ROLL Working Group
Internet-Draft
Updates: 6550 (if approved)
Intended status: Standards Track
Expires: April 24, 2017

M. Robles
Ericsson
M. Richardson
SSW
P. Thubert
Cisco

October 21, 2016

When to use RFC 6553, 6554 and IPv6-in-IPv6
draft-ietf-roll-useofrplinfo-09

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 RFC 6553, RFC 6554 and IPv6-in-IPv6 encapsulation is required. This analysis provides the basis on which to design efficient compression of these headers.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2017.

Table of Contents

1. Introduction

RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [RFC6550] is a routing protocol for constrained networks. RFC 6553 [RFC6553] defines the "RPL option" (RPI), carried within the IPv6 Hop-by-Hop header to quickly identify inconsistencies (loops) in the routing topology. RFC 6554 [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 implementors agree when artifacts are necessary, or when they can be safely omitted, or removed.

An interim meeting went through the 24 cases defined here to discover if there were any shortcuts, and this document is the result of that discussion. This document should not be defining anything new, but it may clarify what is correct and incorrect behaviour.

The related document A Routing Header Dispatch for 6LoWPAN (6LoRH) [I-D.ietf-roll-routing-dispatch] defines a method to compress RPL Option information and Routing Header type 3 [RFC6554], an efficient IP-in-IP technique, and use cases proposed for the [Second6TischPlugtest] involving 6loRH.

The related document updates [RFC6550]. In general, any packet that leaves the RPL domain of an LLN (or leaves the LLN entirely) will NOT be discarded, when it has the [RFC6553] RPL Option Header known as the RPI or [RFC6554] SRH3 Extension Header (S)RH3. Due to changes to [I-D.ietf-6man-rfc2460bis] the RPI Hop-by-Hop option MAY be left in place even if the end host does not understand it.

2. Terminology and Requirements Language

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

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

RPL-node: It is device which implements RPL, thus we can say that the device is RPL-capable or RPL-aware. Please note that the device can be found inside the LLN or outside LLN. In this document a RPL-node which is a leaf is called RPL-aware-leaf.

RPL-not-capable: It is device which do not implement RPL, thus we can say that the device is not-RPL-aware. Please note that the device can be found inside the LLN. In this document a not-RPL-node which is a leaf is called not-RPL-aware-leaf.

2.1. hop-by-hop IPv6-in-IPv6 headers

The term "hop-by-hop IPv6-in-IPv6" header refers to: adding a header that originates from a node to an adjacent node, using the addresses (usually the GUA or ULA, but could use the link-local addresses) of each node. If the packet must traverse multiple hops, then it must be decapsulated at each hop, and then re-encapsulated again in a similar fashion.

3. Sample/reference topology

A RPL network is composed of a 6LBR (6LoWPAN Border Router), Backbone Router (6BBR), 6LR (6LoWPAN Router) and 6LN (6LoWPAN Node) as leaf logically organized in a DODAG structure (Destination Oriented Directed Acyclic Graph).

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 showed in Figure 1.

RPL supports two modes of Downward traffic: in storing mode (RPL-SM), it is fully stateful or an in non-storing (RPL-NSM), 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 storing and non-storing nodes is not supported with the current specifications at the time of writing this document.

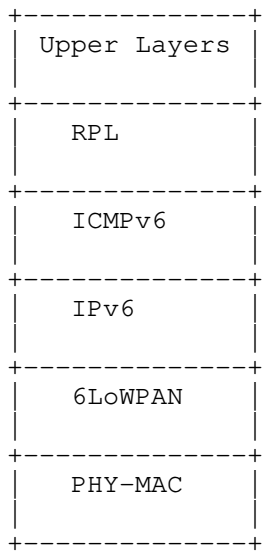


Figure 1: RPL Stack.

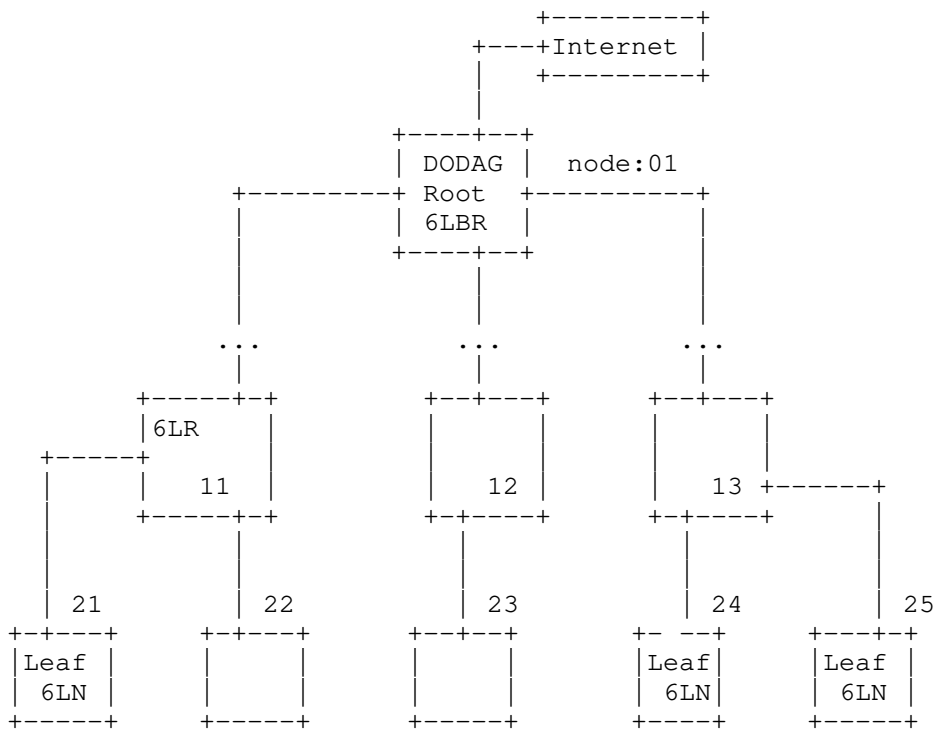


Figure 2: A reference RPL Topology.

In Figure 2 is showed the reference RPL Topology for this document. The numbers in or above the nodes are there so that they may be referenced in subsequent sections. In the figure, a 6LN can be a router or a host. The 6LN leafs marked as (21) is a RPL host that does not have forwarding capability and (25) is a RPL router. The leaf marked 6LN (24) is a device which does not speak RPL at all (not-RPL-aware), but uses Router-Advertisements, 6LowPAN DAR/DAC and efficient-ND only to participate in the network [RFC6775]. In the document this leaf (24) is often named IPv6 node. The 6LBR in the figure is the root of the Global DODAG.

This document is in part motivated by the work that is ongoing at the 6TiSCH working group. The 6TiSCH architecture [I-D.ietf-6tisch-architecture] draft explains the network architecture of a 6TiSCH network.

4. Use cases

In data plane context a combination of RFC6553, RFC6554 and IPv6-in-IPv6 encapsulation is going to be analyzed for the following traffic flows.

This version of the document assumes the changes in [I-D.ietf-6man-rfc2460bis] are passed (at the time to write this specification, the draft is on version 05).

RPL-aware-leaf to root

root to RPL-aware-leaf

not-RPL-aware-leaf to root

root to not-RPL-aware-leaf

RPL-aware-leaf to Internet

Internet to RPL-aware-leaf

not-RPL-aware-leaf to Internet

Internet to not-RPL-aware-leaf

RPL-aware-leaf to RPL-aware-leaf (storing and non-storing)

RPL-aware-leaf to not-RPL-aware-leaf (non-storing)

not-RPL-aware-leaf to RPL-aware-leaf (storing and non-storing)

not-RPL-aware-leaf to not-RPL-aware-leaf (non-storing)

This document assumes 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 [RFC2460]. Extensions may not be added or removed except by the sender or the receiver.

But, options in the Hop-by-Hop option which are marked with option type 01 ([RFC2460] section 4.2 and [I-D.ietf-6man-rfc2460bis]) SHOULD be ignored when received by a host or router which does not understand that option.

This means that in general, any packet that leaves the RPL domain of an LLN (or leaves the LLN entirely) will NOT be discarded, when it

has the [RFC6553] RPL Option Header known as the RPI or [RFC6554] SRH3 Extension Header (S)RH3.

The recent change to the second of these rules it means that the RPI Hop-by-Hop option MAY be left in place even if the end host does not understand it.

NOTE: There is some possible security risk when the RPI information is released to the Internet. At this point this is a theoretical situation. It is clear that the RPI option would waste some network bandwidth when it escapes.

An intermediate router that needs to add an extension header (SHR3 or RPI Option) must encapsulate the packet in an (additional) outer IP header. The new header can be placed is placed after this new outer IP header.

A corollory is that an SHR3 or RPI 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. When it does so, 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 addresses.

Both RPI and RH3 headers may be modified in very specific ways by routers on the path of the packet without the need to add to remove an encapsulating header. Both headers were designed with this modification in mind, and both the RPL RH 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.

RPI should be present in every single RPL data packet. There is one exception in non-storing mode: when a packet is going down from the root. In a downward non-storing mode, the entire route is written, so there can be no loops by construction, nor any confusion about which forwarding table to use (as the root has already made all routing decisions). There still may be cases (such as in 6tisch) where the instanceID portion of the RPI header may still be needed to pick an appropriate priority or channel at each hop.

In the tables present in this document, the term "RPL aware leaf" is has been shortened to "Raf", and "not-RPL aware leaf" has been shortened to "˜Raf" to make the table fit in available space.

The earlier examples are more extensive to make sure that the process is clear, while later examples are more consise.

5. Storing mode

In storing mode (fully stateful), the sender cannot determine whether the destination is RPL-capable and thus would need an IP-in-IP header. The IP-in-IP header needs to be addressed on a hop-by-hop basis so that the last 6LR can remove the RPI header. Additionally, The sender can determine if the destination is inside the LLN by looking if the destination address is matched by the DIO's PIO option.

The following table summarizes what headers are needed in the following scenarios, and indicates when the IP-in-IP header must be inserted on a hop-by-hop basis, and when it can target the destination node directly. There are three possible situations: hop-by-hop necessary (indicated by "hop"), or destination address possible (indicated by "dst"). In all cases hop by hop can be used. In cases where no IP-in-IP header is needed, the column is left blank.

The leaf can be a router 6LR or a host, both indicated as 6LN.

Use Case	RPI	RH3	IP-in-IP	IP-in-IP dst
Raf to root	Yes	No	No	--
root to Raf	Yes	No	No	--
root to ~Raf	Yes	No	No	--
~Raf to root	Yes	No	Yes	root
Raf to Int	Yes	No	No	--
Int to Raf	Yes	No	Yes	raf
~Raf to Int	Yes	No	Yes	root
Int to ~Raf	Yes	No	Yes	hop
Raf to Raf	Yes	No	No	--
Raf to ~Raf	Yes	No	No	--
~Raf to Raf	Yes	No	Yes	dst
~Raf to ~Raf	Yes	No	Yes	hop

Table 1: Headers needed in Storing mode: RPI, RH3, IP-in-IP encapsulation

5.1. Example of Flow from RPL-aware-leaf to root

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

As stated in Section 16.2 of [RFC6550] a RPL-aware-leaf node does not generally issue DIO messages; a leaf node accepts DIO messages

from upstream. (When the inconsistency in routing occurs, a leaf node will generate a DIO with an infinite rank, to fix it). It may issue DAO and DIS messages though it generally ignores DAO and DIS messages.

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR1,... --> 6LRN --> root (6LBR)

As it was mentioned In this document 6LRs, 6LBR are always full-fledge RPL routers.

The 6LN inserts the RPI header, and sends the packet to 6LR which decrements the rank in RPI and sends the packet up. When the packet arrives at 6LBR, the RPI is removed and the packet is processed.

No IP-in-IP header is required.

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

Header	6LN	6LR	6LBR
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Storing: Summary of the use of headers from RPL-aware-leaf to root

5.2. Example of Flow from root to RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR1,... --> 6LRN --> RPL-aware-leaf (6LN)

In this case the 6LBR inserts RPI header and sends the packet down, the 6LR is going to increment the rank in RPI (examines instanceID for multiple tables), the packet is processed in 6LN and RPI removed.

No IP-in-IP header is required.

Header	6LBR	6LR	6LN
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Storing: Summary of the use of headers from root to RPL-aware-leaf

5.3. Example of Flow from root to not-RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR1,... --> 6LRN --> not-RPL-aware-leaf (IPv6)

As the RPI extension can be ignored by the not-RPL-aware leaf, this situation is identical to the previous scenario.

Header	6LBR	6LR(1..N)	6LN
Inserted headers	RPI	--	--
Removed headers	--	--	--
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	RPI (Ignored)

Storing: Summary of the use of headers from root to not-RPL-aware-leaf

5.4. Example of Flow from not-RPL-aware-leaf to root

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR1,... --> 6LRN --> root (6LBR)

When the packet arrives from IPv6 node to 6LR, the 6LR1 will insert an RPI header, encapsuladed in a IPv6-in-IPv6 header. The IPv6-in-IPv6 header can be addressed to the next hop, or to the root. The root removes the header and processes the packet.

Header	IPv6	6LR1	6LRN	6LBR
Inserted headers	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

Storing: Summary of the use of headers from not-RPL-aware-leaf to root

5.5. Example of Flow from RPL-aware-leaf to Internet

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.

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR1,... --> 6LRN --> root (6LBR) --> Internet

No IP-in-IP header is required.

Header	6LN	6LR(1..N)	6LBR	Internet
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	--	RPI (Ignored)

Storing: Summary of the use of headers from RPL-aware-leaf to Internet

5.6. Example of Flow from Internet to RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR1,... --> 6LRN --> RPL-aware-leaf (6LN)

When the packet arrives from Internet to 6LBR the RPI header is added in a outer IPv6-in-IPv6 header and sent to 6LR, which modifies the rank in the RPI. When the packet arrives at 6LN the RPI header is removed and the packet processed.

Header	Internet	6LBR	6LR(1...N)	6LN
Inserted headers	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

Storing: Summary of the use of headers from Internet to RPL-aware-leaf

5.7. Example of Flow from not-RPL-aware-leaf to Internet

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR1,... --> 6LRN --> root (6LBR) --> Internet

The 6LR1 node will add an IP-in-IP(RPI) header addressed either to the root, or hop-by-hop such that the root can remove the RPI header before passing upwards.

The originating node will ideally leave the IPv6 flow label as zero so that it can be better compressed through the LLN, and the 6LBR will set the flow label to a non-zero value when sending to the Internet.

Header	IPv6	6LR1	6LBN	6LBR	Internet
Inserted headers	--	IP-in-IP(RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP(RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--	--
Untouched headers	--	--	--	--	--

Storing: Summary of the use of headers from not-RPL-aware-leaf to Internet

5.8. Example of Flow from Internet to non-RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR1,... --> 6LRN --> not-RPL-aware-leaf (IPv6)

The 6LBR will have to add an RPI header within an IP-in-IP header. The IP-in-IP can be addressed to the not-RPL-aware-leaf, leaving the RPI inside.

The 6LBR MAY set the flow label on the inner IP-in-IP header to zero in order to aid in compression, as the packet will not emerge again from the LLN.

Header	Internet	6LBR	6LR(1...N)	IPv6
Inserted headers	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	IP-in-IP(RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	RPI (Ignored)

Storing: Summary of the use of headers from Internet to non-RPL-aware-leaf

5.9. Example of Flow from RPL-aware-leaf to RPL-aware-leaf

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. Section 9 in [RFC6550].

In this case the flow comprises:

6LN --> 6LR1 --> common parent (6LRx) --> 6LRN --> 6LN

This case is assumed in the same RPL Domain. In the common parent, the direction of RPI is changed (from increasing to decreasing the rank).

While the 6LR nodes will update the RPI, no node needs to add or remove the RPI, so no IP-in-IP headers are necessary. This may be done regardless of where the destination is, as the included RPI will be ignored by the receiver.

Header	6LN src	6LR1	6LRx (common parent)	6LRN	6LN dst
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	RPI
Re-added headers	--	--	--	--	--
Modified headers	--	RPI (decreasing rank)	RPI (increasing rank)	--	--
Untouched headers	--	--	--	--	--

Storing: Summary of the use of headers for RPL-aware-leaf to RPL-aware-leaf

5.10. Example of Flow from RPL-aware-leaf to non-RPL-aware-leaf

In this case the flow comprises:

6LN --> 6LR1 --> common parent (6LRx) --> 6LRN --> not-RPL-aware 6LN (IPv6)

This situation is identical to the previous situation Section 5.9

Header	6LN src	6LR1	6LRx (common parent)	6LRN	IPv6
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	RPI
Re-added headers	--	--	--	--	--
Modified headers	--	RPI (decreasing rank)	RPI (increasing rank)	--	--
Untouched headers	--	--	--	--	RPI(Ignored)

Storing: Summary of the use of headers for RPL-aware-leaf to RPL-aware-leaf

5.11. Example of Flow from not-RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR1 --> common parent (6LRx) --> 6LRN --> 6LN

The 6LR1 receives the packet from the the IPv6 node and inserts and the RPI header encapsulated in IPv6-in-IPv6 header. The IP-in-IP header is addressed to the destination 6LN.

Header	IPv6	6LR1	common parent (6LRx)	6LRn	6LN
Inserted headers	--	IP-in-IP(RPI)	--	--	--
Removed headers	--	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--	--

Storing: Summary of the use of headers from not-RPL-aware-leaf to RPL-aware-leaf

5.12. Example of Flow from not-RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6 src)--> 6LR1 --> 6LR2 --> root (6LBR) --> 6LRn --> not-RPL-aware 6LN (IPv6 dst)

This flow is identical to Section 5.11

The 6LR receives the packet from the the IPv6 node and inserts the RPI header (RPIa) encapsulated in IPv6-in-IPv6 header. The IPv6-in-IPv6 header is addressed to the 6LBR. The 6LBR remove the IPv6-in-IPv6 header and insert another one (RPIb) with destination to 6LRn node.

Header	IPv6 src	6LR1	6LR2	6LBR	6LRn	IPv6 dst
Inserted headers	--	IP-in-IP(RPIa)	--	IP-in-IP(RPIb)	--	--
Removed headers	--	--	--	--	--	--
Re-added headers	--	--	--	--	IP-in-IP(RPIb)	--
Modified headers	--	--	IP-in-IP(RPIa)	--	IP-in-IP(RPIb)	--
Untouched headers	--	--	--	--	--	--

Storing: Summary of the use of headers from not-RPL-aware-leaf to non-RPL-aware-leaf

6. Non Storing mode

Use Case	RPI	RH3	IP-in-IP	IP-in-IP dst
Raf to root	Yes	No	No	--
root to Raf	Opt	Yes	No	--
root to ~Raf	No	Yes	Yes	6LR
~Raf to root	Yes	No	Yes	root
Raf to Int	Yes	No	Yes	root
Int to Raf	Opt	Yes	Yes	dst
~Raf to Int	Yes	No	Yes	root
Int to ~Raf	Opt	Yes	Yes	6LR
Raf to Raf	Yes	Yes	Yes	root/dst
Raf to ~Raf	Yes	Yes	Yes	root/6LR
~Raf to Raf	Yes	Yes	Yes	root/6LN
~Raf to ~Raf	Yes	Yes	Yes	root/6LR

Table 2: Headers needed in Non-Storing mode: RPI, RH3, IP-in-IP encapsulation

6.1. Example of Flow from RPL-aware-leaf to root

In non-storing mode the leaf node uses default routing to send traffic to the root. The RPI header must be included to avoid/detect loops.

RPL-aware-leaf (6LN) --> 6LR --> root (6LBR)

This situation is the same case as storing mode.

Header	6LN	6LR	6LBR
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Non Storing: Summary of the use of headers from RPL-aware-leaf to root

6.2. Example of Flow from root to RPL-aware-leaf

In this case the flow comprises:

root (6LBR)--> 6LR --> RPL-aware-leaf (6LN)

The 6LBR will insert an RH3, and may optionally insert an RPI header. No IP-in-IP header is necessary as the traffic originates with an RPL aware node, the 6LBR. The destination is known to RPL-aware because, the root knows the whole topology in non-storing mode.

Header	6LBR	6LR	6LN
Inserted headers	(opt: RPI), RH3	--	--
Removed headers	--	--	RH3,RPI
Re-added headers	--	--	--
Modified headers	--	RH3	--
Untouched headers	--	--	--

Non Storing: Summary of the use of headers from root to RPL-aware-leaf

6.3. Example of Flow from root to not-RPL-aware-leaf

In this case the flow comprises:

root (6LBR)--> 6LR1...-->6LRn --> not-RPL-aware-leaf (IPv6)

In 6LBR the RH3 is added, modified in each intermediate 6LR (6LR1 and so on) and it is fully consumed in the last 6LR (6LRn), but left there. If RPI is left present, the IPv6 node which does not understand it will ignore it (following 2460bis), thus encapsulation is not necesary. Due the complete knowledge of the topology at the root, the 6LBR is able to address the IP-in-IP header to the last 6LR.

Header	6LBR	6LR1	6LRn	IPv6
Inserted headers	(opt: RPI), RH3	--	--	--
Removed headers	--	RH3	--	--
Re-added headers	--	--	--	--
Modified headers	--	(opt: RPI), RH3	(opt: RPI), RH3	--
Untouched headers	--	--	--	RPI

Non Storing: Summary of the use of headers from root to not-RPL-aware-leaf

6.4. Example of Flow from not-RPL-aware-leaf to root

In this case the flow comprises:

IPv6-node --> 6LR1 ...--> 6LRn --> root (6LBR)

In this case the RPI is added by the first 6LR (6LR1), encapsulated in an IP-in-IP header, and is modified in the followings 6LRs. The RPI and entire packet is consumed by the root.

Header	IPv6	6LR1	6LR2	6LBR
Inserted headers	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	IP-in-IP(RPI)	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

Non Storing: Summary of the use of headers from not-RPL-aware-leaf to root

6.5. Example of Flow from RPL-aware-leaf to Internet

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR1 ...--> 6LRn --> root (6LBR) --> Internet

This case is identical to storing-mode case.

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

Header	6LN	6LR(1..N)	6LBR	Internet
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	--	RPI (Ignored)

Non Storing: Summary of the use of headers from RPL-aware-leaf to Internet

6.6. Example of Flow from Internet to RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR1...--> 6LRn --> RPL-aware-leaf (6LN)

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

The RPI may be added or not, it is optional.

Header	Internet	6LBR	6LR	6LN
Inserted headers	--	IP-in-IP(RH3,opt:RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RH3,opt:RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RH3,opt:RPI)	--
Untouched headers	--	--	--	--

Non Storing: Summary of the use of headers from Internet to RPL-aware-leaf

6.7. Example of Flow from not-RPL-aware-leaf to Internet

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR1..--> 6LRn --> root (6LBR) --> Internet

In this case the flow label is recommended to be zero in the IPv6 node. As RPL headers are added in the IPv6 node, the first 6LN will add an RPI header inside a new IP-in-IP header. The IP-in-IP header will be addressed to the root. This case is identical to the storing-mode case (Section 5.7).

Header	IPv6	6LR1	6LRn	6LBR	Internet
Inserted headers	--	IP-in-IP(RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP(RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--	--
Untouched headers	--	--	--	--	--

Non Storing: Summary of the use of headers from not-RPL-aware-leaf to Internet

6.8. Example of Flow from Internet to non-RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR1...--> 6LRn --> not-RPL-aware-leaf (IPv6)

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

Header	Internet	6LBR	6LR1	6LRn	IPv6
Inserted headers	--	IP-in-IP(RH3,opt:RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP(RH3, RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP(RH3, RPI)	IP-in-IP(RH3, RPI)	--
Untouched headers	--	--	--	--	RPI

NonStoring: Summary of the use of headers from Internet to non-RPL-aware-leaf

6.9. Example of Flow from RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

6LN --> 6LR1 --> root (6LBR) --> 6LRN --> 6LN

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

The originating node SHOULD put the RPI into an IP-in-IP header addressed to the root, so that the 6LBR can remove that header. If it does not, then additional resources are wasted on the way down to carry the useless RPI option.

The 6LBR will need to insert an RH3 header, which requires that it add an IP-in-IP header. It SHOULD be able to remove the RPI, as it was contained in an IP-in-IP header addressed to it. Otherwise, there MAY be an RPI header buried inside the inner IP header, which should get ignored.

Networks that use the RPL P2P extension [RFC6997] are essentially non-storing DODAGs and fall into this scenario or scenario Section 6.2, with the originating node acting as 6LBR.

Header	6LN src	6LR1	6LBR	6LRN	6LN dst
Inserted headers	IP-in-IP(RPI1)	--	IP-in-IP(RH3 to 6LN, opt RPI2)	--	--
Removed headers	--	--	IP-in-IP(RPI1)	--	IP-in-IP(RH3, opt RPI2)
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	--	--
Untouched headers	--	--	--	--	--

Non Storing: Summary of the use of headers for RPL-aware-leaf to RPL-aware-leaf

6.10. Example of Flow from RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

6LN --> 6LR1 --> root (6LBR) --> 6LRn --> not-RPL-aware (IPv6)

As in the previous case, the 6LN will insert an RPI (RPI1) header which MUST be in an IP-in-IP header addressed to the root so that the 6LBR can remove this RPI. The 6LBR will then insert an RH3 inside a new IP-in-IP header addressed to the 6LN destination node. The RPI is optional from 6LBR to 6LRn (RPI2).

Header	6LN	6LR1	6LBR	6LRn	IPv6
Inserted headers	IP-in-IP(RPI1)	--	IP-in-IP(RH3, opt RPI2)	--	--
Removed headers	--	--	IP-in-IP(RPI1)	IP-in-IP(RH3, opt RPI2)	--
Re-added headers	--	--	--	--	--
Modified headers	--	IP-in-IP(RPI1)	--	IP-in-IP(RH3, opt RPI2)	--
Untouched headers	--	--	--	--	opt RPI2

Non Storing: Summary of the use of headers from RPL-aware-leaf to not-RPL-aware-leaf

6.11. Example of Flow from not-RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR1 --> root (6LBR) --> 6LRn --> 6LN

This scenario is mostly identical to the previous one. The RPI is added by the first 6LR (6LR1) inside an IP-in-IP header addressed to the root. The 6LBR will remove this RPI, and add it's own IP-in-IP header containing an RH3 header and optional RPI (RPI2).

Header	IPv6	6LR1	6LBR	6LRn	6LN
Inserted headers	--	IP-in-IP(RPI1)	IP-in-IP(RH3, opt RPI2)	--	--
Removed headers	--	--	IP-in-IP(RPI1)	--	IP-in-IP(RH3, opt RPI2)
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	IP-in-IP(RH3, opt RPI2)	--
Untouched headers	--	--	--	--	--

Non Storing: Summary of the use of headers from not-RPL-aware-leaf to RPL-aware-leaf

6.12. Example of Flow from not-RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6 src)--> 6LR1 --> root (6LBR) --> 6LRn --> not-RPL-aware (IPv6 dst)

This scenario is the combination of the previous two cases.

Header	IPv6 src	6LR1	6LBR	6LRn	IPv6 dst
Inserted headers	--	IP-in-IP(RPI1)	IP-in-IP(RH3)	--	--
Removed headers	--	--	IP-in-IP(RPI1)	IP-in-IP(RH3, opt RPI2)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	--	--
Untouched headers	--	--	--	--	--

Non Storing: Summary of the use of headers from not-RPL-aware-leaf to not-RPL-aware-leaf

7. Observations about the cases

7.1. Storing mode

[I-D.ietf-roll-routing-dispatch] shows that this hop-by-hop IP-in-IP header can be compressed down to {TBD} bytes.

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

Thanks to the relaxation of the RFC2406 rule about discarding unknown Hop-by-Hop options, there is no longer any uncertainty about when to use an IPIP header in the storing mode case. The RPI header SHOULD always be added when 6LRs originate packets (without IPIP headers), and IPIP headers should always be added (addressed to the root when on the way up, to the end-host when on the way down) when a 6LR finds it needs to insert an RPI header. (XXX - this is a problem for storing mode optimization)

In order to support the above two cases with full generality, the different situations (always do IP-in-IP vs never use IP-in-IP) should be signaled in the RPL protocol itself.

7.2. Non-Storing mode

In the non-storing case, dealing with non-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 non-RPL aware leaf nodes because it will receive a DAO about that node from the 6LN immediately above that node. This means that the non-storing mode case can avoid ever using hop-by-hop IP-in-IP headers.

[I-D.ietf-roll-routing-dispatch] shows how the destination=root, and destination=6LN IP-in-IP header can be compressed down to {TBD} bytes.

Unlike in the storing mode case, there is no need for all nodes to know about the existence of non-RPL aware nodes. Only the 6LBR needs to change when there are non-RPL aware nodes. Further, in the non-storing case, the 6LBR is informed by the DAOs when there are non-RPL aware nodes.

8. 6LoRH Compression cases

The [I-D.ietf-roll-routing-dispatch] proposes a compression method for RPI, RH3 and IPv6-in-IPv6.

In Storing Mode, for the examples of Flow from RPL-aware-leaf to non-RPL-aware-leaf and non-RPL-aware-leaf to non-RPL-aware-leaf comprise an IP-in-IP and RPI compression headers. The type of this case is critical since IP-in-IP is encapsulating a RPI header.

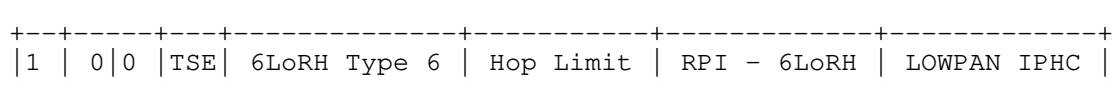


Figure 3: Critical IP-in-IP (RPI).

9. IANA Considerations

There are no IANA considerations related to this document.

10. Security Considerations

The security considerations covering of [RFC6553] and [RFC6554] apply when the packets get into RPL Domain.

11. Acknowledgments

This work is partially funded by the FP7 Marie Curie Initial Training Network (ITN) METRICS project (grant agreement No. 607728).

The authors would like to acknowledge the review, feedback, and comments of Robert Cragie, Simon Duquennoy, Cenk Guendogan, Peter van der Stok, Xavier Vilajosana and Thomas Watteyne.

12. References

12.1. Normative References

[I-D.ietf-6man-rfc2460bis]
Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-ietf-6man-rfc2460bis-07 (work in progress), October 2016.

[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>.

[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>.

[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-routing-dispatch] Thubert, P., Bormann, C., Toutain, L., and R. Cragie, "6LoWPAN Routing Header", draft-ietf-roll-routing-dispatch-02 (work in progress), October 2016.

[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>.

[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>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7102>.

[Second6TischPlugtest] "2nd 6Tisch Plugtest", <http://www.ietf.org/mail-archive/web/6tisch/current/pdfgDMQcdCkRz.pdf>.

Authors' Addresses

Maria Ines Robles
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: maria.ines.robles@ericsson.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
Village d'Entreprises Green Side 400, Avenue de Roumanille
Batiment T3, Biot - Sophia Antipolis 06410
France

Email: pthubert@cisco.com

ROLL Working Group
Internet-Draft
Updates: 6553, 6550, 8138 (if approved)
Intended status: Standards Track
Expires: January 5, 2020

M. Robles
Aalto
M. Richardson
SSW
P. Thubert
Cisco

July 4, 2019

Using RPL Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane
draft-ietf-roll-useofrplinfo-31

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 (RPL 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 RPL 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.

Table of Contents

1. Introduction

RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [RFC6550] is a routing protocol for constrained networks. RFC6553 [RFC6553] defines the "RPL option" (RPL Packet Information or RPI), carried within the IPv6 Hop-by-Hop header to quickly identify inconsistencies (loops) in the routing topology. 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 analysized 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 RPI option value to make RFC8200 routers ignore this option by default.

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.

Since some of the uses cases here described, use IPv6-in-IPv6 encapsulation. It MUST take in consideration, when encapsulation is applied, the RFC6040 [RFC6040], which defines how the explicit congestion notification (ECN) field of the IP header should be constructed on entry to and exit from any IPV6-in-IPV6 tunnel. Additionally, it is recommended the reading of [I-D.ietf-intarea-tunnels] that explains 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 describes the updates to RFC6553, RFC6550 and RFC 8138. Section 4 provides the reference topology used for the uses cases. Section 5 describes the uses cases included. Section 6 describes the storing mode cases and section 7 the non-storing mode cases. Section 8 describes the operational considerations of supporting not-RPL-aware-leaves. Section 9 depicts operational considerations for the proposed change on RPL Option type, section 10 the IANA considerations and then section 11 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.

RPL-aware-node: 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 which is a leaf of a (Destination Oriented Directed Acyclic Graph) DODAG.

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 which is a leaf of a (Destination Oriented Directed Acyclic Graph) DODAG.

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 transition that involves having a network with different values of RPL Option Type. Thus the network does not work correctly (Lack of interoperation).

Hop-by-hop re-encapsulation: The term "hop-by-hop re-encapsulation" header refers to adding a header that originates from a node to an adjacent node, using the addresses (usually the GUA or ULA, but could use the link-local addresses) of each node. If the packet must traverse multiple hops, then it must be decapsulated at each hop, and then re-encapsulated again in a similar fashion.

Non-storing Mode (Non-SM): RPL mode of operation in which the RPL-aware-nodes send information to the root about its parents. Thus,

the root know the topology, then the intermediate 6LRs do not maintain routing state so that 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.

Due to lack of space in some figures (tables) we refers 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.

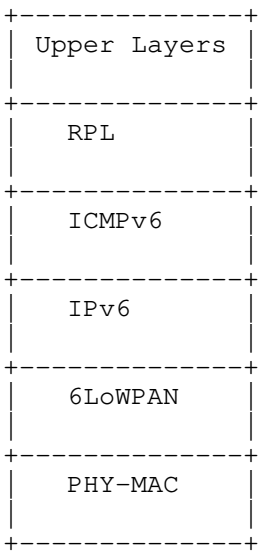


Figure 1: RPL Stack.

RPL supports two modes of Downward 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 storing and non-storing nodes is not supported with the current specifications at the time of writing this document.

4. Updates to RFC6553, RFC6550 and RFC8138

4.1. Updates to RFC6553: Indicating the new RPI value.

This modification is required to be able to send, for example, IPv6 packets from a RPL-Aware-Leaf to a not-RPL-aware node through Internet (see Section 7.2.1), without requiring IPv6-in-IPv6 encapsulation.

[RFC6553] (Section 6, Page 7) states as shown in Figure 2, that in the Option Type field of the RPL Option header, 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 2: Option Type in RPL Option.

This document illustrates that is 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] that compresses the IPv6-in-IPv6 header, this approach yields extra bytes in a packet which 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.

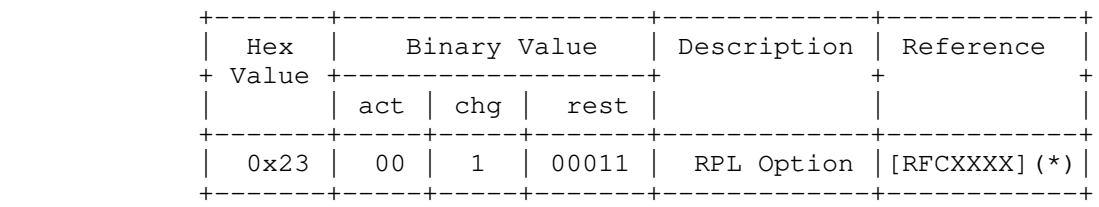
While intention was and remains that the Hop-by-Hop header with a RPL option should be confined within the RPL domain, this specification modifies this behavior in order to reduce the dependency on IPv6-in-IPv6 and protect the constrained devices. Section 4 of [RFC8200] clarifies the behaviour of routers 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 field to (Figure 3): 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 remaining bits serve as the option type and remain as 0x3. 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 [RFC6553] RPL Hop-by-Hop option known as RPI.

With the new Option Type, if an IPv6 (intermediate) node (RPL-not-capable) receives a packet with an 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 3: 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 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 as it was received (This is required because, RPI type code can not be changed by [RFC8200] - Section 4.2). It allows to the network to be incrementally upgraded, and for the DODAG root to know which parts of the network are upgraded.

When originating new packets, implementations SHOULD have an option to determine which value to originate with, this option is controlled by the DIO option described below.

A network which is switching from straight 6LoWPAN compression mechanism to those described in [RFC8138] will experience a flag day in the data compression anyway, and if possible this change can be deployed at the same time.

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 therefore no longer a necessity to remove the artifacts when sending traffic to the Internet. This change clarifies when to use an IPv6-in-IPv6 header, and how to address them: The Hop-by-Hop Options Header containing the RPI option 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 find that it needs to insert a Hop-by-Hop Options Header containing the RPI 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. This means that the non-storing mode case can avoid ever using hop-by-hop re-encapsulation headers for traffic originating from the root to the leafs.

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.

4.2. Updates to RFC6550: 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 (0x23) and old RPI (0x63) nodes, this section defines a flag in the DIO Configuration Option, to indicate when then new RPI value can be safely used. This means, the flag is going to indicate the type of RPI that the network is using. Thus, when a node join to a network will know which value to use. With this, RPL-capable nodes know if it is safe to use 0x23 when creating a new RPI. A node that forwards a packet with an RPI MUST NOT modify the option type of the RPI.

This is done via a DODAG Configuration Option flag which will propagate through the network. 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 (0x63) value.

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.

The DODAG Configuration Option has a Flag field which is modified by this document. Currently, the DODAG Configuration Option in [RFC6550] states: "the unused bits MUST be initialize to zero by the sender and MUST be ignored by the receiver".

Bit number three of the flag field in the DODAG Configuration option is to be used as shown in Figure 4 :

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 4: DODAG Configuration Option Flag to indicate the RPI-flag-day.

In case of rebooting, the node (6LN or 6LR) does not remember the RPL Option Type, that is if the flag is set, so DIO messages sent by the node would be set with the flag unset until a DIO message is received with the flag set indicating the new RPI value. The node sets to 0x23 if the node supports this feature.

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

This modification is required to be able to decompress the RPL RPI option with the new value (0x23).

RPI-6LoRH header provides a compressed form for the RPL RPI [RFC8138] in section 6. A node that is decompressing this header MUST decompress using the RPL 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 4.2. 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, for the examples of Flow from RAL to RUL and RUL to RUL comprise an IPv6-in-IPv6 and RPI compression 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.

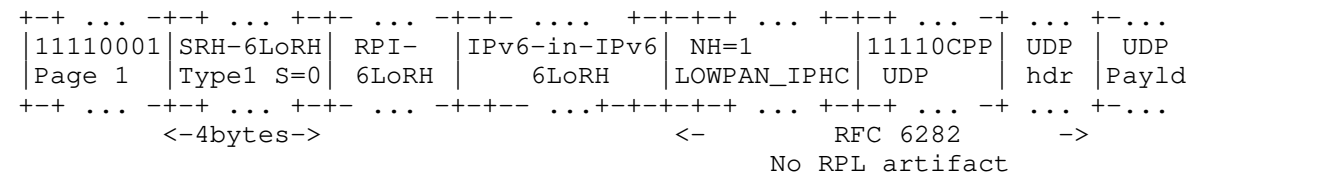


Figure 5: RPI Inserted by the Root in Storing Mode

In Figure 5, the source of the IPv6-in-IPv6 encapsulation is the Root, so it is elided in the IPv6-in-IPv6 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 IPv6-in-IPv6 6LoRH. When the IPv6-in-IPv6 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, Backbone Router (6BBR), 6LR and 6LN as 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, 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 leafs marked as RUL (G and J) are devices which do not speak RPL at all (not-RPL-aware), but uses Router-Advertisements, 6LowPAN DAR/DAC and efficient-ND only to participate in the network [RFC6775]. In the document these leafs (G and J) are also referred to as an IPv6 node.

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

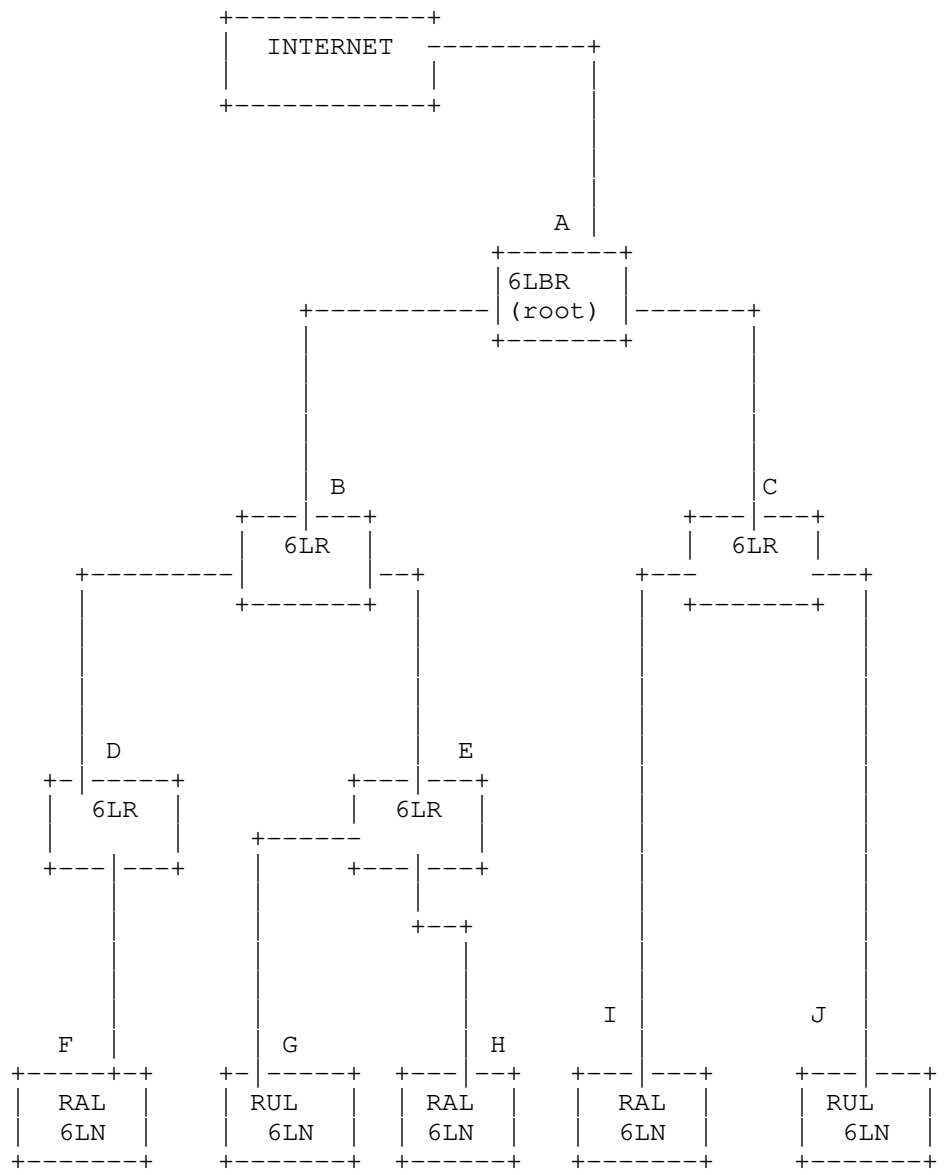


Figure 6: A reference RPL Topology.

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.

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

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 uses 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 Leafs:

- RAL to RAL (storing and non-storing)
- RAL to RUL (non-storing)
- RUL to RAL (storing and non-storing)
- RUL to RUL (non-storing)

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 RPI 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 RH3 or RPI 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. When it does so, 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 RPI and 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.

RPI MUST be present in every single RPL data packet.

Prior to [RFC8138], there was significant interest in removing the RPI for downward flows in non-storing mode. The exception covered a very small number of cases, and causes significant interoperability challenges, yet costed significant code and testing complexity. 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].

The earlier examples are more extensive to make sure that the process is clear, while later examples are more concise.

The uses cases are delineated based on the following requirements:

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

- Because of (1), 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 RPI 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 (0x23).

- Each IPv6 node (including Internet routers) obeys [RFC8200] 8200, so that 0x23 RPI can be safely inserted.

- All 6LRs obey [RFC8200].

- The RPI is ignored at the IPv6 dst node (RPL-unaware-leaf).

- The leaf can be a router 6LR or a host, both indicated as 6LN.

- 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 if the IPv6-in-IPv6 header that is added, must be addressed to the final destination (the RAL node that is the target(tgt)), to the "root" or if a hop-by-hop header must be added (indicated by "hop"). In the hop-by-hop basis, the destination address for the next hop is the link-layer address of the next hop.

In cases where no IPv6-in-IPv6 header is needed, the column states as "No". If the IPv6-in-IPv6 header is needed is a "must".

In all cases the RPI headers are needed, since it identifies inconsistencies (loops) in the routing topology. In all cases the RH3 is not needed because it is not used in storing mode.

In each case, 6LR_i are the intermediate routers from source to destination. "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from source (6LN) to destination.

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	No
	root to RAL	No	No
	root to RUL	No	No
	RUL to root	must	root
Leaf - Internet	RAL to Int	No	No
	Int to RAL	must	RAL (tgt)
	RUL to Int	must	root
	Int to RUL	must	hop
Leaf - Leaf	RAL to RAL	No	No
	RAL to RUL	No	No
	RUL to RAL	must	RAL (tgt)
	RUL to RUL	must	hop

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 --> Node D --> Node B --> Node A root(6LBR)

The 6LN (Node F) inserts the RPI header, and sends the packet to 6LR (Node E) which decrements the rank in 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 header can be removed by the 6LBR because the packet is addressed to the 6LBR. The 6LN must know that it is communicating with the 6LBR to make use of this scenario. The 6LN can know the address of the 6LBR because it knows the address of the root via the DODAGID in the DIO messages.

The Table 1 summarizes what headers are needed for this use case.

Header	6LN src	6LR_i	6LBR dst
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 1: 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 --> Node D --> Node F

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

No IPv6-in-IPv6 header is required.

The Table 2 summarizes what headers are needed for this use case.

Header	6LBR	6LR_i	6LN
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 2: 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)

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

As the RPI extension can be ignored by the not-RPL-aware leaf, this situation is identical to the previous scenario.

The Table 3 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	IPv6 dst node
Inserted headers	RPI	--	--
Removed headers	--	--	--
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	RPI (Ignored)

Table 3: SM: Summary of the use of headers from root to RUL

7.1.4. SM: Example of Flow from RUL to root

In this case the flow comprises:

RUL (IPv6) --> 6LR_1 --> 6LR_i --> root (6LBR)

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

When the packet arrives from IPv6 node (Node G) to 6LR_1 (Node E), the 6LR_1 will insert a RPI header, encapsulated in a IPv6-in-IPv6 header. The IPv6-in-IPv6 header can be addressed to the next hop (Node B), or to the root (Node A). The root removes the header and processes the packet.

The Figure 8 shows the table that summarizes what headers are needed for this use case. [1] refers the case where the IPv6-in-IPv6 header is addressed to the next hop (Node B). [2] refers the case where the IPv6-in-IPv6 header is addressed to the root (Node A).

Header	IPv6 src node	6LR_1	6LR_i	6LBR dst
Inserted headers	--	IP6-IP6(RPI)	IP6-IP6(RPI)[1]	--
Removed headers	--	--	--	IP6-IP6(RPI)[1][2]
Re-added headers	--	--	IP6-IP6(RPI)[1]	--
Modified headers	--	--	IP6-IP6(RPI)[2]	--
Untouched headers	--	--	--	--

Figure 8: 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

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.

In this case the flow comprises:

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

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

No IPv6-in-IPv6 header is required.

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

The Table 4 summarizes what headers are needed for this use case.

Header	6LN src	6LR_i	6LBR	Internet dst
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	RPI	RPI (Ignored)

Table 4: SM: Summary of the use of headers from RAL to Internet

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 --> Node D --> Node F

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

The Figure 9 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	6LN dst
Inserted headers	--	IP6-IP6(RPI)	--	--
Removed headers	--	--	--	IP6-IP6(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP6-IP6(RPI)	--
Untouched headers	--	--	--	--

Figure 9: 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) --> 6LR_1 --> 6LR_i -->root (6LBR) --> Internet

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

The 6LR_1 (i=1) node will add an IPv6-in-IPv6(RPI) header addressed either to the root, or hop-by-hop such that the root can remove the RPI header before passing upwards. The IPv6-in-IPv6 addressed to the root cause less processing overhead. On the other hand, with hop-by-hop the intermediate routers can check the routing tables for a better routing path, thus it could be more efficient and faster. Implementation should decide which approach to take.

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 10 shows the table that summarizes what headers are needed for this use case.

Header	IPv6 src node	6LR_1	6LR_i [i=2,...,n]	6LBR	Internet dst
Inserted headers	--	IP6-IP6(RPI)	IP6-IP6(RPI) [2]	--	--
Removed headers	--	--	IP6-IP6(RPI) [2]	IP6-IP6(RPI) [1][2]	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP6-IP6(RPI) [1]	--	--
Untouched headers	--	--	--	--	--

Figure 10: SM: Summary of the use of headers from RUL to Internet. [1] Case when packet is addressed to the root. [2] Case when the packet is addressed hop-by-hop.

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

In this case the flow comprises:

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

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

The 6LBR will have to add an RPI header within an IPv6-in-IPv6 header. The IPv6-in-IPv6 is addressed hop-by-hop.

The final node should be able to remove one or more IPv6-in-IPv6 headers which are all addressed to it. The final node does not process the RPI, the node ignores the RPI. Furhter details about this are mentioned in [I-D.thubert-roll-unaware-leaves], which specifies RPL routing for a 6LN acting as a plain host and not 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 11 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	IPv6 dst node
Inserted headers	--	IP6-IP6(RPI)	--	--
Removed headers	--	--		IP6-IP6(RPI) RPI Ignored
Re-added headers	--	--	--	--
Modified headers	--	--	IP6-IP6(RPI)	--
Untouched headers	--	--	--	--

Figure 11: 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:

6LN --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> 6LN

For example, a communication flow could be: Node F --> Node D --> Node B --> Node E --> Node H

6LR_ia (Node D) are the intermediate routers from source to the common parent (6LR_x) (Node B) In this case, 1 <= ia <= n, n is the number of routers (6LR) that the packet goes through from 6LN (Node F) to the common parent (6LR_x).

6LR_id (Node E) are the intermediate routers from the common parent (6LR_x) (Node B) to destination 6LN (Node H). In this case, 1 <= id <= m, m is the number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination 6LN.

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 of RPI is changed (from increasing to decreasing the rank).

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 Table 5 summarizes what headers are needed for this use case.

Header	6LN src	6LR_ia	6LR_x (common parent)	6LR_id	6LN dst
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	RPI
Re-added headers	--	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Untouched headers	--	--	--	--	--

Table 5: SM: Summary of the use of headers for RAL to RAL

7.3.2. SM: Example of Flow from RAL to RUL

In this case the flow comprises:

6LN --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> not-RPL-aware 6LN (IPv6)

For example, a communication flow could be: Node F --> Node D --> Node B --> Node E --> Node G

6LR_ia are the intermediate routers from source (6LN) to the common parent (6LR_x) In this case, 1 <= ia <= n, n is the number of routers (6LR) that the packet goes through from 6LN to the common parent (6LR_x).

6LR_id (Node E) are the intermediate routers from the common parent (6LR_x) (Node B) to destination not-RPL-aware 6LN (IPv6) (Node G). In this case, 1 <= id <= m, m is the number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination 6LN.

This situation is identical to the previous situation Section 7.3.1

The Table 6 summarizes what headers are needed for this use case.

Header	6LN src	6LR_ia	6LR_x(common parent)	6LR_id	IPv6 dst node
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	--
Re-added headers	--	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Untouched headers	--	--	--	--	RPI(Ignored)

Table 6: SM: Summary of the use of headers for RAL to RUL

7.3.3. SM: Example of Flow from RUL to RAL

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> 6LN

For example, a communication flow could be: Node G --> Node E --> Node B --> Node D --> Node F

6LR_ia (Node E) are the intermediate routers from source (not-RPL-aware 6LN (IPv6)) (Node G) to the common parent (6LR_x) (Node B). In this case, 1 <= ia <= n, n is the number of routers (6LR) that the packet ges through from source to the common parent.

6LR_id (Node D) are the intermediate routers from the common parent (6LR_x) (Node B) to destination 6LN (Node F). In this case, 1 <= id <= m, m is the number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination 6LN.

The 6LR_ia (ia=1) (Node E) receives the packet from the the IPv6 node (Node G) and inserts and the RPI header encapsulated in IPv6-in-IPv6 header. The IPv6-in-IPv6 header is addressed to the destination 6LN (Node F).

The Figure 12 shows the table that summarizes what headers are needed for this use case.

Header	IPv6 src node	6LR_ia	Common Parent (6LRx)	6LR_id	6LN dst
Inserted headers	--	IP6-IP6(RPI)	--	--	--
Removed headers	--	--	--	--	IP6-IP6(RPI)
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP6-IP6(RPI)	IP6-IP6(RPI)	--
Untouched headers	--	--	--	--	--

Figure 12: 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:

not-RPL-aware 6LN (IPv6 src)--> 6LR_1--> 6LR_ia --> 6LBR --> 6LR_id --> not-RPL-aware 6LN (IPv6 dst)

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

Internal nodes 6LR_ia (e.g: Node E or Node B) is the intermediate router from the not-RPL-aware source (Node G) to the root (6LBR) (Node A). In this case, "1 < ia <= n", n is the number of routers (6LR) that the packet goes through from IPv6 src to the root.

6LR_id (Node C) are the intermediate routers from the root (Node A) to the destination Node J. In this case, 1 <= id <= m, m is the number of routers (6LR) that the packet goes through from the root to destination (IPv6 dst).

The RPI is ignored at the IPv6 dst node.

The 6LR_1 (Node E) receives the packet from the the IPv6 node (Node G) and inserts the RPI header (RPI), encapsulated in an IPv6-in-IPv6 header. The IPv6-in-IPv6 header is addressed hop-by-hop.

The Figure 13 shows the table that summarizes what headers are needed for this use case.

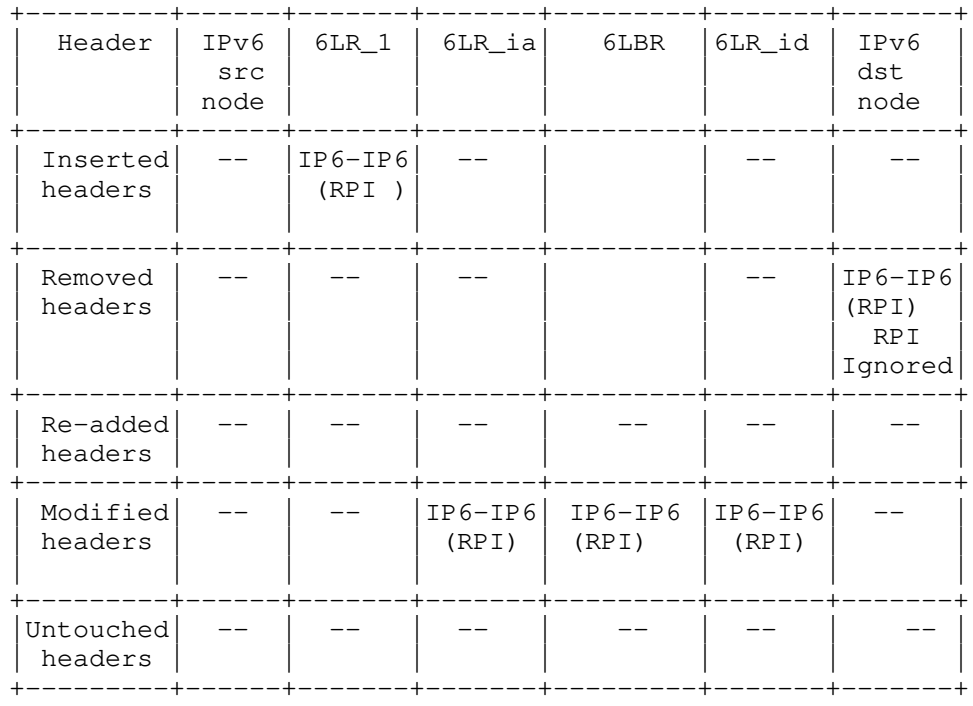


Figure 13: SM: Summary of the use of headers from RUL to RUL

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 not-RPL aware nodes. Only the 6LBR needs to act if compensation is necessary for not-RPL aware receivers.

The following table (Figure 14) summarizes what headers are needed in the following scenarios, and indicates when the RPI, RH3 and IPv6-in-IPv6 header are to be inserted. It depicts the target destination address possible (indicated by "RAL"), to a 6LR (parent of a 6LN) or to the root. In cases where no IPv6-in-IPv6 header is needed, the column states as "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 is always a RPI Present.

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 3). In the Figure the (1) indicates a 6tisch case [RFC8180],

where the RPI header may still be needed for the instanceID to be available for priority/channel selection at each hop.

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

Figure 14: 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 header 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 are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from source (6LN) to destination (6LBR).

This situation is the same case as storing mode.

The Table 7 summarizes what headers are needed for this use case.

Header	6LN src	6LR_i	6LBR dst
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 7: 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 are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from source (6LBR) to destination (6LN).

The 6LBR inserts an RH3, and a RPI header. No IPv6-in-IPv6 header is necessary as the traffic originates with an 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 Table 8 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	6LN dst
Inserted headers	RPI, RH3	--	--
Removed headers	--	--	RH3, RPI
Re-added headers	--	--	--
Modified headers	--	RPI, RH3	--
Untouched headers	--	--	--

Table 8: 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)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from source (6LBR) to destination (IPv6).

In 6LBR the RH3 is added, it is modified at each intermediate 6LR (6LR_1 and so on) and it is fully consumed in the last 6LR (6LR_n), but left there. As the RPI is added, then the IPv6 node which does not understand the RPI, will ignore it (following RFC8200), thus encapsulation is not necessary.

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

Header	6LBR	6LR_i i=(1,..,n-1)	6LR_n	IPv6 dst node
Inserted headers	RPI, RH3	--	--	--
Removed headers	--	--		--
Re-added headers	--	--	--	--
Modified headers	--	RPI, RH3	RPI, RH3(consumed)	--
Untouched headers	--	--	--	RPI, RH3 (both ignored)

Figure 15: 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) --> 6LR_1 --> 6LR_i --> root (6LBR)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet goes through from source (IPv6) to destination (6LBR). For example, 6LR_1 (i=1) is the router that receives the packets from the IPv6 node.

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

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

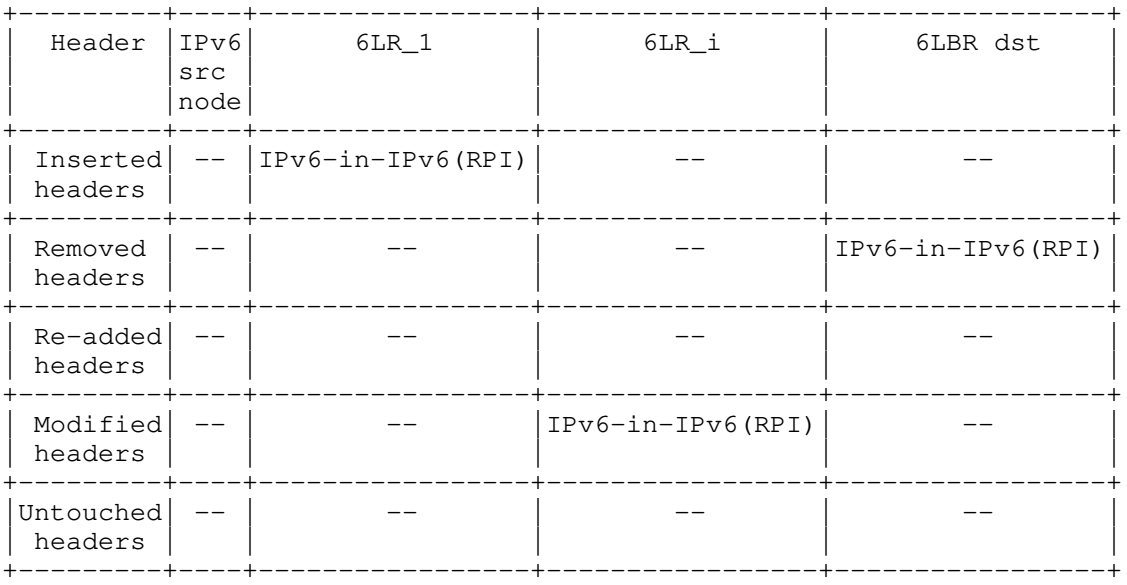


Figure 16: Non-SM: Summary of the use of headers from RUL to root

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

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

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from source (6LN) to 6LBR.

This case is identical to storing-mode case.

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 Table 9 summarizes what headers are needed for this use case.

Header	6LN src	6LR_i	6LBR	Internet dst
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	RPI	RPI (Ignored)

Table 9: Non-SM: Summary of the use of headers from RAL to Internet

8.2.2. Non-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) --> Node B --> Node D --> Node F

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet goes through from 6LBR to destination(6LN).

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 Table 10 summarizes what headers are needed for this use case.

Header	Internet dst	6LBR	6LR_i	6LN src
Inserted headers	--	IPv6-in-IPv6 (RH3,RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3,RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3,RPI)	--
Untouched headers	--	--	--	--

Table 10: 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:

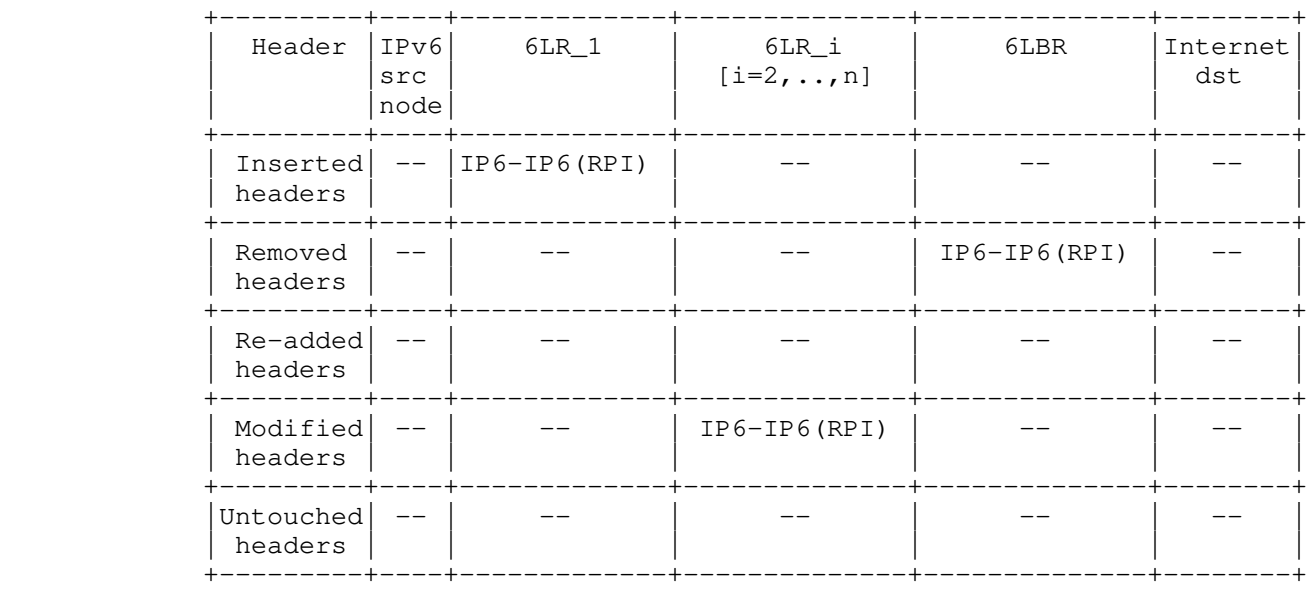
RUL (IPv6) --> 6LR_1 --> 6LR_i -->root (6LBR) --> Internet

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet goes through from source(IPv6) to 6LBR. e.g 6LR_1 (i=1).

In this case the flow label is recommended to be zero in the IPv6 node. As RPL headers are added in the IPv6 node packet, the first 6LR (6LR_1) will add a RPI header 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 17 shows the table that summarizes what headers are needed for this use case.



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

Figure 17: 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:

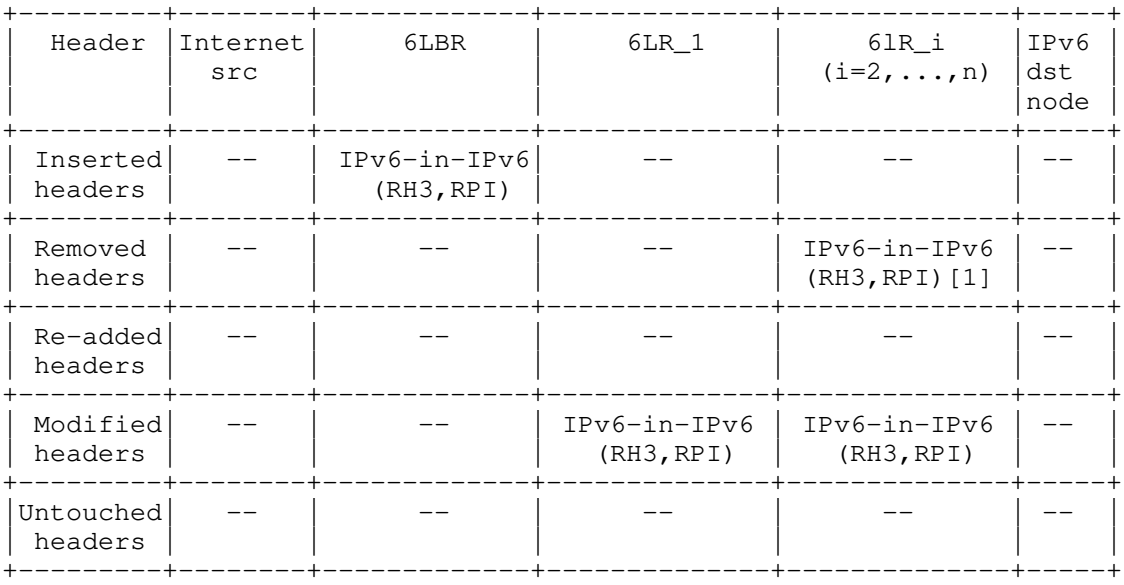
Internet --> root (6LBR) --> 6LR_i --> RUL (IPv6)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet goes through from 6LBR to RUL (IPv6).

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 an 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 18 shows the table that summarizes what headers are needed for this use case.



Header	Internet src	6LBR	6LR_1	6lR_i (i=2,...,n)	IPv6 dst node
Inserted headers	--	IPv6-in-IPv6 (RH3,RPI)	--	--	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3,RPI)[1]	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3,RPI)	IPv6-in-IPv6 (RH3,RPI)	--
Untouched headers	--	--	--	--	--

Figure 18: Non-SM: Summary of the use of headers from Internet to RUL
[1] The last 6LR before the IPv6 node.

8.3. Non-SM: Interaction between Leafs

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:

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

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

6LR_ia are the intermediate routers from source to the root In this case, 1 <= ia <= n, n is the number of routers (6LR) that the packet goes through from 6LN to the root.

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

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

The originating node must put the RPI into an IPv6-in-IPv6 header addressed to the root, so that the 6LBR can remove that header. If it does not, then additional resources are wasted on the way down to carry the useless RPI option.

The 6LBR will need to insert an RH3 header, which requires that it add an IPv6-in-IPv6 header. It should be able to remove the RPI, as it was contained in an IPv6-in-IPv6 header addressed to it. Otherwise, there may be a RPI header buried inside the inner IP header, which should get ignored.

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 19 shows the table that summarizes what headers are needed for this use case.

Header	6LN src	6LR_ia	6LBR	6LR_id	6LN dst
Inserted headers	IPv6-in-IPv6 (RPI1)		IPv6-in-IPv6 (RH3-> 6LN, RPI2)	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI1)	--	IPv6-in-IPv6 (RH3, RPI2)
Re-added headers	--	--	--	--	--
Modified headers	--	IP6-in-IP6 (RPI1)	--	IP6-in-IP6 (RPI2)	--
Untouched headers	--	--	--	--	--

Figure 19: Non-SM: Summary of the use of headers for RAL to RAL. IP6-in-IP6 refers to IPv6-in-IPv6.

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

In this case the flow comprises:

6LN --> 6LR_ia --> root (6LBR) --> 6LR_id --> not-RPL-aware (IPv6)

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

6LR_ia are the intermediate routers from source to the root In this case, 1 <= ia <= n, n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LRs).

As in the previous case, the 6LN will insert a RPI (RPI_1) 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 6LR_id.

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

Header	6LN src	6LR_ia	6LBR	6LR_id	6LR_m	IPv6 dst node
Inserted headers	IP6-IP6 (RPI1)		IP6-IP6 (RH3, RPI2)	--	--	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Re-added headers	--	--	--	--	--	--
Modified headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)		--
Untouched headers	--	--	--	--	--	--

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

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

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id --> 6LN

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

6LR_ia are the intermediate routers from source to the root. In this case, 1 <= ia <= n, n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

This scenario is mostly identical to the previous one. The RPI 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 it's own IPv6-in-IPv6 header containing an RH3 header and an RPI (RPI_2).

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

Header	IPv6 src node	6LR_1	6LR_ia	6LBR	6LR_id	6LN dst
Inserted headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--		--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Re-added headers	--		--	--	--	--
Modified headers	--		IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--		--	--	--	--

Figure 21: 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:

not-RPL-aware 6LN (IPv6 src) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id --> not-RPL-aware (IPv6 dst)

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

6LR_ia are the intermediate routers from source to the root. In this case, 1 <= ia <= n, n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

This scenario is the combination of the previous two cases.

The Figure 22 shows the table that summarizes what headers are needed for this use case.

Header	IPv6 src node	6LR_1	6LR_ia	6LBR	6LR_id	6LR_m	IPv6 dst node
Inserted headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Re-added headers	--	--	--	--	--	--	--
Modified headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--
Untouched headers	--	--	--	--	--	--	--

Figure 22: Non-SM: Summary of the use of headers from RUL to RUL

9. Operational Considerations of supporting not-RPL-aware-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 hosts that are pre-RFC8200, or simply intolerant. Those hosts will drop packets that continue to have RPL artifacts in them. In general, such hosts 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 be create the RH3 header), and therefore knows what the 6LR prior to the leaf. For example, in Figure 5, node E is the 6LR prior to the leaf node G, or node C is the 6LR prior to the 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 either 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 lot 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 value of 0x23.

During bootstrapping the node gets the DIO with the information of RPL 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 RPL Option Type. Thus, the DIO is sent with a flag indicating the new RPI value.

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 it is triggered when the DIO is sent with the flag indicating the new RPI value. Namely, it remains at 0x63 until it is sure that the network is capable of 0x23, then it abruptly change to 0x23. This options allows to send packets to not-RPL nodes, which should ignore the option and continue processing the packets.

In case that a node join to a network that only process 0x63, it would produce a flag day as was mentioned previously. Indicating the new RPI in the DODAG Configuration Option Flag is a way to avoid the flag day in a network. It is recommended that a network process both options to enable interoperability.

11. IANA Considerations

This document updates the registration made in [RFC6553] Destination Options and Hop-by-Hop Options registry from 0x63 to 0x23 as shown in Figure 23.

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 23: Option Type in RPL Option.(*)represents this document

DODAG Configuration option is updated as follows (Figure 24):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 24: DODAG Configuration Option Flag to indicate the RPI-flag-day.

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, they could still have a significant impact, particularly if 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-secure-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 section of

[RFC2473], it was suggested that tunnel entry and exit points can be secured, via "Use IPsec". 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 RFC8183 compression (compression must occur before encryption), and RFC8183 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, "Use 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 BCP38 origin filtering at the entry and exit of the LLN provides a similar level amount of security without all the scaling and trust problems of using IPsec as RFC2473 suggested. 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 (to disguise the origin of traffic and attack other nodes) with an IPv6-in-IPv6 header to add the needed RH3 header. As such, the attacker's RH3 header will not be seen by the network until it reaches the end host, which will decapsulate it. An end-host should be suspicious about a RH3 header which has additional hops which have not yet been processed, and SHOULD ignore such a second RH3 header.

In addition, the LLN will likely use [RFC8138] to compress the IPv6-in-IPv6 and RH3 headers. As such, the compressor at the RPL-root will see the second RH3 header and MAY choose to discard the packet if the RH3 header has not been completely consumed. 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. It should just be noted that an incoming RH3 must be fully consumed, or very carefully inspected.

The RPI header, 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 instanceID, but a change of instanceID would permit an attacker to bypass such filtering. Like the RH3, a RPI header 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 header therefore will not be seen by the network. Upon reaching the destination node the RPI header has no further meaning and is just skipped; the presence of a second RPI header will have no meaning to the end node as the packet has already been identified as being at it's final destination.

The RH3 and RPI headers could be abused by an attacker inside of the network to route packets on non-obvious ways, perhaps eluding observation. This usage is in fact part 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, a 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 by the Stand.ICT 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): Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, Rahul Jadhav, Matthias Kovatsch, Peter van der Stok, Xavier Vilajosana 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-12 (work in progress), April 2019.

[I-D.ietf-6lo-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-11 (work in progress), February 2019.

[I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.

[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-19 (work in progress), March 2019.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-22 (work in progress), June 2019.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-09 (work in progress), July 2018.

[I-D.thubert-roll-unaware-leaves]
Thubert, P., "Routing for RPL Leaves", draft-thubert-roll-unaware-leaves-07 (work in progress), April 2019.

[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>.

[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
Aalto University
Otaniemi
Espoo 02150
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
Village d'Entreprises Green Side 400, Avenue de Roumanille
Batiment T3, Biot - Sophia Antipolis 06410
France

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/1id-abstracts.html

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

Copyright and License Notice

Table of Contents

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 inconsistence. 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:

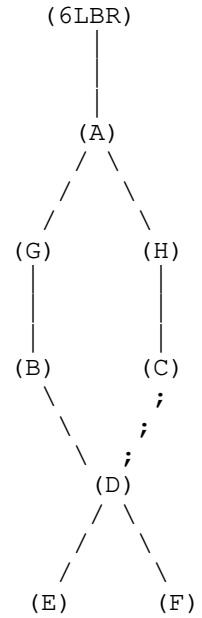


Figure 1: Sample Topology

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-childs, resulting in stale routing entries on behalf of the sub-childs 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-childs, (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: May 3, 2017

S. Anamalamudi
M. Zhang
AR. Sangi
Huawei Technologies
C. Perkins
Futurewei
S.V.R.Anand
Indian Institute of Science

October 30, 2016

Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs) draft-satish-roll-aodv-rpl-02

Abstract

Route discovery for symmetric and asymmetric Point-to-Point (P2P) traffic flows is a desirable feature in Low power and Lossy Networks (LLNs). For that purpose, this document specifies a reactive P2P route discovery mechanism for hop-by-hop routing (storing mode) based on Ad Hoc On-demand Distance Vector Routing (AODV) based RPL protocol. Two separate Instances are used to construct directional paths in case some of the links between source and target node are asymmetric.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2017.

Copyright Notice

Table of Contents

1. Introduction

RPL[RFC6550], the IPv6 distance vector routing protocol for Low-power and Lossy Networks (LLNs), is designed to support multiple traffic flows through a root-based Destination-Oriented Directed Acyclic Graph (DODAG). For traffic flows between routers within the DODAG (i.e., Point-to-Point (P2P) traffic), this means that data packets either have to traverse the root in non-storing mode (source routing), or traverse a common ancestor in storing mode (hop-by-hop routing). Such P2P traffic is thereby likely to flow along sub-optimal routes and may suffer severe traffic congestion near the DAG root [RFC6997], [RFC6998].

To discover optimal paths for P2P traffic flows in RPL, P2P-RPL [RFC6997] specifies a temporary DODAG where the source acts as temporary root. The source initiates "P2P Route Discovery mode (P2P-RDO)" with an address vector for both non-storing mode (H=0) and storing mode (H=1). Subsequently, each intermediate router adds its IP address and multicasts the P2P-RDO message, until the message

reaches the target node (TargNode). TargNode sends the "Discovery Reply" option. P2P-RPL is efficient for source routing, but much less efficient for hop-by-hop routing due to the extra address vector overhead. In fact, when the P2P-RDO message is being multicast from the source hop-by-hop, receiving nodes are able to determine a next hop towards the source in symmetric links. When TargNode subsequently replies to the source along the established forward route, receiving nodes can determine the next hop towards TargNode. In other words, it is efficient to use only routing tables for P2P-RDO message instead of "Address vector" for hop-by-hop routes (H=1) in symmetric links.

RPL and P2P-RPL both specify the use of a single DODAG in networks of symmetric links. But, application-specific routing requirements that are defined in IETF ROLL Working Group [RFC5548], [RFC5673], [RFC5826] and [RFC5867] may need routing metrics and constraints enabling use of asymmetric bidirectional links. For this purpose, [I-D.thubert-roll-asymlink] describes bidirectional asymmetric links for RPL [RFC6550] with Paired DODAGs, for which the DAG root (DODAGID) is common for two Instances. This can satisfy application-specific routing requirements for bidirectional asymmetric links in base RPL [RFC6550]. P2P-RPL for Paired DODAGs, on the other hand, requires two DAG roots: one for the source and another for the target node due to temporary DODAG formation. For networks composed of bidirectional asymmetric links (see Section 4), AODV-RPL specifies P2P route discovery, utilizing RPL with a new MoP. AODV-RPL makes use of two multicast messages to discover possibly asymmetric routes. AODV-RPL eliminates the need for address vector control overhead, significantly reducing the control packet size which is important for Constrained LLN networks. Both discovered routes meet the application specific metrics and constraints that are defined in the Objective Function for each Instance [RFC6552].

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 [RFC2119]. Additionally, this document uses the following terms:

AODV
: Ad Hoc On-demand Distance Vector Routing[RFC3561].

AODV-Instance
: Either the RREQ-Instance or RREP-Instance

Bi-directional Asymmetric Link

A link that can be used in both directions but with different link characteristics (see [I-D.thubert-roll-asymlink]).

DODAG RREQ-Instance (or simply RREQ-Instance)
AODV Instance built using the RREQ option; used for control transmission from OrigNode to TargNode, thus enabling data transmission from TargNode to OrigNode.

DODAG RREP-Instance (or simply RREP-Instance)
AODV Instance built using the RREP option; used for control transmission from TargNode to OrigNode thus enabling data transmission from OrigNode to TargNode.

downstream
Routing along the direction from OrigNode to TargNode.

hop-by-hop routing
Routing when each node stores routing information about the next hop.

OrigNode
The IPv6 router (Originating Node) initiating the AODV-RPL route discovery to obtain a route to TargNode.

Paired DODAGs
Two DODAGs for a single application.

P2P
Point-to-Point -- in other words, not constrained to traverse a common ancestor.

RREP message
An AODV-RPL MoP DIO message containing the RREP option

RREQ message
An AODV-RPL MoP DIO message containing the RREQ option

source routing
The mechanism by which the source supplies the complete route towards the target node along with each data packet [RFC6997].

TargNode
The IPv6 router (Target Node) for which OrigNode requires a route and initiates Route Discovery within the LLN network.

upstream
Routing along the direction from TargNode to OrigNode.

3. Overview of AODV-RPL

With AODV-RPL, routes from OrigNode to TargNode within the LLN network established are "on-demand". In other words, the route discovery mechanism in AODV-RPL is invoked reactively when OrigNode has data for delivery to the TargNode but existing routes do not satisfy the application's requirements. The routes discovered by AODV-RPL are point-to-point; in other words the routes are not constrained to traverse a common ancestor. Unlike base RPL [RFC6550] and P2P-RPL [RFC6997], AODV-RPL can enable asymmetric communication paths in networks with bidirectional asymmetric links. For this purpose, AODV-RPL enables discovery of two routes: namely, one from OrigNode to TargNode, and another from TargNode to OrigNode. When possible, AODV-RPL also enables symmetric routing along Paired DODAGs (see Section 4).

4. AODV-RPL Mode of Operation (MoP)

In AODV-RPL, route discovery is initiated by forming a temporary DAG rooted at the OrigNode. Paired DODAGs (Instances) are constructed according to a new AODV-RPL Mode of Operation (MoP) during route formation between the OrigNode and TargNode. The RREQ-Instance is formed by route control messages from OrigNode to TargNode whereas the RREP-Instance is formed by route control messages from TargNode to OrigNode (as shown in Figure 2). Intermediate routers join the Paired DODAGs based on the rank as calculated from the DIO message. Henceforth in this document, the RREQ-Instance message means the AODV-RPL DIO message from OrigNode to TargNode, containing the RREQ option. Similarly, the RREP-Instance means the AODV-RPL DIO message from TargNode to OrigNode, containing the RREP option. Subsequently, the RREQ-Instance is used for data transmission from TargNode to OrigNode and RREP-Instance is used for Data transmission from OrigNode to TargNode.

The AODV-RPL Mode of Operation defines a new bit, the Symmetric bit ('S'), which is added to the base DIO message as illustrated in Figure 1. OrigNode sets the the 'S' bit to 1 in the RREQ-Instance message when initiating route discovery.

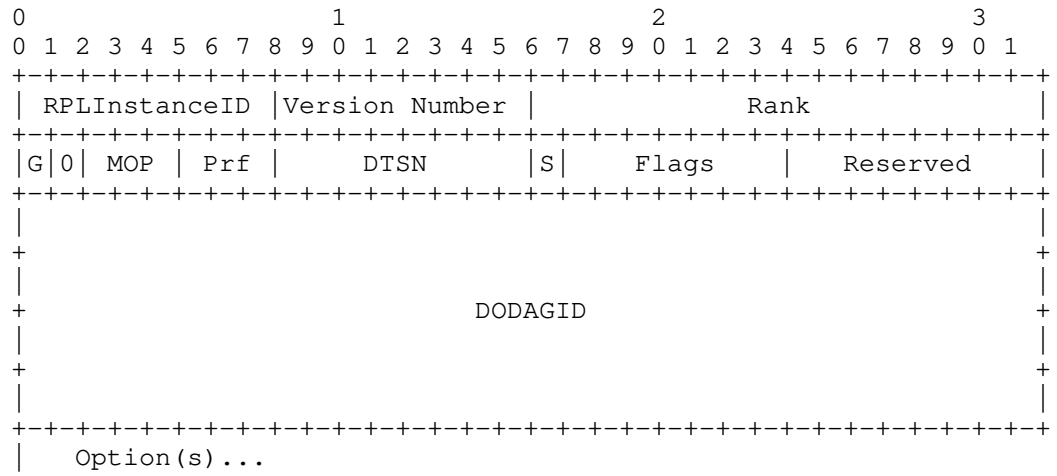


Figure 1: DIO modification to support asymmetric route discovery

A device originating a AODV-RPL message supplies the following information in the DIO header of the message:

'S' bit

Symmetric bit in the DIO base object

MOP

MOP operation in the DIO object MUST be set to "5(TBD1)" for AODV-RPL DIO messages

RPLInstanceID

RPLInstanceID in the DIO object MUST be the InstanceID of AODV-Instance.

DODAGID

DODAGID in the DIO object MUST be the IPv6 address of the device that initiates the RREQ-Instance.

Rank

Rank in the DIO object MUST be the the rank of the AODV-Instance

Metric Container Options

AODV-Instance messages MAY carry one or more Metric Container options to indicate the relevant routing metrics

The 'S' bit is set to mean that the route is symmetric. If the RREQ-Instance arrives over an interface that is known to be symmetric, and the 'S' bit is set to 1, then it remains set at 1, as illustrated in Figure 2.

In this figure:
S := OrigNode; R := Intermediate nodes; D := TargNode

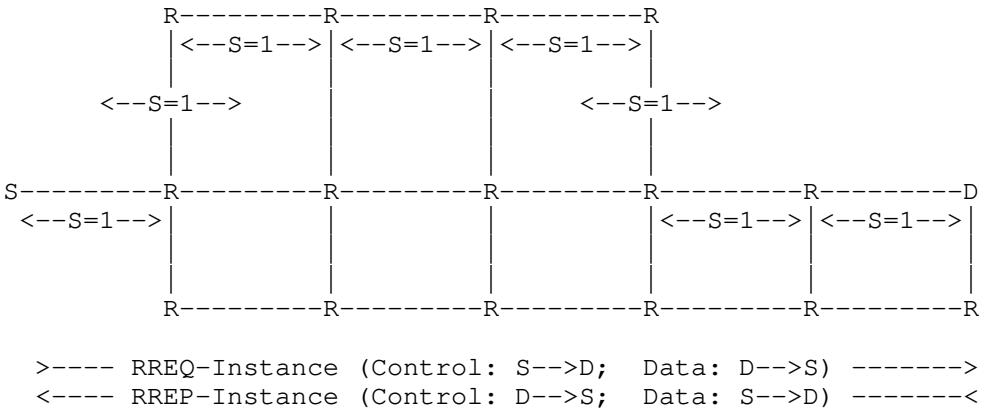


Figure 2: AODV-RPL with Symmetric Paired Instances

If the RREQ-Instance arrives over an interface that is not known to be symmetric, or is known to be asymmetric, the 'S' bit is set to be 0. Moreover, if the 'S' bit arrives already set to be '0', it is set to be '0' on retransmission (Figure 3). Based on the 'S' bit received in RREQ-Instance, the TargNode decides whether or not the route is symmetric before transmitting the RREP-Instance message upstream towards the OrigNode. The metric used to determine symmetry (i.e., set the "S" bit to be "1" (Symmetric) or "0" (asymmetric)) is not specified in this document.

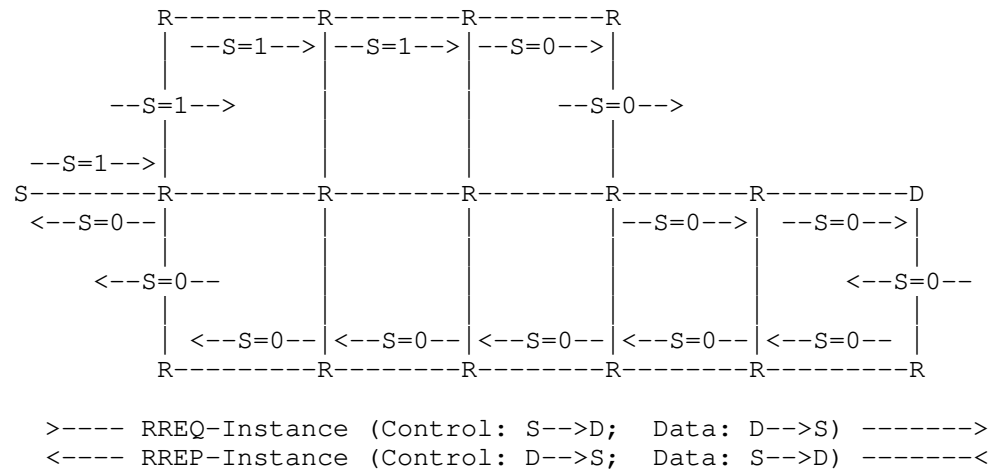


Figure 3: AODV-RPL with Asymmetric Paired Instances

5. RREQ Message

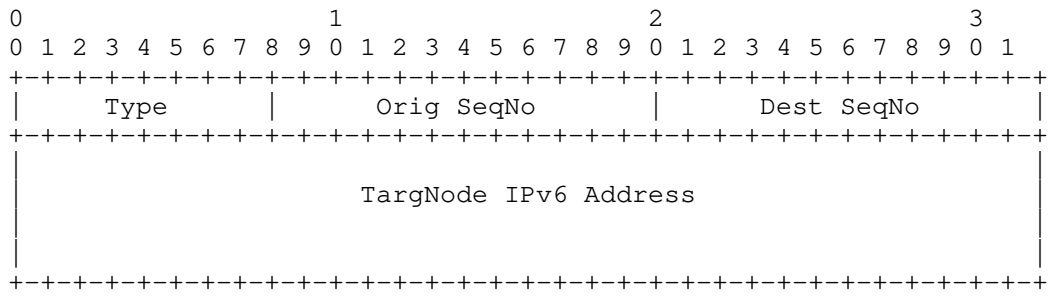


Figure 4: DIO RREQ option format for AODV-RPL MoP

OrigNode supplies the following information in the RREQ option of the RREQ-Instance message:

Type

The type of the RREQ option (see Section 9.2)

Orig SeqNo

Sequence Number of OrigNode. MUST be chosen to be an odd number.

Dest SeqNo

If nonzero, the last known Sequence Number for TargNode for which a route is desired.

TargNode IPv6 Address

IPv6 address of the TargNode that receives RREQ-Instance message. This address MUST be in the RREQ option (see Figure 4) of AODV-RPL.

In order to establish the upstream route from TargNode to OrigNode, OrigNode multicasts the RREQ-Instance message (see Figure 4) to its one-hop neighbours. In order to enable intermediate nodes R_i to associate a future RREP message to an incoming RREQ message, the RREQ message MUST assign an odd number for the OrigNode's Dest SeqNo which is included in the RREQ message.

Each intermediate node R_i computes the rank for RREQ-Instance and creates a routing table entry for the upstream route towards the source if the routing metrics/constraints are satisfied. For this purpose R_i must use the asymmetric link metric measured in the upstream direction, from R_i to its upstream neighbor that multicasted the RREQ-Instance message.

When an intermediate node R_i receives a RREQ message in storing mode, it MUST store the OrigNode's Dest Seqno along with the other routing information needed to establish the route back to the OrigNode. This will enable R_i to determine that a future RREP message (containing a paired Dest Seqno for the TargNode) must be transmitted back to the OrigNode's IP address.

If the paths to and from TargNode are not known, the intermediate node multicasts the RREQ-Instance message with updated rank to its next-hop neighbors until the message reaches TargNode (Figure 2). Based on the 'S' bit in the received RREQ message, the TargNode will decide whether to unicast or multicast the RREP message back to OrigNode.

As described in Section 7, in certain circumstances R_i MAY unicast a Gratuitous RREP towards OrigNode, thereby helping to minimize multicast overhead during the Route Discovery process.

6. RREP Message

The TargNode supplies the following information in the RREP message:

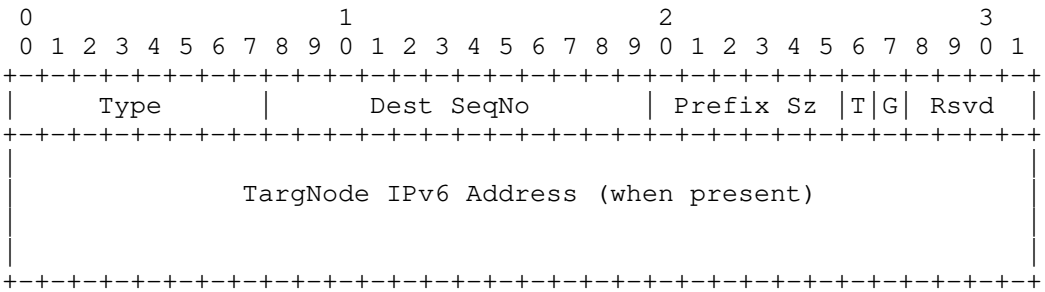


Figure 5: DIO RREP option format for AODV-RPL MoP

Type

The type of the RREP option (see Section 9.2)

Dest SeqNo

The Sequence Number for the TargNode for which a route is established.

Prefix Sz

The size of the prefix which the route to the TargNode is available. This allows routing to other nodes on the same subnet as the TargNode.

'T' bit

'T' is set to true to indicate that the TargNode IPv6 Address field is present

'G' bit

(see Section 7)

TargNode IPv6 Address (when present)

IPv6 address of the TargNode that receives RREP-Instance message.

In order to reduce the need for the TargNode IPv6 Address to be included with the RREP message, the Dest SeqNo of the RREP is paired, whenever possible, with the Dest SeqNo from the RREQ message, which is always an odd number. The pairing is accomplished by adding one to the Dest Seqno from the RREQ message and using that, whenever possible, as the Dest Seqno for the RREP message. If this is not possible (for instance because the incremented sequence number is

still a valid sequence number for another route to the TargNode from an earlier Route Discovery operation), then the 'T' bit is set and an odd number is chosen for the TargNode's Dest SeqNo.

The OrigNode IP address is available as the DODAGID in the DIO base message (see Figure 1). When TargNode receives a RREQ message with the 'S' bit set to 1 (as illustrated in Figure 2), it unicasts the RREP message with the 'S' bit set to 1. In this case, route control messages and application data between OrigNode and TargNode for both RREQ-Instance and RREP-Instance are transmitted along symmetric links.

When (as illustrated in Figure 3) the TargNode receives RREQ message with the 'S' bit set to 0, it also multicasts the RREP message with the 'S' bit set to 0. Intermediate nodes create a routing table entry for the path towards the TargNode while processing the RREP message to OrigNode. Once OrigNode receives the RREP message, it starts transmitting application data to TargNode along the path as discovered through RREP messages. Similarly, application data from TargNode to OrigNode is transmitted through the path that is discovered from RREQ message.

7. Gratuitous RREP

Under some circumstances, an Intermediate Node that receives a RREQ message MAY transmit a "Gratuitous" RREP message back to OrigNode instead of continuing to multicast the RREQ message towards TargNode. For these circumstances, the 'G' bit of the RREP option is provided to distinguish the Gratuitous RREP sent by the Intermediate node from the RREP sent by TargNode.

When an Intermediate node R receives a RREQ message and has recent information about the cost of an upstream route from TargNode to R, then R MAY unicast the Gratuitous RREP (GRREP) message to OrigNode. R determines whether its information is sufficiently recent by comparing the value it has stored for the Sequence Number of TargNode against the DestSeqno in the incoming RREQ message. R also must have information about the metric information of the upstream route from TargNode. The GRREP message MUST have PrefixSz == 0 and the 'G' bit set to 1. R SHOULD also unicast the RREQ message to TargNode, to make sure that TargNode will have a route to OrigNode.

8. Operation of Trickle Timer

The trickle timer operation to control RREQ-Instance/RREP-Instance multicast is similar to that in P2P-RPL [RFC6997].

9. IANA Considerations

9.1. New Mode of Operation: AODV-RPL

IANA is required to assign a new Mode of Operation, named "AODV-RPL" for Point-to-Point(P2P) hop-by-hop routing under the RPL registry. The value of TBD1 is assigned from the "Mode of Operation" space [RFC6550].

Value	Description	Reference
TBD1 (5)	AODV-RPL	This document

Figure 6: Mode of Operation

9.2. AODV-RPL Options: RREQ and RREP

Two entries are required for new AODV-RPL options "RREQ-Instance" and "RREQ-Instance", with values of TBD2 (0x0A) and TBD3 (0x0B) from the "RPL Control Message Options" space [RFC6550].

Value	Meaning	Reference
TBD2 (0x0A)	RREQ Option	This document
TBD3 (0x0B)	RREP Option	This document

Figure 7: AODV-RPL Options

10. Security Considerations

This document does not introduce additional security issues compared to base RPL. For general RPL security considerations, see [RFC6550].

11. References

11.1. Normative References

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

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <http://www.rfc-editor.org/info/rfc3561>.

[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, DOI 10.17487/RFC5548, May 2009, <http://www.rfc-editor.org/info/rfc5548>.

[RFC5673] Pister, K., Ed., Thubert, P., Ed., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, DOI 10.17487/RFC5673, October 2009, <http://www.rfc-editor.org/info/rfc5673>.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, DOI 10.17487/RFC5826, April 2010, <http://www.rfc-editor.org/info/rfc5826>.

[RFC5867] Martocci, J., Ed., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, DOI 10.17487/RFC5867, June 2010, <http://www.rfc-editor.org/info/rfc5867>.

[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>.

[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, <http://www.rfc-editor.org/info/rfc6997>.

[RFC6998] Goyal, M., Ed., Baccelli, E., Brandt, A., and J. Martocci, "A Mechanism to Measure the Routing Metrics along a Point-to-Point Route in a Low-Power and Lossy Network", RFC 6998, DOI 10.17487/RFC6998, August 2013, <http://www.rfc-editor.org/info/rfc6998>.

11.2. Informative References

[I-D.thubert-roll-asymlink]
Thubert, P., "RPL adaptation for asymmetrical links", draft-thubert-roll-asymlink-02 (work in progress), December 2011.

Authors' Addresses

Satish Anamalamudi
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
China

Email: satishnaidu80@gmail.com

Mingui Zhang
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
China

Email: zhangmingui@huawei.com

Abdur Rashid Sangi
Huawei Technologies
No.156 Beiqing Rd. Haidian District
Beijing 100095
P.R. China

Email: rashid.sangi@huawei.com

Charles E. Perkins
Futurewei
2330 Central Expressway
Santa Clara 95050
Unites States

Email: charliep@computer.org

S.V.R Anand
Indian Institute of Science
Bangalore 560012
India

Email: anand@ece.iisc.ernet.in

ROLL
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2017

S. Anamalamudi
M. Zhang
AR. Sangi
Huawei Technologies
C. Perkins
Futurewei
S.V.R.Anand
Indian Institute of Science

November 15, 2016

Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs)
draft-satish-roll-aodv-rpl-03

Abstract

Route discovery for symmetric and asymmetric Point-to-Point (P2P) traffic flows is a desirable feature in Low power and Lossy Networks (LLNs). For that purpose, this document specifies a reactive P2P route discovery mechanism for hop-by-hop routing (storing mode) based on Ad Hoc On-demand Distance Vector Routing (AODV) based RPL protocol. Two separate Instances are used to construct directional paths in case some of the links between source and target node are asymmetric.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 19, 2017.

Copyright Notice

Table of Contents

1. Introduction

RPL[RFC6550], the IPv6 distance vector routing protocol for Low-power and Lossy Networks (LLNs), is designed to support multiple traffic flows through a root-based Destination-Oriented Directed Acyclic Graph (DODAG). For traffic flows between routers within the DODAG (i.e., Point-to-Point (P2P) traffic), this means that data packets either have to traverse the root in non-storing mode (source routing), or traverse a common ancestor in storing mode (hop-by-hop routing). Such P2P traffic is thereby likely to flow along sub-optimal routes and may suffer severe traffic congestion near the DAG root [RFC6997], [RFC6998].

To discover optimal paths for P2P traffic flows in RPL, P2P-RPL [RFC6997] specifies a temporary DODAG where the source acts as temporary root. The source initiates "P2P Route Discovery mode (P2P-RDO)" with an address vector for both non-storing mode (H=0) and storing mode (H=1). Subsequently, each intermediate router adds its IP address and multicasts the P2P-RDO message, until the message

reaches the target node (TargNode). TargNode sends the "Discovery Reply" option. P2P-RPL is efficient for source routing, but much less efficient for hop-by-hop routing due to the extra address vector overhead. In fact, when the P2P-RDO message is being multicast from the source hop-by-hop, receiving nodes are able to determine a next hop towards the source in symmetric links. When TargNode subsequently replies to the source along the established forward route, receiving nodes can determine the next hop towards TargNode. In other words, it is efficient to use only routing tables for P2P-RDO message instead of "Address vector" for hop-by-hop routes (H=1) in symmetric links.

RPL and P2P-RPL both specify the use of a single DODAG in networks of symmetric links. But, application-specific routing requirements that are defined in IETF ROLL Working Group [RFC5548], [RFC5673], [RFC5826] and [RFC5867] may need routing metrics and constraints enabling use of asymmetric bidirectional links. For this purpose, [I-D.thubert-roll-asymlink] describes bidirectional asymmetric links for RPL [RFC6550] with Paired DODAGs, for which the DAG root (DODAGID) is common for two Instances. This can satisfy application-specific routing requirements for bidirectional asymmetric links in base RPL [RFC6550]. P2P-RPL for Paired DODAGs, on the other hand, requires two DAG roots: one for the source and another for the target node due to temporary DODAG formation. For networks composed of bidirectional asymmetric links (see Section 4), AODV-RPL specifies P2P route discovery, utilizing RPL with a new MoP. AODV-RPL makes use of two multicast messages to discover possibly asymmetric routes. AODV-RPL eliminates the need for address vector control overhead, significantly reducing the control packet size which is important for Constrained LLN networks. Both discovered routes meet the application specific metrics and constraints that are defined in the Objective Function for each Instance [RFC6552].

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 [RFC2119]. Additionally, this document uses the following terms:

AODV
: Ad Hoc On-demand Distance Vector Routing[RFC3561].

AODV-Instance
: Either the RREQ-Instance or RREP-Instance

Bi-directional Asymmetric Link

A link that can be used in both directions but with different link characteristics (see [I-D.thubert-roll-asymlink]).

DODAG RREQ-Instance (or simply RREQ-Instance)
AODV Instance built using the RREQ option; used for control transmission from OrigNode to TargNode, thus enabling data transmission from TargNode to OrigNode.

DODAG RREP-Instance (or simply RREP-Instance)
AODV Instance built using the RREP option; used for control transmission from TargNode to OrigNode thus enabling data transmission from OrigNode to TargNode.

downstream
Routing along the direction from OrigNode to TargNode.

hop-by-hop routing
Routing when each node stores routing information about the next hop.

OrigNode
The IPv6 router (Originating Node) initiating the AODV-RPL route discovery to obtain a route to TargNode.

Paired DODAGs
Two DODAGs for a single application.

P2P
Point-to-Point -- in other words, not constrained to traverse a common ancestor.

RREQ message
An AODV-RPL MoP DIO message containing the RREQ option. The InstanceID in DIO object of RREQ option MUST be always an odd number.

RREP message
An AODV-RPL MoP DIO message containing the RREP option. The InstanceID in DIO object of RREP option MUST be always an even number (InstanceID of RREQ-Instance+1).

source routing
The mechanism by which the source supplies the complete route towards the target node along with each data packet. [RFC6997].

TargNode
The IPv6 router (Target Node) for which OrigNode requires a route and initiates Route Discovery within the LLN network.

upstream
Routing along the direction from TargNode to OrigNode.

3. Overview of AODV-RPL

With AODV-RPL, routes from OrigNode to TargNode within the LLN network established are "on-demand". In other words, the route discovery mechanism in AODV-RPL is invoked reactively when OrigNode has data for delivery to the TargNode but existing routes do not satisfy the application's requirements. The routes discovered by AODV-RPL are point-to-point; in other words the routes are not constrained to traverse a common ancestor. Unlike base RPL [RFC6550] and P2P-RPL [RFC6997], AODV-RPL can enable asymmetric communication paths in networks with bidirectional asymmetric links. For this purpose, AODV-RPL enables discovery of two routes: namely, one from OrigNode to TargNode, and another from TargNode to OrigNode. When possible, AODV-RPL also enables symmetric routing along Paired DODAGs (see Section 4).

4. AODV-RPL Mode of Operation (MoP)

In AODV-RPL, route discovery is initiated by forming a temporary DAG rooted at the OrigNode. Paired DODAGs (Instances) are constructed according to a new AODV-RPL Mode of Operation (MoP) during route formation between the OrigNode and TargNode. The RREQ-Instance is formed by route control messages from OrigNode to TargNode whereas the RREP-Instance is formed by route control messages from TargNode to OrigNode (as shown in Figure 2). Intermediate routers join the Paired DODAGs based on the rank as calculated from the DIO message. Henceforth in this document, the RREQ-Instance message means the AODV-RPL DIO message from OrigNode to TargNode, containing the RREQ option. Similarly, the RREP-Instance means the AODV-RPL DIO message from TargNode to OrigNode, containing the RREP option. Subsequently, the RREQ-Instance is used for data transmission from TargNode to OrigNode and RREP-Instance is used for Data transmission from OrigNode to TargNode.

The AODV-RPL Mode of Operation defines a new bit, the Symmetric bit ('S'), which is added to the base DIO message as illustrated in Figure 1. OrigNode sets the the 'S' bit to 1 in the RREQ-Instance message when initiating route discovery.

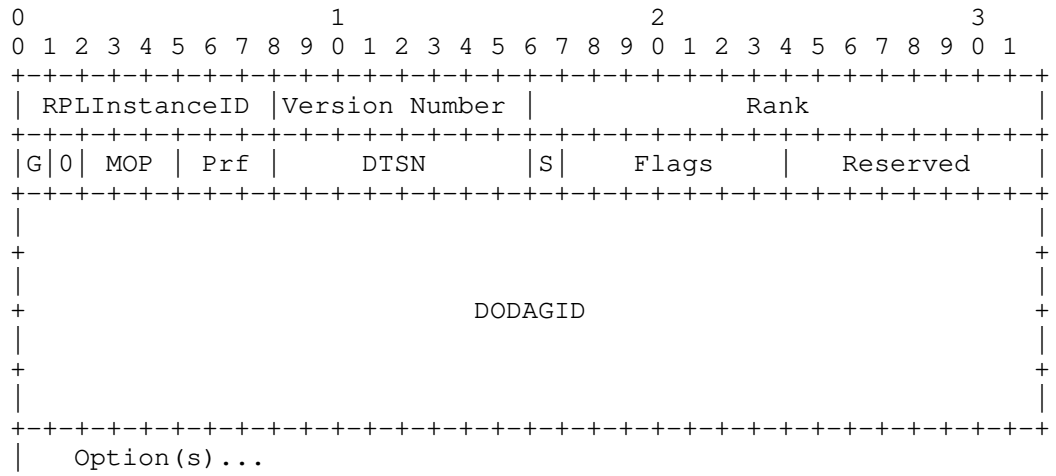


Figure 1: DIO modification to support asymmetric route discovery

A device originating a AODV-RPL message supplies the following information in the DIO header of the message:

'S' bit

Symmetric bit in the DIO base object

MOP

MOP operation in the DIO object MUST be set to "5(TBD1)" for AODV-RPL DIO messages

RPLInstanceID

RPLInstanceID in the DIO object MUST be the InstanceID of AODV-Instance(RREQ-Instance). The InstanceID for RREQ-Instance MUST be always an odd number.

DODAGID

For RREQ-Instance :

DODAGID in the DIO object MUST be the IPv6 address of the device that initiates the RREQ-Instance.

For RREP-Instance

DODAGID in the DIO object MUST be the IPv6 address of the device that initiates the RREP-Instance.

Rank

Rank in the DIO object MUST be the the rank of the AODV-Instance (RREQ-Instance).

Metric Container Options

AODV-Instance(RREQ-Instance) messages MAY carry one or more Metric Container options to indicate the relevant routing metrics.

The 'S' bit is set to mean that the route is symmetric. If the RREQ-Instance arrives over an interface that is known to be symmetric, and the 'S' bit is set to 1, then it remains set at 1, as illustrated in Figure 2.

In this figure:
S := OrigNode; R := Intermediate nodes; D := TargNode

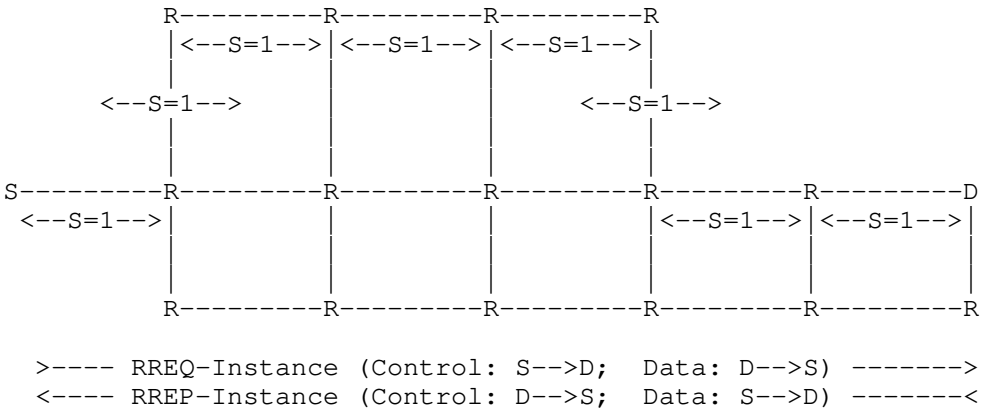


Figure 2: AODV-RPL with Symmetric Paired Instances

If the RREQ-Instance arrives over an interface that is not known to be symmetric, or is known to be asymmetric, the 'S' bit is set to be 0. Moreover, if the 'S' bit arrives already set to be '0', it is set to be '0' on retransmission (Figure 3). Based on the 'S' bit received in RREQ-Instance, the TargNode decides whether or not the route is symmetric before transmitting the RREP-Instance message upstream towards the OrigNode. The metric used to determine symmetry (i.e., set the "S" bit to be "1" (Symmetric) or "0" (asymmetric)) is not specified in this document.

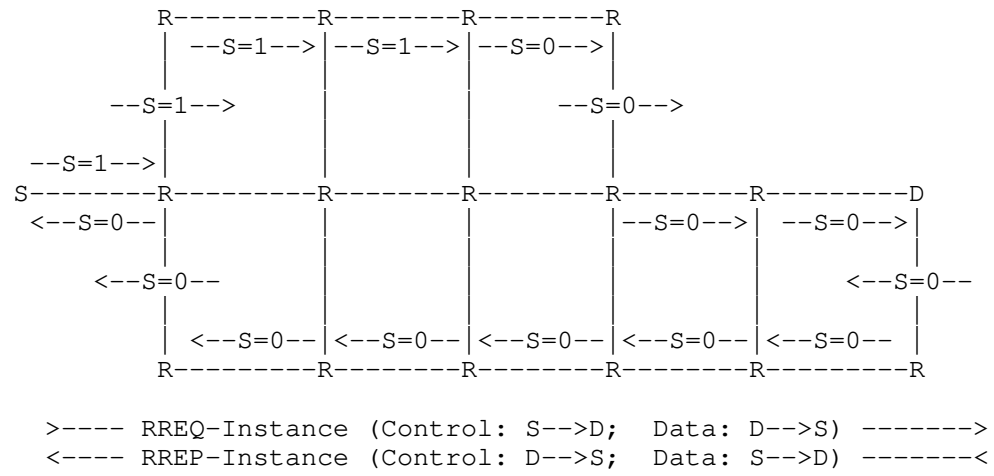


Figure 3: AODV-RPL with Asymmetric Paired Instances

5. RREQ Message

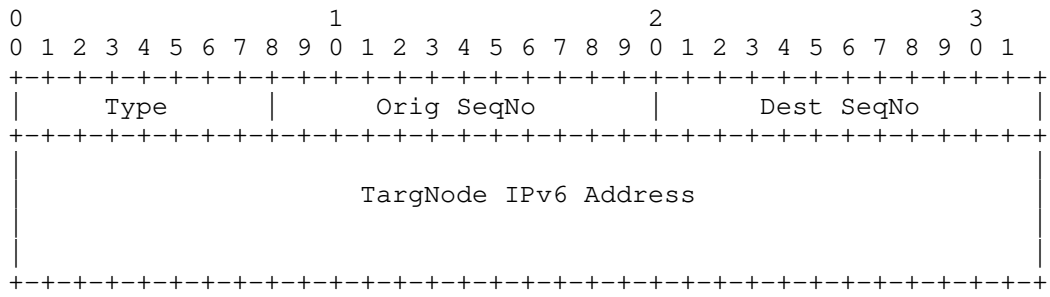


Figure 4: DIO RREQ option format for AODV-RPL MoP

OrigNode supplies the following information in the RREQ option of the RREQ-Instance message:

Type

The type of the RREQ option (see Section 9.2)

Orig SeqNo

Sequence Number of OrigNode.

Dest SeqNo

If nonzero, the last known Sequence Number for TargNode for which a route is desired.

TargNode IPv6 Address

IPv6 address of the TargNode that receives RREQ-Instance message. This address MUST be in the RREQ option (see Figure 4) of AODV-RPL.

In order to establish the upstream route from TargNode to OrigNode, OrigNode multicasts the RREQ-Instance message (see Figure 4) to its one-hop neighbours. In order to enable intermediate nodes R_i to associate a future RREP message to an incoming RREQ message, the InstanceID of RREQ-Instance MUST assign an odd number.

Each intermediate node R_i computes the rank for RREQ-Instance and creates a routing table entry for the upstream route towards the source if the routing metrics/constraints are satisfied. For this purpose R_i must use the asymmetric link metric measured in the upstream direction, from R_i to its upstream neighbor that multicasted the RREQ-Instance message.

When an intermediate node R_i receives a RREQ message in storing mode, it MUST store the OrigNode's InstanceID (RREQ-Instance) along with the other routing information needed to establish the route back to the OrigNode. This will enable R_i to determine that a future RREP message (containing a paired InstanceID for the TargNode) must be transmitted back to the OrigNode's IP address.

If the paths to and from TargNode are not known, the intermediate node multicasts the RREQ-Instance message with updated rank to its next-hop neighbors until the message reaches TargNode (Figure 2). Based on the 'S' bit in the received RREQ message, the TargNode will decide whether to unicast or multicast the RREP message back to OrigNode.

As described in Section 7, in certain circumstances R_i MAY unicast a Gratuitous RREP towards OrigNode, thereby helping to minimize multicast overhead during the Route Discovery process.

6. RREP Message

The TargNode supplies the following information in the RREP message:

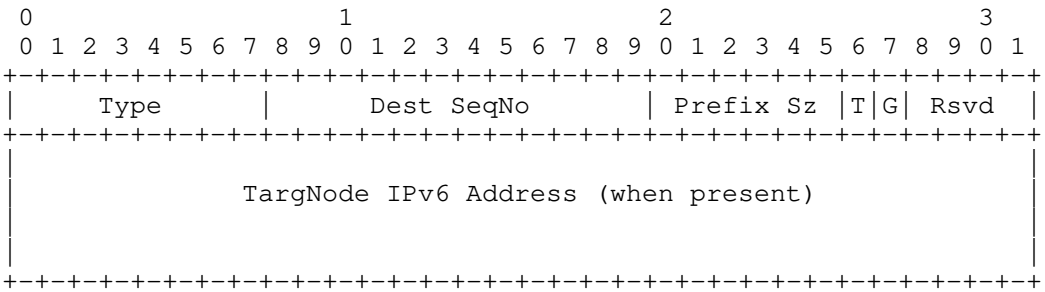


Figure 5: DIO RREP option format for AODV-RPL MoP

Type

The type of the RREP option (see Section 9.2)

Dest SeqNo

The Sequence Number for the TargNode for which a route is established.

Prefix Sz

The size of the prefix which the route to the TargNode is available. This allows routing to other nodes on the same subnet as the TargNode.

'T' bit

'T' is set to true to indicate that the TargNode IPv6 Address field is present

'G' bit

(see Section 7)

TargNode IPv6 Address (when present)

IPv6 address of the TargNode that receives RREP-Instance message.

In order to reduce the need for the TargNode IPv6 Address to be included with the RREP message, the InstanceID of the RREP-Instance is paired, whenever possible, with the InstanceID from the RREQ message, which is always an odd number. The pairing is accomplished by adding one to the InstanceID from the RREQ message and using that, whenever possible, as the InstanceID for the RREP message. If this is not possible (for instance because the incremented InstanceID is

still a valid InstanceID for another route to the TargNode from an earlier Route Discovery operation), then the 'T' bit is set and an odd number is chosen for the InstanceID of RREP from TargNode.

The OrigNode IP address for RREQ-Instance is available as the DODAGID in the DIO base message (see Figure 1). When TargNode receives a RREQ message with the 'S' bit set to 1 (as illustrated in Figure 2), it unicasts the RREP message with the 'S' bit set to 1. In this case, route control messages and application data between OrigNode and TargNode for both RREQ-Instance and RREP-Instance are transmitted along symmetric links. When the InstanceID of RREP-Instance is even number then the TargNode IPv6 Address is elided in RREP option. When the InstanceID of RREP-Instance is an odd number with "T" bit set to "1" then TargNode IPv6 Address is transmitted in RREP option.

When (as illustrated in Figure 3) the TargNode receives RREQ message with the 'S' bit set to 0, it also multicasts the RREP message with the 'S' bit set to 0. Intermediate nodes create a routing table entry for the path towards the TargNode while processing the RREP message to OrigNode. Once OrigNode receives the RREP message, it starts transmitting application data to TargNode along the path as discovered through RREP messages. Similarly, application data from TargNode to OrigNode is transmitted through the path that is discovered from RREQ message.

7. Gratuitous RREP

Under some circumstances, an Intermediate Node that receives a RREQ message MAY transmit a "Gratuitous" RREP message back to OrigNode instead of continuing to multicast the RREQ message towards TargNode. For these circumstances, the 'G' bit of the RREP option is provided to distinguish the Gratuitous RREP sent by the Intermediate node from the RREP sent by TargNode.

When an Intermediate node R receives a RREQ message and has recent information about the cost of an upstream route from TargNode to R, then R MAY unicast the Gratuitous RREP (GRREP) message to OrigNode. R determines whether its information is sufficiently recent by comparing the value it has stored for the Sequence Number of TargNode against the DestSeqno in the incoming RREQ message. R also must have information about the metric information of the upstream route from TargNode. The GRREP message MUST have PrefixSz == 0 and the 'G' bit set to 1. R SHOULD also unicast the RREQ message to TargNode, to make sure that TargNode will have a route to OrigNode.

8. Operation of Trickle Timer

The trickle timer operation to control RREQ-Instance/RREP-Instance multicast is similar to that in P2P-RPL [RFC6997].

9. IANA Considerations

9.1. New Mode of Operation: AODV-RPL

IANA is required to assign a new Mode of Operation, named "AODV-RPL" for Point-to-Point(P2P) hop-by-hop routing under the RPL registry. The value of TBD1 is assigned from the "Mode of Operation" space [RFC6550].

Value	Description	Reference
TBD1 (5)	AODV-RPL	This document

Figure 6: Mode of Operation

9.2. AODV-RPL Options: RREQ and RREP

Two entries are required for new AODV-RPL options "RREQ-Instance" and "RREQ-Instance", with values of TBD2 (0x0A) and TBD3 (0x0B) from the "RPL Control Message Options" space [RFC6550].

Value	Meaning	Reference
TBD2 (0x0A)	RREQ Option	This document
TBD3 (0x0B)	RREP Option	This document

Figure 7: AODV-RPL Options

10. Security Considerations

This document does not introduce additional security issues compared to base RPL. For general RPL security considerations, see [RFC6550].

11. References

11.1. Normative References

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

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <http://www.rfc-editor.org/info/rfc3561>.

[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, DOI 10.17487/RFC5548, May 2009, <http://www.rfc-editor.org/info/rfc5548>.

[RFC5673] Pister, K., Ed., Thubert, P., Ed., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, DOI 10.17487/RFC5673, October 2009, <http://www.rfc-editor.org/info/rfc5673>.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, DOI 10.17487/RFC5826, April 2010, <http://www.rfc-editor.org/info/rfc5826>.

[RFC5867] Martocci, J., Ed., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, DOI 10.17487/RFC5867, June 2010, <http://www.rfc-editor.org/info/rfc5867>.

[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>.

[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, <http://www.rfc-editor.org/info/rfc6997>.

[RFC6998] Goyal, M., Ed., Baccelli, E., Brandt, A., and J. Martocci, "A Mechanism to Measure the Routing Metrics along a Point-to-Point Route in a Low-Power and Lossy Network", RFC 6998, DOI 10.17487/RFC6998, August 2013, <http://www.rfc-editor.org/info/rfc6998>.

11.2. Informative References

[I-D.thubert-roll-asymlink] Thubert, P., "RPL adaptation for asymmetrical links", draft-thubert-roll-asymlink-02 (work in progress), December 2011.

Authors' Addresses

Satish Anamalamudi
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
China

Email: satishnaidu80@gmail.com

Mingui Zhang
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
China

Email: zhangmingui@huawei.com

Abdur Rashid Sangi
Huawei Technologies
No.156 Beiqing Rd. Haidian District
Beijing 100095
P.R. China

Email: rashid.sangi@huawei.com

Charles E. Perkins
Futurewei
2330 Central Expressway
Santa Clara 95050
Unites States

Email: charliep@computer.org

S.V.R Anand
Indian Institute of Science
Bangalore 560012
India

Email: anand@ece.iisc.ernet.in

ROLL
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2017

P. Thubert, Ed.
J. Pylakutty
Cisco

October 24, 2016

Root initiated routing state in RPL
draft-thubert-roll-dao-projection-03

Abstract

This document proposes a protocol extension to RPL that enables to install a limited amount of centrally-computed routes in a RPL graph, enabling loose source routing down a non-storing mode DODAG, or transversal routes inside the DODAG. As opposed to the classical route injection by DAO messages, this draft projects the routes from the root of the DODAG.

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 27, 2017.

Table of Contents

1. Introduction

The Routing Protocol for Low Power and Lossy Networks [RFC6550] (LLN) (RPL) specification defines a generic Distance Vector protocol that is designed for very low energy consumption and adapted to a variety of LLNs. RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) which root often acts as the Border Router to connect the RPL domain to the Internet. The root is responsible to select the RPL Instance that is used to forward a packet coming from the Internet into the RPL domain and set the related RPL information in the packets.

In the non-storing mode (NSM) of operation (MOP), the root also computes routes down the DODAG towards the end device and leverages source routing to get there, while the default route via the root is used for routing upwards within the LLN and to the Internet at large. NSM is the dominant MOP because because networks may get arbitrary large and in Storing Mode, the amount of memory in nodes close to the root may unexpectedly require memory beyond a node's capabilities.

But as a network gets deep, the size of the source routing header that the root must add to all the downward packets may also become an issue for far away target devices. In some use cases, a RPL network forms long lines and a limited amount of well-targeted routing state would allow to make the source routing operation loose as opposed to strict, and save packet size. Limiting the packet size is directly beneficial to the energy budget, but, mostly, it reduces the chances of frame loss and/or packet fragmentation, which is highly detrimental to the LLN operation. Because the capability to store a

routing state in every node is limited, the decision of which route is installed where can only be optimized with a global knowledge of the system, a knowledge that the root has in non-storing mode.

Additionally, RPL storing mode is optimized or Point-to-Multipoint (P2MP), root to leaves and Multipoint-to-Point (MP2P) leaves to root operations, whereby routes are always installed along the RPL DODAG. Transversal Peer to Peer (P2P) routes in a RPL network will generally suffer from some stretch since routing between 2 peers always happens via a common parent. In NSM, all peer-to-peer routes travel all the way to the root, which adds a source routing header and forwards the packet down to the destination, resulting in the longest stretch and overload of the radio bandwidth near the root. A controller, for instance collocated with the RPL root, with enough topological awareness of the connectivity between nodes, would be able to compute more direct routes, avoiding the vicinity of the root whenever possible.

The 6TiSCH architecture [I-D.ietf-6tisch-architecture] leverages the Deterministic Networking Architecture [I-D.finn-detnet-architecture] as one possible model whereby the device resources and capabilities are exposed to an external controller which installs routing states into the network based on some objective functions that reside in that external entity.

Based on heuristics of usage, path length, and knowledge of device capacity and available resources such as battery levels and reservable buffers, a Path Computation Element ([PCE]) with a global visibility on the system could install additional P2P routes that are more optimized for the current needs as expressed by the objective function.

This draft enables a RPL root, with optionally the assistance of a PCE, to install and maintain additional storing mode routes within the RPL domain, along a selected set of nodes and for a selected duration, thus providing routes from suitable than those obtained from the distributed operation of RPL in either storing and non-storing modes.

2. Terminology

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

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

3. New RPL Control Message Options

Section 6.7 of [RFC6550] specifies Control Message Options (CMO) to be placed in RPL messages such as the DAO message. The RPL Target Option and the Transit Information Option (TIO) are such options; the former indicates a node to be reached and the latter specifies a parent that can be used to reach that node. Options may be factorized; one or more contiguous TIOs apply to the one or more contiguous Target options that immediately precede the TIOs in the RPL message.

This specification introduces a new Control Message Option, the Via Information option (VIO). Like the TIO, the VIO MUST be preceded by one or more RPL Target options to which it applies. Unlike the TIO, the VIO are not factorized: multiple contiguous Via options indicate an ordered sequence of hops to reach the target(s), presented in the same order as they would appear in a routing header.

3.1. Via Information

The Via Information option MAY be present in DAO messages, and its format is as follows:

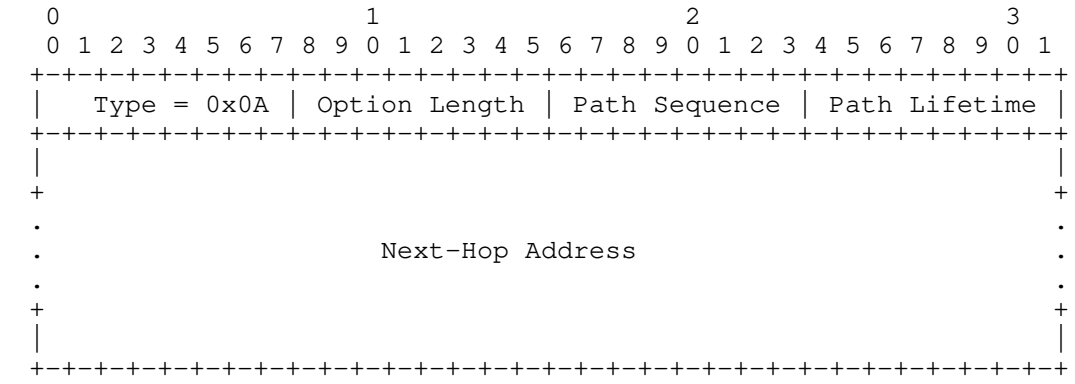


Figure 1: Eliding the RPLInstanceID

Option Type: 0x0A (to be confirmed by IANA)

Option Length: Variable, depending on whether or not Parent Address is present.

Path Sequence: 8-bit unsigned integer. When a RPL Target option is issued by the root of the DODAG (i.e. in a DAO message), that root sets the Path Sequence and increments the Path Sequence

each time it issues a RPL Target option with updated information. The indicated sequence deprecates any state for a given Target that was learned from a previous sequence and adds to any state that was learned for that sequence.

Path Lifetime: 8-bit unsigned integer. The length of time in Lifetime Units (obtained from the Configuration option) that the prefix is valid for route determination. The period starts when a new Path Sequence is seen. A value of all one bits (0xFF) represents infinity. A value of all zero bits (0x00) indicates a loss of reachability. A DAO message that contains a Via Information option with a Path Lifetime of 0x00 for a Target is referred as a No-Path (for that Target) in this document.

Next-Hop Address: 8 or 16 bytes. IPv6 Address of the next hop towards the destination(s) indicated in the target option that immediately precede the VIO. The /64 prefix can be elided if it is the same as that of (all of) the target(s). In that case, the Next-Hop Address is expressed as the 8-bytes suffix only, otherwise it is expressed as 16 bytes.

4. Loose Source Routing in Non-storing Mode

A classical RPL implementation in a very constrained LLN uses the non-storing mode of operation whereby a RPL node indicates a parent-child relationship to the root, using a Destination Advertisement Object (DAO) that is unicast from the node directly to the root, and the root builds a path to a destination down the DODAG by concatenating this information.

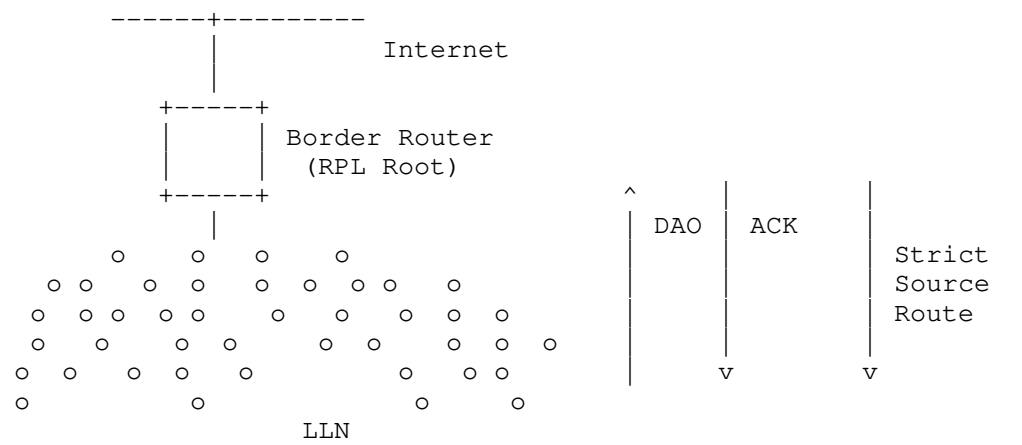


Figure 2: RPL non-storing operation

Nodes are not expected to store downward routing state via their children, and the routing operates in strict source routing mode as detailed in An IPv6 Routing Header for Source Routes with RPL [RFC6554]

This draft proposes an addition whereby the root projects a route through an extended DAO to an arbitrary node down the DODAG, indicating a child or a direct sequence of children via which a certain destination (target) may be reached. The root is expected to use the mechanism optimally and with required parsimony to fit within the device resources, but how the root figures the amount of resources that are available is out of scope.

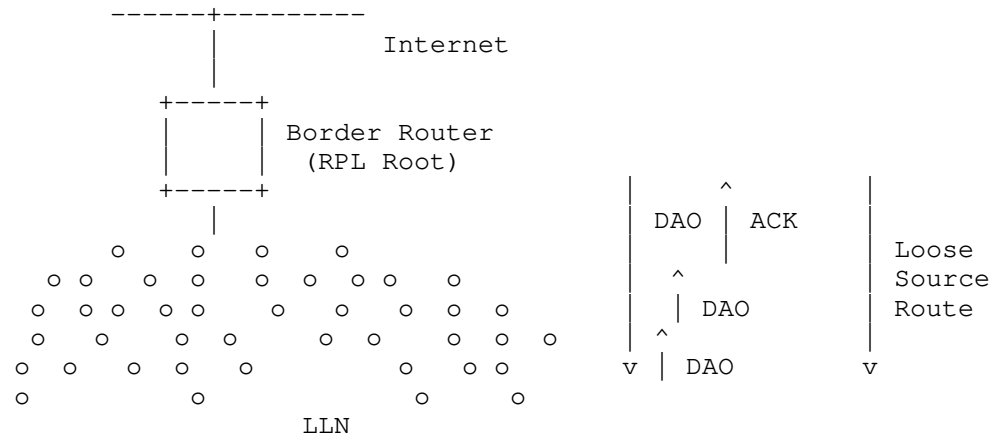


Figure 2: Non-Storing with Projected routes

When a RPL domain operates in non-storing Mode of Operation (NS-MOP), only the root possesses routing information about the whole network. A packet that is generated within the domain first reaches the root, which can then apply a source routing information to reach the destination. Similarly, a packet coming from the outside of the domain for a destination that is expected to be in a RPL domain reaches the root.

In NS-MOP, the root, or some associated centralized computation engine, can thus determine the amount of packets that reach a destination in the RPL domain, and thus the amount of energy and bandwidth that is wasted for transmission, between itself and the destination, as well as the risk of fragmentation, any potential delays because of a paths longer than necessary (shorter paths exist that would not traverse the root).

Additionally, the DAG root knows the whole DAG topology, so when the source of a packet is also in the RPL domain, the root can determine the common parent that would have been used in storing mode, and thus the list of nodes in the path between the common parent and the destination. For instance in the below diagram, if the source is 41 and the destination 52, the common parent is the node 22.

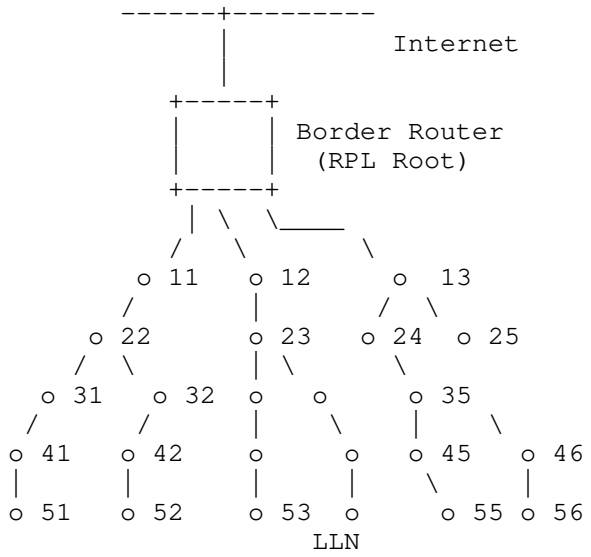


Figure 3: Non-Storing with Projected routes

With this draft, the root can install routing states along a segment that is either itself to the destination, or from one or more common parents for a particular source/destination pair towards that destination (in our example, this would be the segment made of nodes 22, 32, 42).

The draft expects that the root has enough information about the capability for each node to store a number of routes, which can be discovered for instance using a Network Management System (NMS) and/or the RPL routing extensions specified in Routing for Path Calculation in LLNs [RFC6551]. Based on that information, the root computes which segment should be routed and which relevant state should be installed in which nodes. The algorithm is out of scope but it is envisaged that the root could compute the ratio between the optimal path (existing path not traversing the root, and the current path), the application SLA for specific flows that could benefit from shorter paths, the energy wasted in the network, local congestion on various links that would benefit from having flows routed along other paths.

This draft introduces a new mode of operation for loose source routing in the LLN, the Non-Storing with Projected routes MOP. With this new MOP, the root sends a unicast DAO message to the last node of the routing segment that must be installed. The DAO message contains the ordered list of hops along the segment as a list of Via Information options that are preceded by one or more RPL Target options to which they relate. Each Via Information option contains a lifetime for which state is to be maintained.

The root sends the DAO directly to the last node in the segment, which is expected to be able to route to the targets on its own.

The last node in the segment may have another information to reach the target(s), such as a connected route or an already installed projected route. If it does not have such a route then the node should lookup the address on the relevant interfaces. If one of the targets cannot be located, the node MUST answer to the root with a negative DAO-ACK listing the target(s) that could not be located (suggested status 10), and continue the process for those targets that could be located if any.

For the targets that could be located, last node in the segment generates a DAO to its loose predecessor in the segment as indicated in the list of Via Information options.

The node strips the last Via Information option which corresponds to self, and uses it as source address for the DAO to the predecessor. The address of the predecessor to be used as destination for the DAO message is found in the now last Via Information option. The predecessor is expected to have a route to the address used as source, either connected, installed previously as another DAO, or from other means.

The predecessor is expected to have a route to the address used as source and that is his successor. If it does not and cannot locate the successor, the predecessor node MUST answer to the root with a negative DAO-ACK indicating the successor that could not be located. The DAO-ACK contains the list of targets that could not be routed to (suggested status 11).

If the predecessor can route to the successor node, then it installs a route to the targets via the successor. If that route is not connected then a recursive lookup will take place to reach the target(s). From there, the node strips the last Via Information option and either answers to the root with a positive DAO-ACK that contains the list of targets that could be routed to, or propagates the DAO to its own predecessor.

A NULL lifetime in the Via Information option along the segment is used to clean up the state.

In the example below, say that there is a lot of traffic to nodes 55 and 56 and the root decides to reduce the size of routing headers to those destinations. The root can first send a DAO to node 45 indicating target 55 and a Via segment (35, 45), as well as another DAO to node 46 indicating target 56 and a Via segment (35, 46). This will save one entry in the routing header on both sides. The root may then send a DAO to node 35 indicating targets 55 and 56 a Via segment (13, 24, 35) to fully optimize that path.

Alternatively, the root may send a DAO to node 45 indicating target 55 and a Via segment (13, 24, 35, 45) and then a DAO to node 46 indicating target 56 and a Via segment (13, 24, 35, 46), indicating the same DAO Sequence.

5. Centralized Computation of Optimized Peer-to-Peer Routes

With the initial specifications of RPL [RFC6550], the P2P path from a source to a destination is often stretched, as illustrated in [RFC6550]:

- in non-storing mode, all packets routed within the DODAG flow all the way up to the root of the DODAG. If the destination is in the same DODAG, the root must encapsulate the packet to place a Routing Header that has the strict source route information down the DODAG to the destination. This will be the case even if the destination is relatively close to the source and the root is relatively far off.

- in storing mode, unless the destination is a child of the source, the packets will follow the default route up the DODAG as well. If the destination is in the same DODAG, they will eventually reach a common parent that has a DAO route to the destination; at worse, the common parent may also be the root. From that common parent, the packet will follow a path down the DODAG that is optimized for the Objective Function that was used to build the DODAG.

It results that it is often beneficial to enable additional P2P routes, either if the RPL route present a stretch from shortest path, or if the new route is engineered with a different objective.

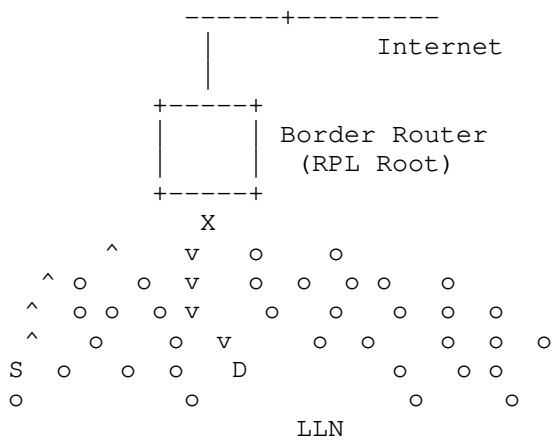


Figure 4: Routing Stretch

For that reason, earlier work at the IETF introduced the Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks [RFC6997], which specifies a distributed method for establishing optimized P2P routes. This draft proposes an alternate based on a centralized route computation.

It must be noted that RPL has a concept of instance but does not have a concept of an administrative distance, which exists in certain proprietary implementations to sort out conflicts between multiple sources. This draft conforms the instance model as follows:

- if the PCE needs to influence a particular instance to add better routes in conformance with the routing objectives in that instance, it may do so. When the PCE modifies an existing instance then the added routes must not create a loop in that instance. This is achieved by always preferring a route obtained from the PCE over a route that is learned via RPL.

- If the PCE installs a more specific (Traffic Engineering) route between a particular pair of nodes then it should use a Local Instance from the ingress node of that path. Only packets associated with that instance will be routed along that path.

In all cases, the path is indicated by VIA options, and the flow is similar to the flow used to obtain loose source routing.

The root sends the DAO with the target option and the Via Option to the lest router in the path; the last router removes the last Via Option and passes the DAO to the previous hop.

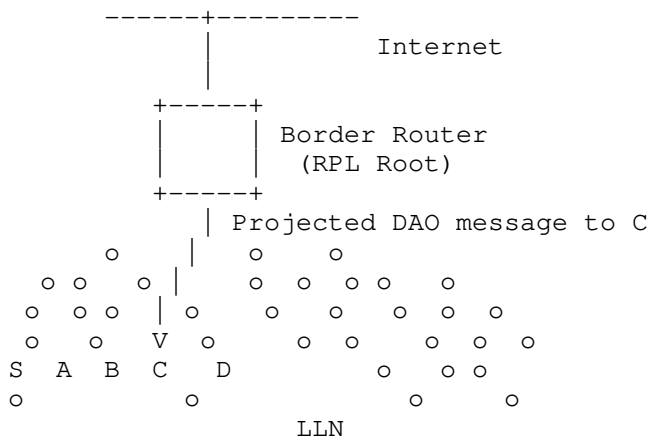


Figure 5: Projected DAO from root

The process recurses till the destination which sends a DAO-ACK to the root. In the example above, for target D, the list of via options is S, A, B and C. The projected DAO is sent by the root to

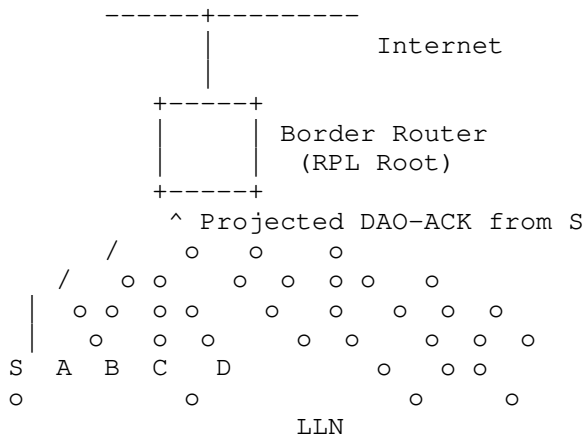


Figure 6: Projected DAO-ACK to root

The process recurses till the destination which sends a DAO-ACK to the root. In the example above, for target D, the list of via options is S, A, B and C. The projected DAO is sent by the root to

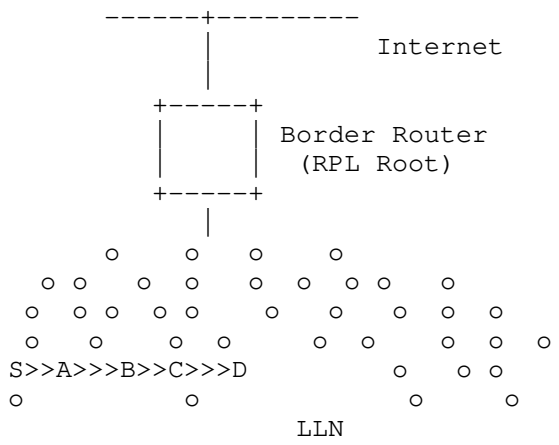


Figure 7: Projected Transversal Route

6. Security Considerations

This draft uses messages that are already present in [RFC6550] with optional secured versions. The same secured versions may be used with this draft, and whatever security is deployed for a given network also applies to the flows in this draft.

7. IANA Considerations

This document updates the IANA registry for the Mode of Operation (MOP)

4: Non-Storing with Projected routes [this]

This document updates IANA registry for the RPL Control Message Options

0x0A: Via descriptor [this]

8. Acknowledgments

The authors wish to acknowledge JP Vasseur and Patrick Wetterwald for their contributions to the ideas developed here.

9. References

9.1. Normative References

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

[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>.

[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, <http://www.rfc-editor.org/info/rfc6551>.

[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>.

[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>.

9.2. Informative References

[I-D.finn-detnet-architecture] Finn, N. and P. Thubert, "Deterministic Networking Architecture", draft-finn-detnet-architecture-08 (work in progress), August 2016.

[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.

[PCE] IETF, "Path Computation Element", <https://datatracker.ietf.org/doc/charter-ietf-pce/>.

[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, <http://www.rfc-editor.org/info/rfc6997>.

Authors' Addresses

Pascal Thubert (editor)
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

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

James Pylakutty
Cisco Systems
Cessna Business Park
Kadubeesanahalli
Marathalli ORR
Bangalore, Karnataka 560087
INDIA

Phone: +91 80 4426 4140
Email: mundenma@cisco.com

roll
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2017

P. van der Stok
consultant
AR. Sangi
Huawei Technologies

October 14, 2016

MPL Forwarder Select (MPLFS)
draft-vanderstok-roll-mpl-forw-select-01

Abstract

This document describes a Forwarder Selection (MPLFS) protocol for the Multicast Protocol for Low-Power and lossy Networks (MPL) to reduce the density of forwarders such that the number of forwarded messages is reduced. The protocol uses Trickle to distribute link-local information about the identity of the neighbours of the nodes that have MPL-enabled interfaces. In the end-state all nodes are connected to a minimum number, N_DUPLICATE, of forwarders, where N_DUPLICATE is application dependent, and there is a path between any two forwarders.

Note

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

Status of This Memo

Table of Contents

1. Introduction

The Multicast Protocol for Low-Power and Lossy Networks (MPL) [RFC7731] is designed for small devices interconnected by a lossy wireless network such as IEEE 802.15.4. A seed sends a multicast message with a realm-local scope, admin-local scope or higher as specified in [RFC4291].

Forwarders forward these messages with an increasing interval size. When the density of forwarders is high, the message may be forwarded by a high number of forwarders that conflict on the link. With extreme forwarder densities and small Trickle intervals, just sending one multicast message may lead to an overload of the communication medium.

The number of forwarded messages can be reduced by selecting a minimal set of forwarders. However, for large networks, manually selecting the forwarders is much work, and changing network conditions and configurations make the manual selection an unwanted burden to the network management.

This document specifies a protocol that selects the forwarders such that each MPL-enabled device is connected to N_DUPLICATE forwarders, where N_DUPLICATE > 0 can be set. The parameter N_DUPLICATE determines how much path redundancy there is for each MPL message. The value of N_DUPLICATE should be at least 1, because a value of 0 has as result that no forwarder exists in the network during the protocol execution. Moreover, the protocol is distributed and dynamic in nature to face a continuously changing topology.

The protocol is inspired by the work described for NeighbourHood Discovery (NHDP) [RFC6130] and Simple Multicast Forwarding (SMF) [RFC6621]. In contrast to the "HELLO" messages described in [RFC6130], this protocol uses the Trickle protocol [RFC6206] to multicast link-local messages, containing a CBOR payload [RFC7049].

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].

Readers of this specification should be familiar with all the terms and concepts discussed in [RFC7731]. The following terms are defined in this document:

synchronization time The moment that a node can change its state at messages reception.

The following list contains the abbreviations used in this document.

XXXX: TODO, and others to follow.

2. Protocol overview

Nodes participating in MPLFS exchange messages with a format that is described in Section 6. A participating node communicates to all its neighbours with link-local multicast messages as described in Section 4.

Failing links provide a lot of instability. Only messages sent over stable links are accepted. Section 4 describes a mechanism to refuse messages from unstable links.

Each node maintains a set of 1-hop neighbours where each neighbour contains information about its own 1-hop neighbours. On the basis of the contents of the set, the node can decide to become a Forwarder or not, as explained in Section 5.

The protocol never ends, with a minimum frequency of exchanging maintenance messages specified by an interval size of I_MAX_SELECT. When the set of links is stable, the protocol stabilizes such that there is a path between any two forwarders, and every MPL-enabled node is connected to at least N_DUPLICATE MPL forwarders (when existing), where N_DUPLICATE > 0. N_DUPLICATE can be set dependent on the application requirements. With N_DUPLICATE = 2, it is expected that a multicast message arrives at an intended recipient with very high probability.

Nodes have a state that determines whether they are forwarder or not. The state of a node can only be changed by the node itself. To avoid race conditions, (e.g. two nodes simultaneously decide to be no forwarder, while only one is intended) the node with the highest address of all 1-hop neighbours is the only one allowed to change state. Unlike [RFC5614], that considers 3-tuple (Router Priority, MDR Level and Router ID) to allow self state change, this approach only takes into account the node address. Consequently, only k-hop neighbours, with k > 2, can change state simultaneously, and the 1-hop and 2-hop neighbours of a given node can change state one by one.

3. Data sets

Each node, n_0, maintains a state with two values: Fixed Forwarder (FF) and No Forwarder (NF). Each node also maintains a set, S1_0, containing information about n_0's 1-hop neighbours and n_0 itself. Each entry, n_i, in S1_0 has the following attributes:

address of n_i: the address can be the 64 bit IPv6 address or the short 16 bit address.

average-rssi-in: the average rssi of the messages received by n_0 from n_i.

average-rssi-out: the average rssi of the messages received by n_i from n_0.

nr_FF: the number of neighbours, n_ij, of n_i (including n_i) with state = FF.

nr_Under: the number of neighbours, n_ij, of n_i with nr_FF < N_DUPLICATE.

nr_Above: the number of neighbours, n_ij, of n_i with nr_FF > N_DUPLICATE.

size: the size of S1_i, the set of 1-hop neighbours of n_i.

state: the state of n_i.

4. Neighbor distribution

A participating node multicasts link-local so-called "neighbour messages" with the Trickle protocol. It uses the multicast address LINK_LOCAL_ALL_NODES as destination. The message sent by n_0 contains the contents of S1_0. The contents of a "neighbour message" from n_i received by n_0 is called M_i. The rssi value associated with the reception of the "neighbour message" is called new_rssi. The message M_i contains information about the set S1_i with the following attributes for all nodes in S1_i:

- address
- average-rssi-in
- nr_FF
- nr_Under
- nr_Above
- size
- state

On reception of M_i from n_i for the first time, the receiving node adds n_i to S1_0, and sets average-rssi-in of n_i in S1_0 to new_rssi. For all following messages from n_i, the average-rssi-in for n_i is calculated in the following way: average-rssi-in := (average-rssi-in*WEIGHT_AVERAGE + new_rssi)/(WEIGHT_AVERAGE+1).

The neighbour nodes of M_i are called n_ij. For the n_ij with an address that is equal to the address of n_0: the value of average-rssi-out of n_0 is set equal to the value of average-rssi-in of n_ij.

The contents of n_0 is updated with the contents of M_i. Updating includes the following actions:

- Add n_i to S1_0, if n_i not present in S1_0.
- Set size of n_i equal to the number of entries in M_i.

- When n_ij.address = n_j.address, copy the values of nr_Under, nr_Above, nr_FF, and state of n_ij to n_j.

When the average-rssi-in and average-rssi-out values of n_i have been averaged over more than WEIGHT_AVERAGE messages, and the averaged RSSI values are smaller than MAXIMUM_RSSI, n_i is called "valid".

5. Selection Algorithm

The protocol aims at allocating forwarders in the densest part of the network. A dense network is characterized by a high number of neighbours. Therefore, the protocol attempts to assign status FF to the nodes with the highest number of neighbours that have less than N_DUPLICATE neighbours with state = FF (nr_FF < N_DUPLICATE).

It is required that a path exists between every two forwarders to prevent network partitioning. Therefore, a node can become forwarder iff one of its neighbours is a forwarder. The consequence of this rule is that one so-called "source-forwarder" must be selected by the network administrator. A likely choice for the "source-forwarder" is the border router.

At the start of the selection protocol the node, n_0, sets its state to No Forwarder (NF). It sets the Trickle timer to its minimum interval, I_MIN_SELECT, and starts multicasting M_0 to its neighbours. Every time entries are added to, or removed from, S1_0, the Trickle interval timer is set to I_MIN_SELECT.

The executing node, n_0, calculates the following parameter values:

- max-under is the maximum of the nr_Under attribute of all valid n_i in S1_0.

- max_address_u is the maximum of the addresses of valid n_i with nr_Under = max-under.

- max_address_a is the maximum of the addresses of all valid n_i.

- connected is true iff nr_FF of all neighbouring forwarders is equal to nr_FF of n_0.

The information about the state and the nr_Under value of the neighbours comes in asynchronously. Time is needed before the state in a node correctly reflects the state changes of the network. A node can change its state when during the reception of messages of all neighbours, the value of nr_Under has not changed.

To calculate its new state, n_0 does the following:

When the state is NF, a neighbour with state = FF exists, and address = max-address_u:
set state to FF.

When the state is FF, nr_Above = size S1_0, connected is true, and address = max_address_a:
set state to NF.

6. CBOR payload

The payload format is /application/cbor [RFC7049]. The contents of the message is a CBOR array (Major type 4) of CBOR arrays composed of neighbour address, rssi value, size of S1_i, forwarder state, nr_FF, nr_Under, and nr_Above. Assuming two neighbours, in diagnostic JSON the payload looks like:

```
[
[address_1, average-rssi-in_1, size_1, state_1,
    nr_FF_1, nr_Under_1, nr_Above_1],
[address_2, average-rssi-in_2, size_2, state_2,
    nr_FF_2, nr_Under_2, nr_Above_2]
]
```

Figure 1: CBOR payload

7. Default parameter values

The following text recommends default values for the MPLFS protocols.

I_MIN_SELECT = 0,2; minimum Trickle timer interval.

I_MAX_SELECT = 10; maximum Trickle timer interval.

WEIGHT_AVERAGE = 10; number of values to average rssi.

MAXIMUM_RSSI = 3; maximum acceptable average rssi value.

N_DUPLICATE = 2; requested number of MPL forwarder neighbours for every MPL enabled node.

8. Acknowledgements

We are very grateful to

9. Changelog

Changes from version 00 to version 01

- Definition of S1_0 improved
- Algorithm changed and simulated
- Moment of state change specified

10. References

10.1. Normative References

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

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[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>.

10.2. Informative References

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5614] Ogier, R. and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding", RFC 5614, DOI 10.17487/RFC5614, August 2009, <http://www.rfc-editor.org/info/rfc5614>.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, DOI 10.17487/RFC6130, April 2011, <http://www.rfc-editor.org/info/rfc6130>.

[RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <http://www.rfc-editor.org/info/rfc6621>.

Authors' Addresses

Peter van der Stok
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI: www.vanderstok.org

Abdur Rashid Sangi
Huawei Technologies
No.156 Beiqing Rd. Haidian District
Beijing 100095
P.R. China

Email: rashid.sangi@huawei.com