ROLL Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

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

October 19, 2015

When to use RFC 6553, 6554 and IPv6-in-IPv6
draft-robles-roll-useofrplinfo-02

Abstract

This document states different cases where RFC 6553, RFC 6554 and IPv6-in-IPv6 encapsulation is required to set the bases to help defining the compression of RPL routing information in LLN environments.

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 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

RPL [RFC6550] is a routing protocol for constrained networks. RFC 6553 [RFC6553] defines the "RPL option", carried within the IPv6 Hop-by-Hop header to quickly identify inconsistencies in the routing topology. RFC 6554 [RFC6554] defines the "RPL Source Route Header", an IPv6 Extension Header to deliver datagrams within a RPL routing domain.

Several discussions in the ROLL/6lo/6TiSCH Mailing Lists took place focusing in the definition of how to compress RPL Information in constrained environment. ROLL Virtual Interim Meeting (02-2015) concluded that there is a need to define how to use [RFC6553], [RFC6554] and IPv6-in-IPv6 encapsulation to be able to set the correct environment for compression. A Routing Header Dispatch for 6LoWPAN (6LoRH) [I-D.thubert-6lo-routing-dispatch] defines a method to compress RPL Option information and Routing Header type 3 (RFC6554) and an efficient IP-in-IP technique.

This document is going to be focused in data plane messages and how can be transmitted within the above mentioned RFCs.

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]

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 message with Type 155. DIS, DIO and DAO messages are all RPL Control messages but with different Code values.

RPL supports two modes of Downward traffic: in storing mode, it is fully stateful or an in non-storing, 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.

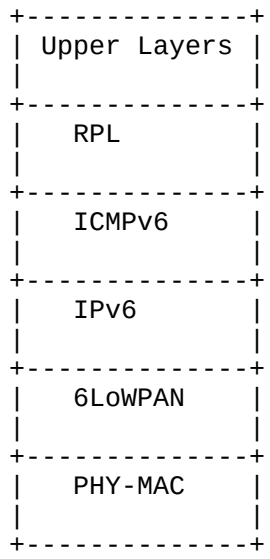

Figure 1: RPL Stack.

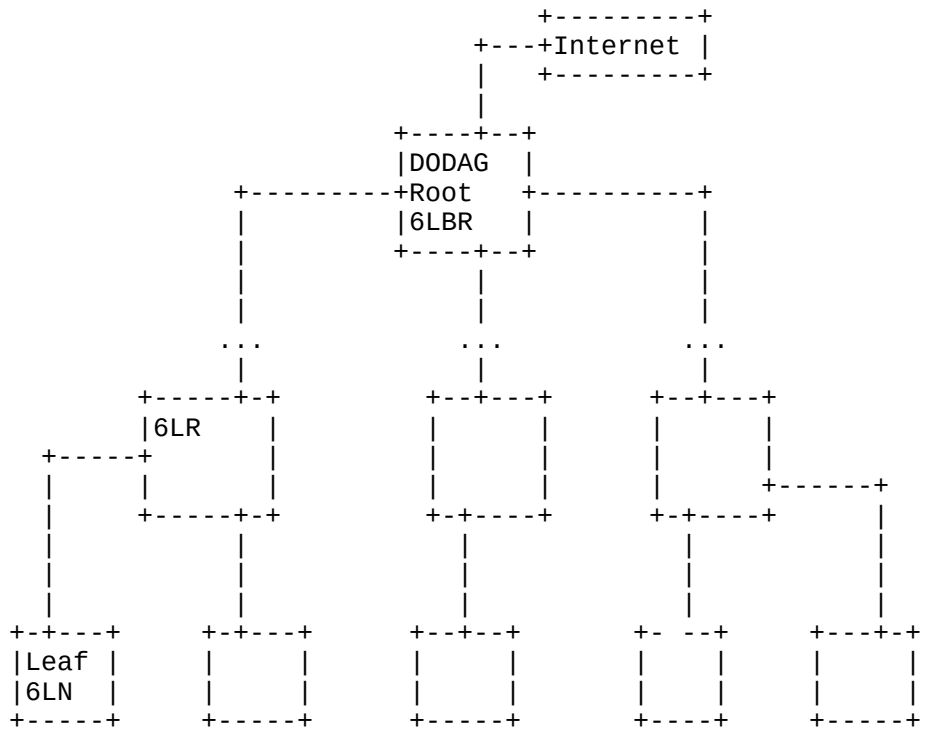

Figure 2: A reference RPL Topology.

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. This architecture is used for the remainder of this document.

The scope of the 6TiSCH Architecture is a Backbone Link that federates multiple LLNs (mesh) as a single IPv6 Multi-Link Subnet. Each LLN in the subnet is anchored at a Backbone Router (6BBR). The Backbone Routers interconnect the LLNs over the Backbone Link and emulate that the LLN nodes are present on the Backbone thus creating a so-called: Multi-Link Subnet. An LLN node can move freely from an LLN anchored at a Backbone Router to another LLN anchored at the same or a different Backbone Router inside the Multi-Link Subnet and conserve its addresses.

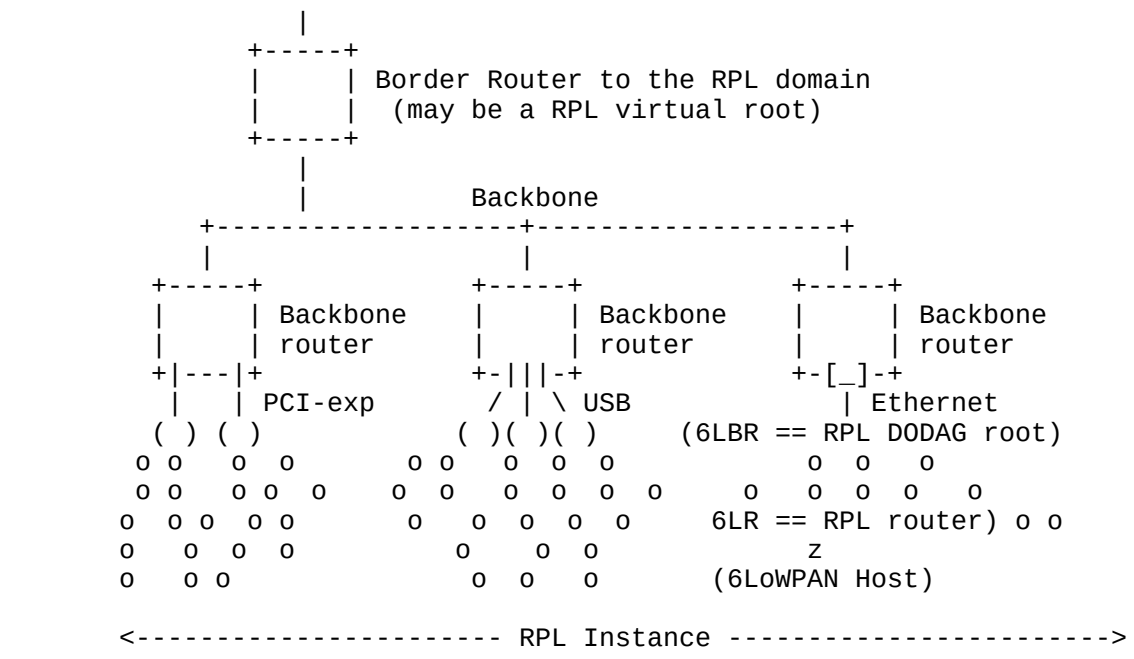

Figure 3: RPL domain architecture

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:

-Flow from RPL-aware-leaf to root

-Flow from root to RPL-aware-leaf

-Flow from not-RPL-aware-leaf to root

-Flow from root to not-RPL-aware-leaf

-Flow from RPL-aware-leaf to Internet

-Flow from Internet to RPL-aware-leaf

-Flow from not-RPL-aware-leaf to Internet

-Flow from Internet to not-RPL-aware-leaf

-Flow from RPL-aware-leaf to RPL-aware-leaf

-Flow from RPL-aware-leaf to not-RPL-aware-leaf

-Flow from not-RPL-aware-leaf to RPL-aware-leaf

-Flow from not-RPL-aware-leaf to not-RPL-aware-leaf

This document assumes a rule that a Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed.

- This means that an intermediate router that needs to add a header must encapsulate the packet in an outer IP header where the new header can be placed.

- This also means that a Header can only be removed by an intermediate router if it is placed in an encapsulating IPv6 Header, and in that case, the whole encapsulating header must be removed - a replacement may be added.

This document recognizes that some headers such as a Routing Header or a Hop-by-Hop header may be modified by routers on the path of the packet without the need to add to remove an encapsulating header.

The RPL RH and the RPL option are mutable but recoverable .

RPI should be present in every single RPL data packet. There is an exception in non-storing mode, when a packet is going down from the route: the entire route is written, so there are no loops of confusion about which table to use (purpose of instanceID).

The applicability for storing (RPL-SN) and non-Storing (RPL-NSN) modes for the previous cases is showed as follows:

Use Case	RPL-SN RPI (RFC 6553)	RPL-SN RH3 (RFC 6554)	RPL-SN IP-in-IP	RPL-NSN RPI	RPL-NSN RH3	RPL-NSN IP-in-IP
RPL-aware-leaf to root	Yes	No	No	Yes	No	No
root to RPL-aware-leaf	Yes	No	No	Yes	Yes	No
not-RPL-aware-leaf to	Yes	No	Yes	Yes	No	Yes

root							
root to not-RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	
RPL-aware-leaf to Internet	Yes	No	Yes	Yes	No	Yes	
Internet to RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	
not-RPL-aware-leaf to Internet	Yes	No	Yes	Yes	No	Yes	
Internet to not-RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	
RPL-aware-leaf to RPL-aware-leaf	Yes	No	No	Yes	Yes	Yes	
RPL-aware-leaf to not-RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	
not-RPL-aware-leaf to RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	
not-RPL-aware-leaf to not-RPL-aware-leaf	Yes	No	Yes	Yes	Yes	Yes	

Table 1: Posibility to transmit in Storing or Non-Storing mode: RPI, RH3, IP-in-IP encapsulation

5. Storing mode

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

As states in Section 16.2 of [RFC6550] a RPL-aware-leaf node does not generally issue DIO messages, a leaf node accepts DIO messages (In inconsistency a leaf node generates DIO with infinite rank, to fix it). It may issue DAO and DIS messages though it generally ignores DAO and DIS messages.

In storing mode is suitable the use of RFC 6553 to send RPL Information through HBH field checking the routing table to find out where to send the message.

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR --> 6LR,... --> root (6LBR) Note: In this document 6LRs, 6LBR are always full-fledge RPL routers

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

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

In this case the 6LBR insert RPI header and send 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.

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)--> 6LR --> not-RPL-aware-leaf (6LN)

It includes IPv6-in-IPv6 encapsulation to transmit information not related with the RPL domain. In the 6LBR the RPI header is inserted into an IPv6-in-IPv6 header addressed to the last 6LR, which removes the header before pass the packet to the IPv6 node.

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

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 (6LN) --> 6LR --> root (6LBR)

When the packet arrives from IPv6 node to 6LR. This router insert the RPI encapsuladed in a IPv6-in-IPv6 header addressed to the root. The root removes the header and process the packet

Header	IPv6	6LR	6LBR
Inserted headers	--	IPv6-in-IPv6(RPI)	--
Removed headers	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--
Modified headers	--	--	--
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 should not go out to Internet. The router should take this information out before send the packet to Internet. The HBH Option is going to be analyzed in each node to the root.

In this case the flow comprises:

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

6LN insert RPI in a IPv6-in-IPv6 in a outer header, and send the packet to 6LR, which modified the rank in the RPI. When the packet arrives to 6LBR, the RPI is removed.

Header	6LN	6LR	6LBR	Internet
Inserted headers	IPv6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	IPv6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	--	--

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) --> 6LR --> 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 send to 6LR, which modifies the rank in the RPI. When the packet arrives 6LN the RPI header is removed and the packet processed.

Header	Internet	6LBR	6LR	6LN
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	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 (6LN) = IPv6 node --> 6LR --> root (6LBR) --> Internet

In the IPv6 node the flow label is assumed to be zero, the packet is transmited to 6LR which encapsule the RPI header in an outer IPv6-in-IPv6 header and send to 6LBR, which removes this header and send the packet to Internet and might set the flow label field.

Header	IPv 6	6LR	6LBR	Internet
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	IPv6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
Untouched headers	--	--	--	--

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

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

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR --> not-RPL-aware-leaf (6LN)

6LBR get the packet from Internet and add a RPI header encapsulated in a IPv6-in-IPv6 header addressed to 6LR and send the packet down. The flow label is set to zero on inner IP. The last 6LR removes the RPI header. The IPv6 node might set the flow label since may arrive with zero value.

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

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

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

In [RFC6550] RPL allows a simple one-hop P2P optimization for both storing and non-storing networks. A node may send a P2P packet destined to a one-hop neighbor directly to that node. Section 9 in [RFC6550].

In this case the flow comprises:

6LN --> 6LR --> common parent (6LR) --> 6LR --> 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).

Header	6LN src	6LR	6LR (common parent)	6LR	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 --> 6LR --> common parent (6LR) --> 6LR --> not-RPL-aware 6LN

Somehow, the sender has to know that the receiver is not RPL aware, and needs to know 6LR, and not even the root knows where the 6LR is (in storing mode). This case FAILS.

Header	6LN	6LR	6LR (common parent)	6LR	IPv6
Inserted header s	IPv6-in-IPv6(RPI)	--	--	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--	--
Modified header s	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(RPI)	--	--
Untouched headers	--	--	--	--	--

Storing: Summary of the use of headers from RPL-aware-leaf to not-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 --> 6LR --> common parent (6LR) --> 6LR --> 6LN

The 6LR that get the packet from IPv6 node, insert the RPI header encapsulated in IPv6-in-IPv6 header with destination to 6LN, the common parent change the direction of RPI and finally it is removed by 6LN.

Header	IPv6	6LR	common parent (6LR)	6LR	6LN
Inserted header s	--	IPv6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--	--	--
Modified header s	--	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(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 --> 6LR --> root (6LBR) --> 6LR --> not-RPL-aware 6LN

The problem to solve is how to indicate where to send the packet when get into LLN.

Header	IPv6 src	6LR	6LR (common parent)	6LR	IPv6 dst
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	--	--
Untouched headers	--	--	--	--	--

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

6. Non Storing mode

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

In non-storing mode the leaf node uses Hop-By-Hop option (RFC 6553) to indicate the routing information to send messages to the DODAG root, this message is going to be analyzed in each node until arrive the DODAG root.

In this case not need to use IPv6-in-IPv6 because no header is not going to be removed, neither RH3, the flow comprises:

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

This case 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)

6LBR might instert RPI header, and the rute is indicated in RH3. 6LR updated RH3 and 6LN remove these headers.

Header	6LBR	6LR	6LN
Inserted headers	(optional: 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)--> 6LR --> not-RPL-aware-leaf (IPv6 node)

In 6LBR the RH3 is added, and modified in 6LR where is fully consumed, but left there. If the RPI is present, the IPv6 node which does not understand it will drop it. To avoid it the RPI should be removed before reach IPv6 node or it is recommended that RPI be omitted. An IPv6-in-IPv6 header should be necessary in this case. The DAO from 6LR about IPv6 could say if that the final IPv6 is not RPL (RPI) capable.

Header	6LBR	6LR	IPv6
Inserted headers	RH3	--	--
Removed headers	--	--	--
Re-added headers	--	--	--
Modified headers	--	RH3	--
Untouched headers	--	--	--

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 --> 6LR2 --> root (6LBR)

In this case the RPI is encapsulated in the first 6LR, and is not modified in the followings 6LRs.

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

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

This case requires that the network is awareness of what is external to the LLN. Internet node never sees RPI or IPv6-in-IPv6 header. In

the 6LBR the flow label is computed if it is zero. RPI remains unmodified.

Header	6LN	6LR	6LBR	Internet
Inserted headers	IPV6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	IPV6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
Untouched headers	--	RPI	--	--

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

If the last RH3 entry is the 6LR, then the IPv6-in-IPv6 will be removed there, if the last entry is the 6LN, then the RH3 will go all the way to the leaf. In 6LBR the flow label should be set to zero.

Header	Internet	6LBR	6LR	6LN
Inserted headers	--	IPv6-in-IPv6(RH3,optional:RPI)	--	--
Removed headers	--	--	IPv6-in-IPv6 can be removed if RH3 consumed	--
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RH3)	--
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 (6LN) --> 6LR --> root (6LBR) --> Internet

In this case the flow label is recommended to be zero in the IPv6 node. no RPL headers are added in the IPv6 node, since it is ignorant of RPL. Internet node does not see special headers. In 6LBR the flow label is computed if it is zero.

Header	IPv 6	6LR	6LBR	Internet
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	IPv6-in-IPv6(RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
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 not-RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR --> not-RPL-aware-leaf (6LN)

In this case the flow label in 6LBR should be set zero in 6LBR, where RH3 is inserted and optionally RHI. RH3 must end at 6LR.

Header	Internet	6LBR	6LR	IPv6
Inserted headers	--	IPv6-in-IPv6(RH3,optional:RPI)	--	--
Removed headers	--	--	IPv6-in-IPv6(RH3, RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
Untouched headers	--	--	--	--

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

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

In this case the flow comprises:

6LN --> 6LR --> root (6LBR) --> 6LR --> 6LN

This case comprises in the same RPL Domain. In the 6LN the RPI header is inserted. In the 6LBR the RH3 header is inserted in a IPv6-in-IPv6 header and removed at the 6LN destination.

Header	6LN src	6LBR	6LR	6LN dst
Inserted headers	RPI	IPv6-in-IPv6(RH3 to 6LN) {IP,RPI,payload}	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RH3) {IP,RPI,payload}
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 --> 6LR --> root (6LBR) --> 6LR --> not-RPL-aware 6LN

The 6LN insert the RPI in a IPv6-in-IPv6 header, which is addressed to 6LBR. The 6LBR remove this RPI header and insert a RH3 header with an optional RPI. These headers are removed by 6LR before send the packet to the IPv6 node.

Header	6LN	6LBR	6LR	IPv6
Inserted headers	IPv6-in-IPv6(RPI)	IPIP(RH3, opt RPI)	--	--
Removed headers	--	IPIP(RPI)	IPIP(RH3, opt RPI)	--
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
Untouched headers	--	--	--	--

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 --> 6LR --> root (6LBR) --> 6LR --> 6LN

RPI is added in 6LR until the root and then removed, then RH3 is added and removed at destination.

Header	IPv6	6LR	6LBR	6LN
Inserted headers	--	IPIP(RPI)	IPIP(RH3)	--
Removed headers	--	IPIP(RPI)	--	IPIP(RH3)
Re-added headers	--	--	--	--
Modified headers	--	--	--	--
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 --> 6LR --> root (6LBR) --> 6LR --> not-RPL-aware 6LN

RPI is added in 6LR until the root and then might be removed, then RH3 is added. These headers are removed at 6LR before go to destination.

Header	IPv6	6LR	6LBR	6LR	IPv6
Inserted headers	--	IPIP(RPI)	IPIP(RH3)	--	--
Removed headers	--	--	IPIP(RPI)	IPIP(RH3, opt RPI)	--
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. Future RPL work

There are cases from above that are not clear how to send the information. It requires furhter analysis on how to proceed to send the information from source to destination.

From the above cases, we have in storing mode:

- Flow from RPL-aware-leaf to non-RPL-aware-leaf: Somehow, the sender has to know that the receiver is not RPL aware, and needs to know 6LR, and not even the root knows where the 6LR is located.

- Flow from not-RPL-aware-leaf to not-RPL-aware-leaf: The problem to solve is how to indicate where to send the packet when get into LLN.

8. IANA Considerations

There are no IANA considerations related to this document.

9. Security Considerations

TODO.

10. 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 Thomas Watteyne, Xavier Vilajosana and Robert Cragie.

To be completed with additional Acknowledgments.

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>.

[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>.

11.2. 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-08 (work in progress), May 2015.

[I-D.thubert-6lo-routing-dispatch] Thubert, P., Bormann, C., Toutain, L., and R. Cragie, "A Routing Header Dispatch for 6LoWPAN", draft-thubert-6lo-routing-dispatch-06 (work in progress), August 2015.

[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>.

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/

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
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2016

P. Thubert, Ed.
J. Pylakutty
Cisco

November 2, 2015

Root initiated routing state in RPL
draft-thubert-roll-dao-projection-02

Abstract

This document proposes a root-initiated protocol extension to RPL that enables to install a limited amount of downward routes in non-storing mode. This enables loose source routing down 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 May 5, 2016.

Copyright Notice

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.

The non-storing mode of operation (MOP) is largely utilized because networks can get very large and the amount of memory in nodes close to the root may become prohibitive in storing mode.

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 as well. In some cases, RPL network form long lines and a limited number of well-targeted routes would enable a loose source routing operation and save packet size, energy, and eventually fragmentation which is highly detrimental to the LLN operation. Because the capability to store 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.

RPL storing mode is optimized or Point-to-Multipoint (P2MP), root to leaves and Multipoint-to-Point (MP2P) leaves to root operations. 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.

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:

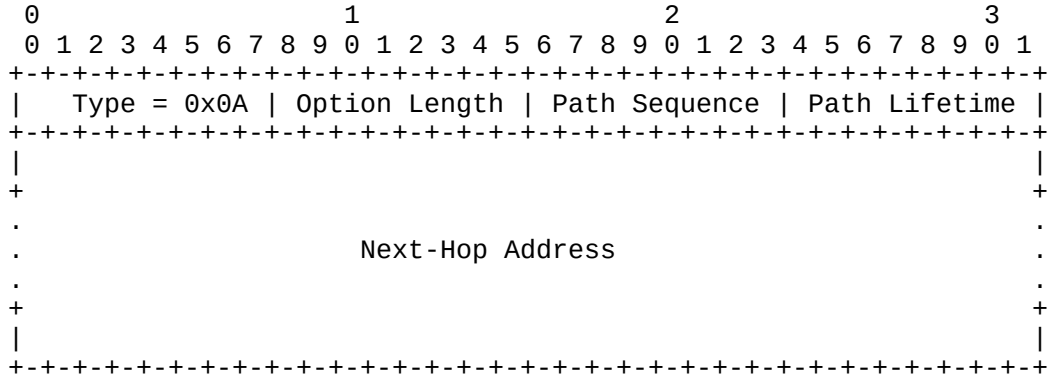

Figure 1: Eliding the RPLInstanceID

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.

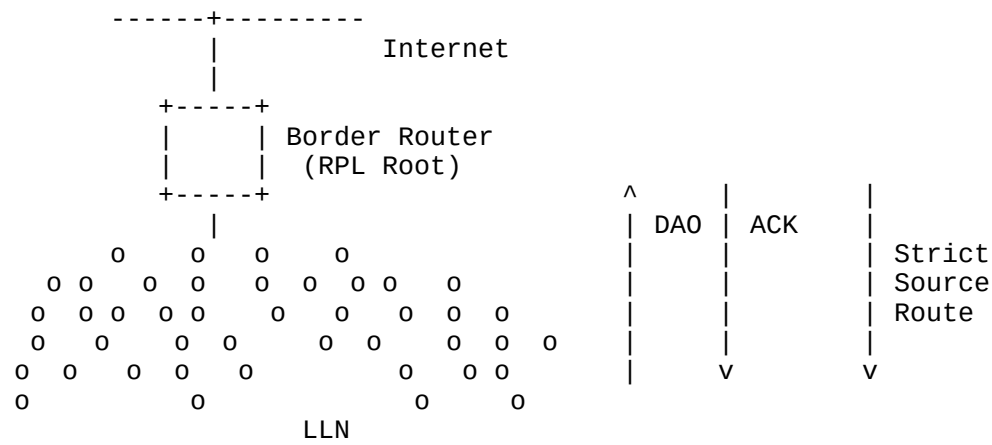

Figure 2: RPL non-storing operation

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.

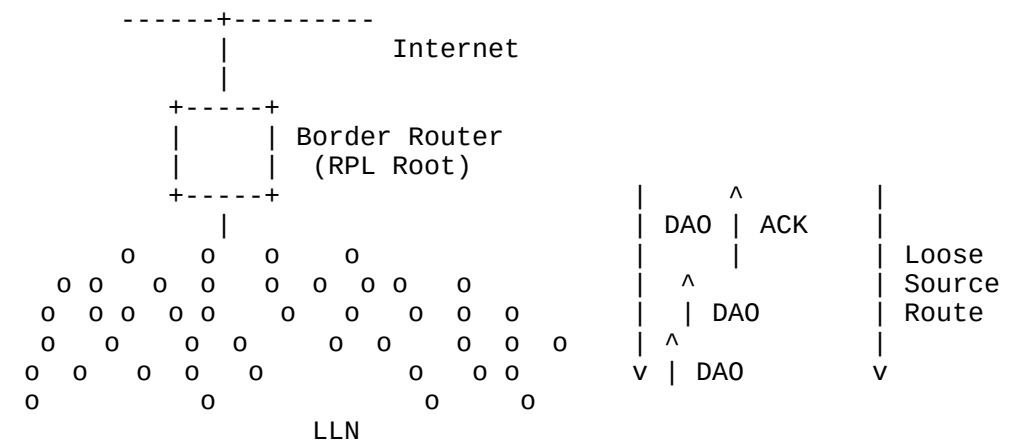

Figure 2: Non-Storing with Projected routes

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.

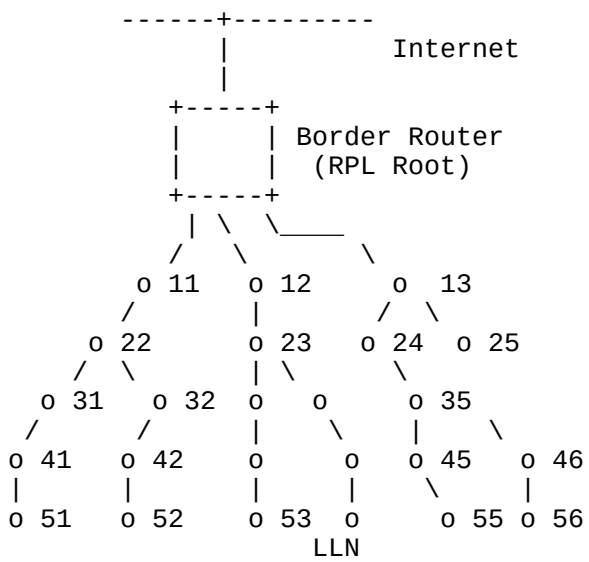

Figure 3: Non-Storing with Projected routes

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.

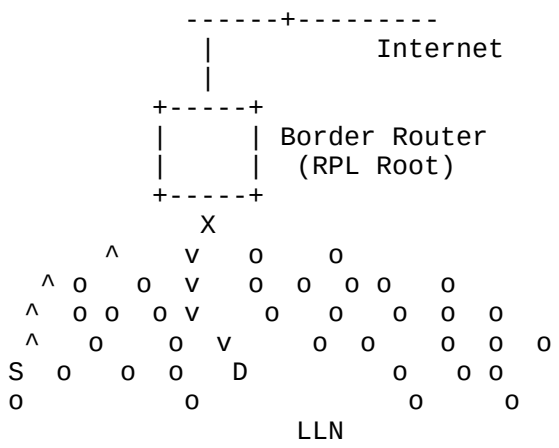

Figure 4: Routing Stretch

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.

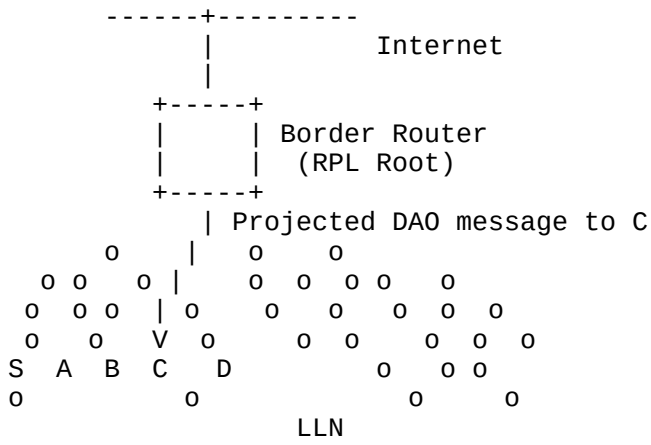

Figure 5: Projected DAO from root

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

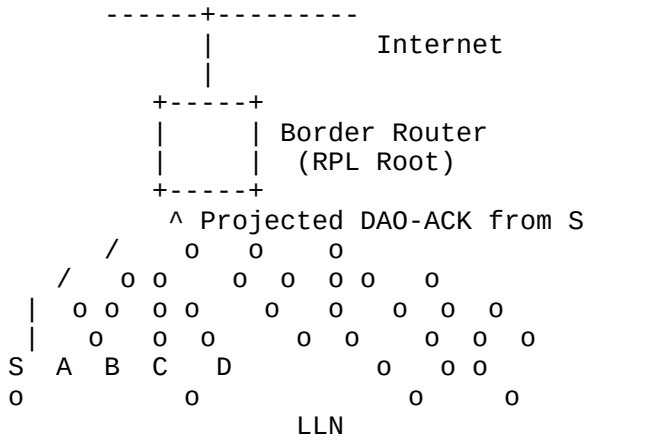

Figure 6: Projected DAO-ACK to root

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

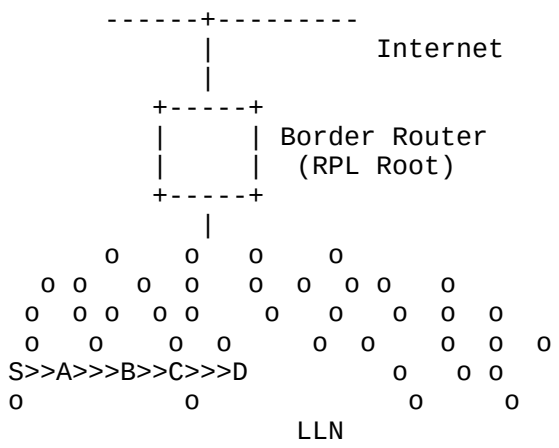

Figure 7: Optimized Projected Route

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., Thubert, P., and M. Teener, "Deterministic Networking Architecture", draft-finn-detnet-architecture-02 (work in progress), November 2015.

[I-D.ietf-6tisch-architecture] Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-08 (work in progress), May 2015.

[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