received in this interval.";
    }

    leaf e{
      type uint8;
      description
        "number of Trickle time expirations since last
         Trickle timer reset.";
    }

    leaf t{
      type uint8;
      description
        " Time expressed in SE-LIFETIME units
         that message will be (is) forwarded";
    }
  } // list seed-timers
} // list MPL-seeds
} // module ietf-yang-mpl-seeds
```

<CODE ENDS>

3.4. yang-mpl-statistics module

This module specifies the operation of the MPL forwarder expressed in number of messages and copies. The values are acquired by the client and set by the server. Statistics are specified per seed and domain. In constrained devices there will be only one domain, but probably multiple seeds.

The parameter *k* determines how many copies of a message can be forwarded. The counters *c-too-high*, *nr-forwarded*, and *nr-not-forwarded* give insight in the consequences of the current value of *k*.

The other counters give insight in the loss of messages caused by the medium or forwarding delays. The inconsistent/consistent counters indicate when consistent or inconsistent messages were received according to the definition of consistent in [RFC7731].

```
<CODE BEGINS>file "ietf-yang-mpl-statistics@2016-10-25.yang"

module ietf-yang-mpl-statistics {

  yang-version 1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-mpl-statistics";

  prefix mpl;

  organization
    "IETF ROLL (Routing over Low power and lossy networks) Working Group";

  contact
    "WG Web:   http://tools.ietf.org/wg/roll/
    WG List:  mailto:roll@ietf.org

    WG Chair: Peter van der Stok
               mailto:consultancy@vanderstok.org

    WG Chair: Ines Robles
               mailto:maria.ines.robles@ericsson.com

    Editor:   Peter van der Stok
               mailto:consultancy@vanderstok.org";

  description
    "This module contains information about the operation of the MPL protocol.

    Copyright (c) 2016 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision "2016-10-25" {
    description "Initial revision.";
```

```
reference
  "I-D:draft-vanderstok-roll-mpl-yang: A YANG model for Multicast Protocol
for Low power and lossy Networks (MPL)";
}

list mpl-statistics{
  key "seedID domainID";

  config false;

  description
    "List describes performance statistics integrated over the messages iden-
tified by seed and domain identifiers. A forwarder can receive and forward multi-
ple copies of a message uniquely identified by seqno, domain, and seed.";

  leaf seedID{
    type uint64;
    description
      "value uniquely identifies the MPL Seed within a MPL
domain.";
  }

  leaf domainID{
    type uint16;
    description
      "together with seed-ID uniquely identifies buffer set.";
  }

  leaf c-too-high {
    type uint64;
    description
      "Number of times that a copy was not forwarded
because c > k.";
  }

  leaf nr-forwarded {
    type uint64;
    description
      "number of times copies are forwarded, while c <= k.";
  }

  leaf nr-of-messages-received{
    type uint64;
    description
      "number of messages (one or more copies) received, must be smaller tha
n or equal to seqno.";
  }

  leaf nr-of-copies-received{
    type uint64;
```

```
    description
      "total number of message copies received.";
  }

  leaf nr-of-messages-forwarded{
    type uint64;
    description
      "number of forwarded messages, must be smaller than or equal to nr-of-
messages-received.";
  }

  leaf nr-of-copies-forwarded{
    type uint64;
    description
      "number of forwarded copies, can be larger than number-of-copies-recei
ved.";
  }

  leaf nr-of-refused{
    type uint64;
    description
      "number of refused copies because seqno too small.";
  }

  leaf nr-of-missed{
    type uint64;
    description
      "number of messages that were not received (derived from gaps in recei
ved seqno's.)";
  }

  leaf nr-of-notreceived{
    type uint64;
    description
      "number of messages that were not received
      according to control message.";
  }

  leaf nr-of-inconsistent-data{
    type uint64;
    description
      "number of inconsistent data messages.";
  }

  leaf nr-of-consistent-data{
    type uint64;
    description
      "number of consistent data messages.";
  }

  leaf nr-of-consistent-control{
```



```
    type uint64;
    description
        "number of consistent control messages.";
    }

    leaf nr-of-inconsistent-control{
    type uint64;
    description
        "number of inconsistent control messages.";
    }
    } // list mpl statistics
} // module ietf-yang-mpl-statistics
```

<CODE ENDS>

4. IANA Considerations

This specification has no consequences for IANA.

5. Acknowledgements

Andy Bierman has commented on the use of YANG for mpl. YANG doctors pointed out a wrong use of config.

6. Changelog

Changes from version 00 to version 01

- o config false in "statistics" and "seeds" modules
- o separated into 4 modules
- o inserted choice in domain modules
- o renamed some parameters
- o Introduced section per module

7. References

7.1. Normative References

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

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <<http://www.rfc-editor.org/info/rfc7223>>.
- [RFC7731] Hui, J. and R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)", RFC 7731, DOI 10.17487/RFC7731, February 2016, <<http://www.rfc-editor.org/info/rfc7731>>.

7.2. Informative References

- [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <<http://www.rfc-editor.org/info/rfc6206>>.
- [RFC7774] Doi, Y. and M. Gillmore, "Multicast Protocol for Low-Power and Lossy Networks (MPL) Parameter Configuration Option for DHCPv6", RFC 7774, DOI 10.17487/RFC7774, March 2016, <<http://www.rfc-editor.org/info/rfc7774>>.

Author's Address

Peter van der Stok (editor)
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI: www.vanderstok.org