

ROLL Working Group  
INTERNET-DRAFT  
Intended Status: Standard Track  
Expires: September 13, 2017

J. Hou, Ed.  
R. Jadhav  
Z. Luo  
Huawei Technologies  
March 12, 2017

Optimization of Parent-node Selection in RPL-based Networks  
draft-hou-roll-rpl-parent-selection-00

Abstract

This document describes the problems in the DODAG construction of RPL-based network including "Thundering Herd" problem and Randomly Unbalanced Networking. The corresponding optimization methods are proposed to improve balancing the selection of parent nodes.

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). 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 September 13, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements Notation and Terminology . . . . .	3
2. DODAG Construction and Objective Functions . . . . .	3
3. Problems with Current DODAG construction . . . . .	4
3.1. Problem 1: "Thundering Herd" Phenomenon . . . . .	4
3.2. Problem 2: Randomly Unbalanced Network . . . . .	5
4. Optimized Solution . . . . .	6
4.1. Solution 1: Increment the ETX_Initial value . . . . .	6
4.2. Solution 2: Introducing the Child Node Count Metric . . . . .	7
5. Security Considerations . . . . .	9
6. IANA Considerations . . . . .	9
7. References . . . . .	9
7.1. Normative References . . . . .	9
7.2. Informative References . . . . .	10
8. Acknowledgments . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is a route-over distance vector routing protocol for networks in constrained conditions such as limited power and bandwidth. This protocol defines a Destination Oriented Directed Acyclic Graph (DODAG) to avoid the emergence of loops in the network [RFC 6550]. The routing procedure is based on an objective function (OF) including a set of metrics/constraints. The selection of the parent node in the RPL-based network directly influences the networking balance, and particularly, a poor balance causes the frequent switches of parent nodes. During the switching process, the delayed communication directly affects the stability of the entire network, so networking balance is an important indicator of the stability of mesh network.

In the RPL-based mesh network, due to the lack of balance algorithm, a large batch of nodes possibly select the same parent node and leave others empty, turning this focused parent node to be a forwarding point. A higher rate of transmission failure will result in a sharp increment in RANK, which triggers all child nodes to re-select and switch their parent node, and so on, causing frequent switching of the parent node. This phenomenon is particularly frequent at the beginning of networking that greatly decrements the efficiency of networking and communication stability. Therefore, the mechanism to select the parent node, making 6LoWPAN reach a balanced mesh network, is an urgent problem to be solved.

This document points out the problems associated with the RPL-based

networking, and proposes solutions for optimizing the balance of parent selection problems. Modifications are incrementing the RANK default value and introducing a new metric called "Child Node Count".

### 1.1. Requirements Notation and 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].

Below are the terms used in this document:

6LBR: 6Lo Border Router

6LN: 6Lo Node

CNC: Child Node Count

DAO: Destination Advertisement Object

DIO: DODAG Information Object

ETX: Expected Transmission Time

MOP: Mode of Operation

MRHOF: Minimum Rank with Hysteresis Objective Function

OP0: Objective Function Zero

RSSI: Received Signal Strength Indicator

## 2. DODAG Construction and Objective Functions

In general, an RPL-based network consists of three types of nodes: root node, connecting to another network as a gateway; router, forwarding topology information and data packets to their neighbors; leaf node, only joining a DODAG as an end member. The construction of a DODAG starts at the root node, through the routers, down to the leaf nodes. The root node broadcasts to its sub-nodes the DODAG Information Object (DIO) messages that contain the RANK information. Once receiving DIO messages, a sub-node chooses the node with the minimum RANK as its appropriate parent node. Afterwards, the routers will compute their own RANK according to the Objective Function (OF), update the DIO message, and forward to their neighbors.

Objective Function Zero (OF0) is designed as a default OF that allows

interoperation between implementations in a wide spectrum of use cases [RFC 6552]. OF0 specifies that a node calculates its own RANK  $R(N)$  according to the following equation:

$$R(N) = R(P) + \text{rank\_increase} \quad (1)$$

where  $\text{rank\_increase} = (R_f * S_p + S_r) * \text{MinHopRankIncrease}$  ( $R_f$ , Rank\_Factor;  $S_p$ , Step\_of\_Rank;  $S_r$ , Stretch\_of\_Rank).  $R(P)$  is the RANK of the parent node.

Apart from OF0, the Minimum Rank with Hysteresis Objective Function (MRHOF) is another OF that selects routes that minimize a metric. One common metric used in OF is the Expected Transmission Count (ETX), indicating the number of transmissions a node expects to make to a destination in order to successfully deliver a packet [RFC 6551]. When MRHOF uses ETX as its metric, nodes locally compute the ETX of links to its neighbors and add this value to their advertised Rank to compute the associated Rank of routes [RFC 6719]. In this case, the calculation of RANK can be simplified to be equation

$$R(N) = R(P) + \text{ETX}(N) * 128 \quad (2)$$

where  $\text{ETX}(N)$  is the ETX of links to its parent node. The ETX value normally follows an update formula

$$\text{ETX} = \text{ETX\_Old} * 0.9 + \text{ETX\_New} * 0.1 \quad (3)$$

where  $\text{ETX\_Old}$  is the previous ETX value, and  $\text{ETX\_New}$  is calculated by  $\text{ETX} = 1 / (D_f * D_r)$  in which  $D_f$  is the measured probability that a packet is received by the neighbor and  $D_r$  is the measured probability that the acknowledgment packet is successfully received [RFC 6551]. Once a node joins an RPL network with a default value  $\text{ETX}(N)$ , e.g.  $1 * R$ , the ETX gradually converges to the actual value after several times of update.

### 3. Problems with Current DODAG construction

#### 3.1. Problem 1: "Thundering Herd" Phenomenon

When a node joins the RPL-based network (such as 6LN\_New in Figure 1), the transmission path may be better than other nodes because the Default\_Rank\_Increase is  $3 * 256$ , calculated by equation (1) with the following default values specified in [RFC 6552]:

DEFAULT\_MIN\_HOP\_RANK\_INCREASE = 256

DEFAULT\_STEP\_OF\_RANK: 3

DEFAULT\_RANK\_STRETCH: 0

DEFAULT\_RANK\_FACTOR: 1

Suppose the RANK of 6LN in Figure 1 is much higher than  $4 \times 256$  and meets the requirement of parent switching when 6LN\_New joins. Then once 6LN\_New broadcasts the DIO messages including the RANK value of  $4 \times 256$  ( $1 \times 256$  for Root\_Rank;  $3 \times 256$  for Rank\_Increase), it may trigger a switch of numerous child nodes (circles in Figure 1) from their original parent nodes. Note that the dashed box depicted in Figure 1 is the common coverage of 6LN and 6LN\_New. So once a new node joins a network with a small RANK default value, it may suddenly attract numerous sub-nodes, impacting on the stability of the network. This phenomenon is called "Thundering Herd".

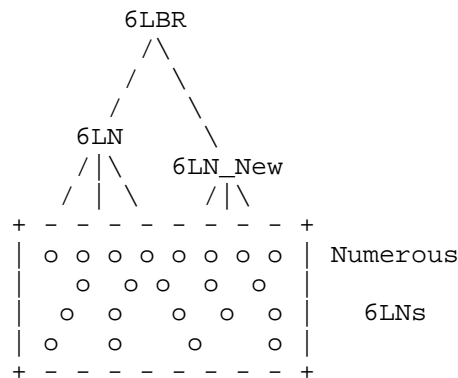


Figure 1: Example of Network Topology of "Thundering Herd"

### 3.2. Problem 2: Randomly Unbalanced Network

Although the RANK in DIO messages reflects various features of the network quality (e.g. Hop\_Count, Latency, ETX), there is another issue waiting to be solved: balancing the parent selection among nodes with the same RANK value. Currently in this case a node randomly chooses one as its parent node from the candidate parents with the same RANK, which gives the possibility of unbalanced networking. As depicted in Figure 2, the RANK of node A, B and C are supposed to be equal, but node B by chance dominates in the number of child nodes even though they share the common coverage (dashed square). It is a waste of resource that Node A is left empty while reachable for the sub-nodes. After a long period of time, the network only achieves a relative balance which is easy to be broken when new nodes join in.

This problem is particularly serious when the unbalanced networking

happens near the root node and above a large number of sub-nodes. Suppose in Figure 2 there are numerous child nodes connected to node D, E, F, G, and H, which put high load pressure on node B and C. In this case, overload and traffic block are likely to happen near the root, which further forces the network structure conversion. Node D, E, F may switch their parent node after network reconstruction but the best solution is achieving a balance in the parent selection at the beginning of the networking.

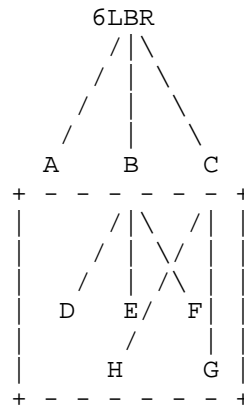


Figure 2: Example of Randomly Unbalanced Network Topology

#### 4. Optimized Solution

##### 4.1. Solution 1: Increment the ETX\_Initial value

In order to reduce the impact of the Thundering Herd phenomenon in the network, the `Default_Step_of_Rank` value defined in OF0 SHOULD per this document be incremented to

`DEFAULT_STEP_OF_RANK: 4`

So the `Default_Rank_Increase` is  $4 \times 256$ , and then based on the actual transmission result, the RANK gradually approaches the actual value. In this way, a new node joining the RPL network is initially set to be with a larger RANK default value. If transmissions through this node are of high quality, the RANK of this node will decrement gradually and attract child nodes one by one, which reduces the appearance of the Thundering Herd phenomenon. It is worthwhile noting that this solution is particularly applicable for the scenarios with numerous nodes and high node density.

This modification is also suitable for the MRHOF. Take ETX metric for an example: the default `ETX(N)` according to the modification

SHALL be set to 8, which gives  $R(N) = R(P) + 8 \cdot 128$  according to equation (2). The high initial RANK lowers the possibility of Thundering Herd phenomenon while afterwards the RANK gradually converges to the actual value according to equation (3).

#### 4.2. Solution 2: Introducing the Child Node Count Metric

In order to optimize the randomly unbalanced networking, this document proposes a method: introducing the number of child nodes as a new metric/constraint in the DAG Metric Container, which can be included in the Option field in the DIO message. The newly added information is 2 octets named by Child Node Count (CNC) which is per this document defined in the DAG Metric Container. The Routing Metric/Constraint Type is per this document RECOMMENDED to be value 9.

The Child Node Count (CNC) object is used to provide information related to the number of child nodes in the DIO source node, and may be used as a metric or as constraint.

The CNC object MAY be present in the DAG Metric Container. There MUST NOT be more than one CNC object as a constraint per DAG Metric Container, and there MUST NOT be more than one CNC object as a metric per DAG Metric Container.

The format of the CNC object body is as follows:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-----+
| (sub-object) .....
+-----+
```

Figure 3: Child Node Count Object Body Format

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+-----+-----+
|           CNC           |   MAX_CNC   |
+-----+-----+-----+
```

Figure 4: Child Node Count Sub-Object Format

CNC: 8 bits. The Child Node Count is encoded in 8 bits in unsigned integer format, expressed in number count, representing the number of child nodes.

MAX\_CNC: 8 bits. The Maximum Child Node Count is encoded in 8 bits

in unsigned integer format, expressed in number count, representing the maximum number of child nodes allowed in the neighbor cache.

The CNC field gives the number of child nodes while together with the MAX\_CNC field, sub-nodes are able to know the capacity of the candidate parent nodes, minimizing the need of NACK messages. The CNC object may be used as a constraint or a path metric. For example, one may want the CNC not to exceed some value. In this case, the CNC object common header indicates that the provided value relates to a constraint. In another example, the CNC object may be used as an aggregated additive metric where the value is updated along the path to reflect the number of child nodes along the path.

To solve the Randomly Unbalanced Network addressed in problem 2, this document proposes a optimization: using a combined metrics of ETX and CNC in the parent selection process. In this method, Prec field in the Routing Metric/Constraint Object is used with the following precedence: ETX > CNC, e.g. Prec of ETX is set to 0 and Prec of CNC is 1. In this case, the DAG Metric Container carries two Routing Metric/Constraint objects: one is an ETX metric object with header (C=0, O=0, A=00, R=0) and the second one is a Child Node Count object with header (C=0, O=0, A=00, R=0). Since Prec of ETX is set to be 0, which gives a higher priority, then nodes first choose the candidate parent nodes with the best link quality according to ETX. Afterwards, among the candidate parent nodes, the node with minimum CNC is selected to be the parent node. Note that this method is applicable for the storing mode of operation in which the nodes stores the information of their neighbors and thus are able to calculate the number of child nodes. In addition, when the candidate parent nodes contain both the same ETX and CNC, the child node SHOULD randomly choose one as the parent node among them.

This method is applicable for the storing mode of Mode of Operation (MOP) in which the Destination Advertisement Object (DAO) message is unicast from the child to the selected parent(s). For the non-storing mode, the CNC metric/constraint SHOULD NOT be used in the DIO messages, or if used, the CNC value MUST be set to 0. It is worthy to note that [draft-jadhav-lwig-nbr-mgmt-policy-00] gives a conceptual idea of this method in its section 2.5.3 while this document proposes the standard method to solve the unbalanced networking problem.

In the wireless networking scenarios, it would be better to take the wireless signal strength into consideration in the parent selection process, otherwise the optimally balanced network may lead to worse network communication quality for some child nodes. The wireless signal strength can be quantified by the Received Signal Strength Indicator (RSSI). This element can be added in the parent selection



process with the priority:  $ETX > RSSI > CNC$ . Note that the selection in the RSSI processes are not to choose the best value but to filter out candidate parent nodes according to the ranges (e.g.  $-80\text{dB} < RSSI < 20\text{dB}$ ). The RSSI value can be measured by the child nodes thus no additional modification is needed in the DAG Metric Container. So the parent node selection in wireless mesh networks can be processed as follows: Nodes in the network obtain the identity of candidate parent nodes, the number of child nodes, and the RANK. A filtering process takes place in the candidate parent nodes based on the identification and the RANK. Then these candidate parent nodes are filtered again according to the wireless communication quality index, RSSI. Finally, the candidate parent node with the minimum number of child nodes is determined as the target parent node in the network. Note that the RANK of each candidate parent node is the average value based on the current ETX, which is set to be greater than the default path coefficient R. The RANK of a node describes the total path to the border router (BR), which is determined by the RANK of the parent node together with the current ETX.

## 5. Security Considerations

This document has no security consideration beyond those in [RFC 6550], [RFC 6551], [RFC 6552] and [RFC 6719].

## 6. IANA Considerations

This document creates an IANA registry for the Routing Metric/Constraint Type. The assigned value is shown below in comparison with the existing values:

Value	Meaning	Reference
1	Node State and Attribute	RFC6551
2	Node Energy	RFC6551
3	Hop Count	RFC6551
4	Link Throughput	RFC6551
5	Link Latency	RFC6551
6	Link Quality Level	RFC6551
7	Link ETX	RFC6551
8	Link Color	RFC6551
9	Child Node Count(*)	This document

## 7. References

### 7.1. Normative References

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

10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

[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, 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, March 2012, <<http://www.rfc-editor.org/info/rfc6552>>.

[RFC6719] Gnawali, O. and P. Levis, "The Minimum Rank with Hysteresis Objective Function", RFC 6719, September 2012, <<http://www.rfc-editor.org/info/rfc6719>>.

## 7.2. Informative References

[draft-jadhav-lwig-nbr-mgmt-policy-00] Jadhav, R., Sahoo, R., Duquennoy, S., and J. Eriksson, "Neighbor Management Policy for 6LoWPAN", draft-jadhav-lwig-nbr-mgmt-policy-00, January 2017, <<https://tools.ietf.org/html/draft-jadhav-lwig-nbr-mgmt-policy-00>>.

[RFC6551] Vasseur, J., Kim, M., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012, <<http://www.rfc-editor.org/info/rfc6551>>.

## 8. Acknowledgments

Authors wish to thank Yizhou li and Yuefeng Wu for their valuable comments and contributions.

### Authors' Addresses

Jianqiang Hou (editor)  
Huawei Technologies  
101 Software Avenue,  
Nanjing 210012  
China

Phone: +86-15852944235  
Email: [houjianqiang@huawei.com](mailto:houjianqiang@huawei.com)

Rahul Arvind Jadhav

Huawei Technologies  
Kundalahalli Village, Whitefield,  
Bangalore, Karnataka 560037  
India

Phone: +91-080-49160700  
Email: rahul.ietf@gmail.com

Zhenhui Luo  
Huawei Technologies  
Bantian, Longgang District,  
Shenzhen 518129  
China

Phone: +86-18680331295  
Email: luozhenhui@huawei.com

ROLL  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

C.E. Perkins  
Lupin Lodge  
S.V.R.Anand  
Indian Institute of Science  
S. Anamalamudi  
SRM University-AP  
B. Liu  
Huawei Technologies  
7 March 2022

Supporting Asymmetric Links in Low Power Networks: AODV-RPL  
draft-ietf-roll-aodv-rpl-13

## Abstract

Route discovery for symmetric and asymmetric Peer-to-Peer (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 both hop-by-hop routing and source routing: Ad Hoc On-demand Distance Vector Routing (AODV) based RPL protocol (AODV-RPL). Paired Instances are used to construct directional paths, for cases where there are asymmetric links between source and target nodes.

## Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Overview of AODV-RPL . . . . .	6
4. AODV-RPL DIO Options . . . . .	7
4.1. AODV-RPL RREQ Option . . . . .	7
4.2. AODV-RPL RREP Option . . . . .	9
4.3. AODV-RPL Target Option . . . . .	10
5. Symmetric and Asymmetric Routes . . . . .	12
6. AODV-RPL Operation . . . . .	14
6.1. Route Request Generation . . . . .	14
6.2. Receiving and Forwarding RREQ messages . . . . .	14
6.2.1. Step 1: RREQ reception and evaluation . . . . .	15
6.2.2. Step 2: TargNode and Intermediate Router determination . . . . .	15
6.2.3. Step 3: Intermediate Router RREQ processing . . . . .	16
6.2.4. Step 4: Symmetric Route Processing at an Intermediate Router . . . . .	16
6.2.5. Step 5: RREQ propagation at an Intermediate Router . . . . .	17
6.2.6. Step 6: RREQ reception at TargNode . . . . .	17
6.3. Generating Route Reply (RREP) at TargNode . . . . .	17
6.3.1. RREP-DIO for Symmetric route . . . . .	18
6.3.2. RREP-DIO for Asymmetric Route . . . . .	18
6.3.3. RPLInstanceID Pairing . . . . .	18
6.4. Receiving and Forwarding Route Reply . . . . .	19
6.4.1. Step 1: Receiving and Evaluation . . . . .	19
6.4.2. Step 2: OrigNode or Intermediate Router . . . . .	19
6.4.3. Step 3: Build Route to TargNode . . . . .	20
6.4.4. Step 4: RREP Propagation . . . . .	20
7. Gratuitous RREP . . . . .	20
8. Operation of Trickle Timer . . . . .	21
9. IANA Considerations . . . . .	21
10. Security Considerations . . . . .	22
11. Acknowledgements . . . . .	23
12. References . . . . .	23
12.1. Normative References . . . . .	23
12.2. Informative References . . . . .	24

Appendix A. Example: Using ETX/RSSI Values to determine value of S	
bit . . . . .	25
Appendix B. Changelog . . . . .	27
B.1. Changes from version 12 to version 13 . . . . .	27
B.2. Changes from version 11 to version 12 . . . . .	28
B.3. Changes from version 10 to version 11 . . . . .	28
B.4. Changes from version 09 to version 10 . . . . .	29
B.5. Changes from version 08 to version 09 . . . . .	30
B.6. Changes from version 07 to version 08 . . . . .	30
B.7. Changes from version 06 to version 07 . . . . .	31
B.8. Changes from version 05 to version 06 . . . . .	31
B.9. Changes from version 04 to version 05 . . . . .	31
B.10. Changes from version 03 to version 04 . . . . .	32
B.11. Changes from version 02 to version 03 . . . . .	32
Appendix C. Contributors . . . . .	32
Authors' Addresses . . . . .	33

## 1. Introduction

Routing Protocol for Low-Power and Lossy Networks (RPL) [RFC6550] is an IPv6 distance vector routing protocol designed to support multiple traffic flows through a root-based Destination-Oriented Directed Acyclic Graph (DODAG). Typically, a router does not have routing information for most other routers. Consequently, for traffic between routers within the DODAG (i.e., Peer-to-Peer (P2P) traffic) data packets either have to traverse the root in non-storing mode, or traverse a common ancestor in storing mode. Such P2P traffic is thereby likely to traverse longer routes and may suffer severe congestion near the root (for more information see [RFC6997], [RFC6998]). The network environment that is considered in this document is assumed to be the same as described in Section 1 of [RFC6550].

The route discovery process in AODV-RPL is modeled on the analogous procedure specified in AODV [RFC3561]. The on-demand nature of AODV route discovery is natural for the needs of peer-to-peer routing in RPL-based LLNs. AODV terminology has been adapted for use with AODV-RPL messages, namely RREQ for Route Request, and RREP for Route Reply. AODV-RPL currently omits some features compared to AODV -- in particular, flagging Route Errors, "blacklisting" unidirectional links ([RFC3561]), multihoming, and handling unnumbered interfaces.

AODV-RPL reuses and extends the core RPL functionality to support routes with bidirectional asymmetric links. It retains RPL's DODAG formation, RPL Instance and the associated Objective Function (defined in [RFC6551]), trickle timers, and support for storing and non-storing modes. AODV-RPL adds basic messages RREQ and RREP as part of RPL DODAG Information Object (DIO) control message, which go

in separate (paired) RPL instances. AODV-RPL does not utilize the Destination Advertisement Object (DAO) control message of RPL. AODV-RPL uses the "P2P Route Discovery Mode of Operation" (MOP == 4) with three new Options for the DIO message, dedicated to discover P2P routes. These P2P routes may differ from routes discoverable by native RPL. Since AODV-RPL uses newly defined Options, there is no conflict with P2P-RPL [RFC6997], a previous document using the same MOP. AODV-RPL can be operated whether or not P2P-RPL or native RPL is running otherwise. For many networks AODV-RPL could be a replacement for RPL, since it can find better routes at very moderate extra cost. Consequently, it is unlikely that RPL would be needed in a network that is running AODV-RPL, even though it would be possible to run both protocols at the same time.

## 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

AODV-RPL reuses names for messages and data structures, including Rank, DODAG and DODAGID, as defined in RPL [RFC6550].

### AODV

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

### Asymmetric Route

The route from the OrigNode to the TargNode can traverse different nodes than the route from the TargNode to the OrigNode. An asymmetric route may result from the asymmetry of links, such that only one direction of the series of links satisfies the Objective Function during route discovery.

### Bi-directional Asymmetric Link

A link that can be used in both directions but with different link characteristics.

### DIO

DODAG Information Object

### DODAG RREQ-Instance (or simply RREQ-Instance)

RPL Instance built using the DIO with RREQ option; used for transmission of control messages from OrigNode to TargNode, thus enabling data transmission from TargNode to OrigNode.

DODAG RREP-Instance (or simply RREP-Instance)

RPL Instance built using the DIO with RREP option; used for transmission of control messages from TargNode to OrigNode thus enabling data transmission from OrigNode to TargNode.

Downward Direction

The direction from the OrigNode to the TargNode.

Downward Route

A route in the downward direction.

hop-by-hop routing

Routing when each router stores routing information about the next hop.

on-demand routing

Routing in which a route is established only when needed.

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 route discovery process between OrigNode and TargNode.

P2P

Peer-to-Peer -- in other words, not constrained a priori to traverse a common ancestor.

reactive routing

Same as "on-demand" routing.

RREQ-DIO message

A DIO message containing the RREQ option. The RPLInstanceID in RREQ-DIO is assigned locally by the OrigNode. The RREQ-DIO message has a secure variant as noted in [RFC6550].

RREQ-InstanceID

The RPLInstanceID for the RREQ-Instance. This term is used to indicate the value of the RPLInstanceID as provided by OrigNode in the RREQ message. The RPLInstanceID in the RREP message along with the Delta value determines the associated RREQ-InstanceID.



**RREP-DIO message**

A DIO message containing the RREP option. OrigNode pairs the RPLInstanceID in RREP-DIO to the one in the associated RREQ-DIO message (i.e., the RREQ-InstanceID) as described in Section 6.3.2. The RREP-DIO message has a secure variant as noted in [RFC6550].

**Source routing**

A mechanism by which the source supplies the complete route towards the target node along with each data packet [RFC6550].

**Symmetric route**

The upstream and downstream routes traverse the same routers and over the same links.

**TargNode**

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

**Upward Direction**

The direction from the TargNode to the OrigNode.

**Upward Route**

A route in the upward direction.

**ART option**

AODV-RPL Target option: a target option defined in this document.

### 3. Overview of AODV-RPL

With AODV-RPL, routes from OrigNode to TargNode within the LLN network are established "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. AODV-RPL works without requiring the use of RPL or any other routing protocol.

The routes discovered by AODV-RPL are not constrained to traverse a common ancestor. 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. AODV-RPL also enables discovery of symmetric routes along Paired DODAGs, when symmetric routes are possible (see Section 5).

In AODV-RPL, routes are discovered by first forming a temporary DAG rooted at the OrigNode. Paired DODAGs (Instances) are constructed 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. Intermediate routers join the DODAGs based on the Rank [RFC6550] as calculated from the DIO message. Henceforth in this document, "RREQ-DIO message" means the DIO message from OrigNode toward TargNode, containing the RREQ option as specified in Section 4.1. Similarly, "RREP-DIO message" means the DIO message from TargNode toward OrigNode, containing the RREP option as specified in Section 4.2. The route discovered in the RREQ-Instance is used for transmitting data from TargNode to OrigNode, and the route discovered in RREP-Instance is used for transmitting data from OrigNode to TargNode.

#### 4. AODV-RPL DIO Options

##### 4.1. AODV-RPL RREQ Option

OrigNode selects one of its IPv6 addresses and sets it in the DODAGID field of the RREQ-DIO message. Exactly one RREQ option MUST be present in a RREQ-DIO message, otherwise the message MUST be dropped.

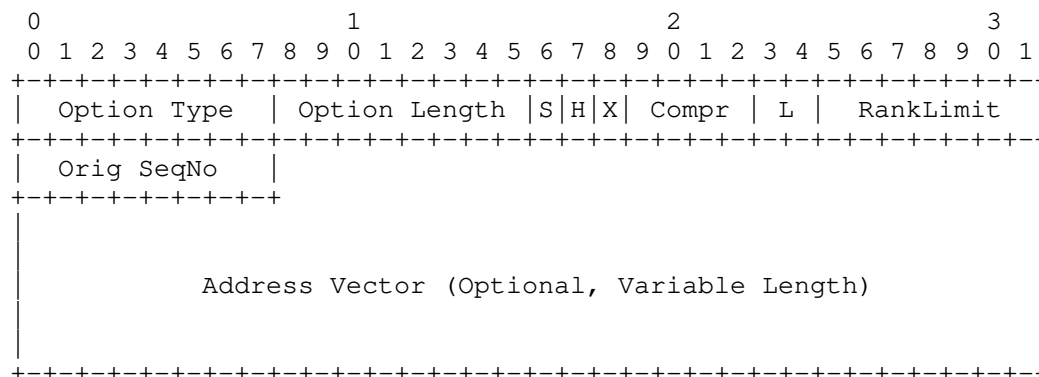


Figure 1: Format for AODV-RPL RREQ Option

OrigNode supplies the following information in the RREQ option:

Option Type  
TBD2

Option Length  
The length of the option in octets, excluding the Type and Length fields. Variable due to the presence of the address vector and the number of octets elided according to the Compr value.

S

Symmetric bit indicating a symmetric route from the OrigNode to the router transmitting this RREQ-DIO. See Section 5.

H

Set to one for a hop-by-hop route. Set to zero for a source route. This flag controls both the downstream route and upstream route.

X

Reserved; MUST be initialized to zero and ignored upon reception.

Compr

4-bit unsigned integer. When Compr is nonzero, exactly that number of prefix octets MUST be elided from each address before storing it in the Address Vector. The octets elided are shared with the IPv6 address in the DODAGID. This field is only used in source routing mode (H=0). In hop-by-hop mode (H=1), this field MUST be set to zero and ignored upon reception.

L

2-bit unsigned integer determining the length of time that a node is able to belong to the RREQ-Instance (a temporary DAG including the OrigNode and the TargNode). Once the time is reached, a node MUST leave the RREQ-Instance and stop sending or receiving any more DIOs for the RREQ-Instance. This naturally depends on the node's ability to keep track of time. Once a node leaves an RREQ-Instance, it MUST NOT rejoin the same RREQ-Instance. L is independent from the route lifetime, which is defined in the DODAG configuration option.

- \* 0x00: No time limit imposed.
- \* 0x01: 16 seconds
- \* 0x02: 64 seconds
- \* 0x03: 256 seconds

RankLimit

This field indicates the upper limit on the integer portion of the Rank (calculated using the DAGRank() macro defined in [RFC6550]). A value of 0 in this field indicates the limit is infinity.

Orig SeqNo

Sequence Number of OrigNode. See Section 6.1.

#### Address Vector

A vector of IPv6 addresses representing the route that the RREQ-DIO has passed. It is only present when the H bit is set to 0. The prefix of each address is elided according to the Compr field.

TargNode can join the RREQ instance at a Rank whose integer portion is less than or equal to the RankLimit. Any other node MUST NOT join a RREQ instance if its own Rank would be equal to or higher than RankLimit. A router MUST discard a received RREQ if the integer part of the advertised Rank equals or exceeds the RankLimit.

#### 4.2. AODV-RPL RREP Option

TargNode sets one of its IPv6 addresses in the DODAGID field of the RREP-DIO message. Exactly one RREP option MUST be present in a RREP-DIO message, otherwise the message MUST be dropped. TargNode supplies the following information in the RREP option:

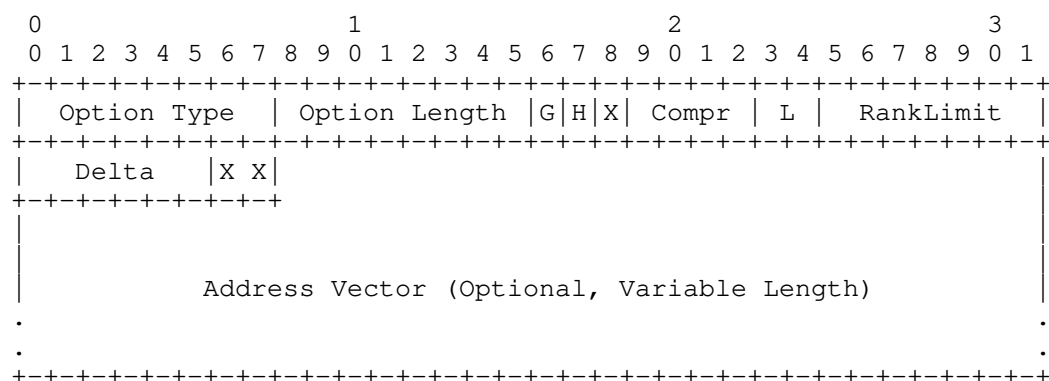


Figure 2: Format for AODV-RPL RREP option

#### Option Type

TBD3

#### Option Length

The length of the option in octets, excluding the Type and Length fields. Variable due to the presence of the address vector and the number of octets elided according to the Compr value.

#### G

Gratuitous route (see Section 7).

- H  
The H bit in the RREP option MUST be set to be the same as the H bit in RREQ option. It requests either source routing (H=0) or hop-by-hop (H=1) for the downstream route.
- X  
Reserved; MUST be initialized to zero and ignored upon reception.
- Compr  
4-bit unsigned integer. Same definition as in RREQ option.
- L  
2-bit unsigned integer defined as in RREQ option. The lifetime of the RREP-Instance MUST be shorter than the lifetime of the RREQ-Instance to which it is paired.
- RankLimit  
Similarly to RankLimit in the RREQ message, this field indicates the upper limit on the integer portion of the Rank. A value of 0 in this field indicates the limit is infinity.
- Delta  
6-bit unsigned integer. This field is used to recover the RREQ-InstanceID (see Section 6.3.3); 0 indicates that the RREQ-InstanceID has the same value as the RPLInstanceID of the RREP message.
- X X  
Reserved; MUST be initialized to zero and ignored upon reception.
- Address Vector  
Only present when the H bit is set to 0. For an asymmetric route, the Address Vector represents the IPv6 addresses of the path through the network the RREP-DIO has passed. For a symmetric route, it is the Address Vector when the RREQ-DIO arrives at the TargNode, unchanged during the transmission to the OrigNode.

#### 4.3. AODV-RPL Target Option

The AODV-RPL Target (ART) Option is based on the Target Option in core RPL [RFC6550]. The Flags field is replaced by the Destination Sequence Number of the TargNode and the Prefix Length field is reduced to 7 bits so that the value is limited to be no greater than 127.

A RREQ-DIO message MUST carry at least one ART Option. A RREP-DIO message MUST carry exactly one ART Option. Otherwise, the message MUST be dropped.

OrigNode can include multiple TargNode addresses via multiple AODV-RPL Target Options in the RREQ-DIO, for routes that share the same requirement on metrics. This reduces the cost to building only one DODAG.

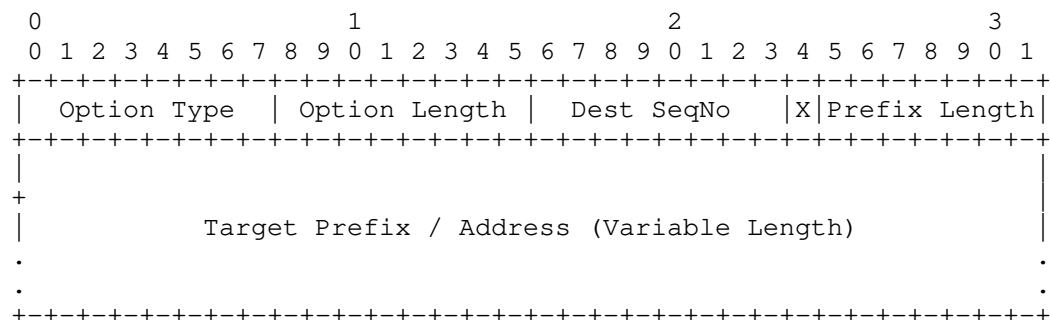


Figure 3: ART Option format for AODV-RPL

Option Type

TBD4

Option Length

Length of the option in octets excluding the Type and Length fields.

Dest SeqNo

In RREQ-DIO, if nonzero, it is the Sequence Number for the last route that OrigNode stored to the TargNode for which a route is desired. In RREP-DIO, it is the destination sequence number associated to the route. Zero is used if there is no known information about the sequence number of TargNode, and not used otherwise.

X

A one-bit reserved field. This field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Prefix Length

7-bit unsigned integer. Number of valid leading bits in the IPv6 Prefix. If Prefix Length is 0, then the value in the Target Prefix / Address field represents an IPv6 address, not a prefix.

Target Prefix / Address

(variable-length field) An IPv6 destination address or prefix. The Prefix Length field contains the number of valid leading bits

in the prefix. The Target Prefix / Address field contains the least number of octets that can represent all of the bits of the Prefix, in other words  $\text{Ceil}(\text{Prefix Length}/8)$  octets. The initial bits in the Target Prefix / Address field preceding the prefix length (if any) MUST be set to zero on transmission and MUST be ignored on receipt. If Prefix Length is zero, the Address field is 128 bits for IPv6 addresses.

## 5. Symmetric and Asymmetric Routes

Links are considered symmetric until indication to the contrary is received. In Figure 4 and Figure 5, BR is the Border Router, O is the OrigNode, each R is an intermediate router, and T is the TargNode. If the RREQ-DIO arrives over an interface that is known to be symmetric, and the S bit is set to 1, then it remains as 1, as illustrated in Figure 4. If an intermediate router sends out RREQ-DIO with the S bit set to 1, then each link en route from the OrigNode O to this router has met the requirements of route discovery, and the route can be used symmetrically.

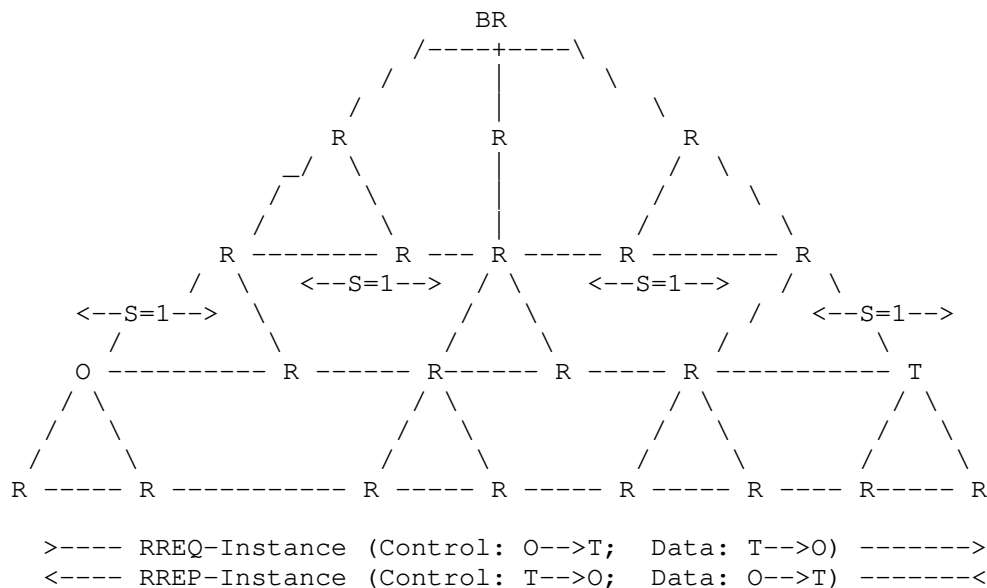


Figure 4: AODV-RPL with Symmetric Instances

Upon receiving a RREQ-DIO with the S bit set to 1, a node determines whether this link can be used symmetrically, i.e., both directions meet the requirements of data transmission. If the RREQ-DIO arrives over an interface that is not known to be symmetric, or is known to be asymmetric, the S bit is set to 0. If the S bit arrives already

set to be '0', it is set to be '0' when the RREQ-DIO is propagated (Figure 5). For an asymmetric route, there is at least one hop which doesn't satisfy the Objective Function. Based on the S bit received in RREQ-DIO, TargNode T determines whether or not the route is symmetric before transmitting the RREP-DIO message upstream towards the OrigNode O.

It is beyond the scope of this document to specify the criteria used when determining whether or not each link is symmetric. As an example, intermediate routers can use local information (e.g., bit rate, bandwidth, number of cells used in 6tisch [RFC9030]), a priori knowledge (e.g., link quality according to previous communication) or use averaging techniques as appropriate to the application. Other link metric information can be acquired before AODV-RPL operation, by executing evaluation procedures; for instance test traffic can be generated between nodes of the deployed network. During AODV-RPL operation, OAM techniques for evaluating link state (see [RFC7548], [RFC7276], [co-ioam]) MAY be used (at regular intervals appropriate for the LLN). The evaluation procedures are out of scope for AODV-RPL.

Appendix A describes an example method using the upstream Expected Number of Transmissions (ETX) and downstream Received Signal Strength Indicator (RSSI) to estimate whether the link is symmetric in terms of link quality using an averaging technique.

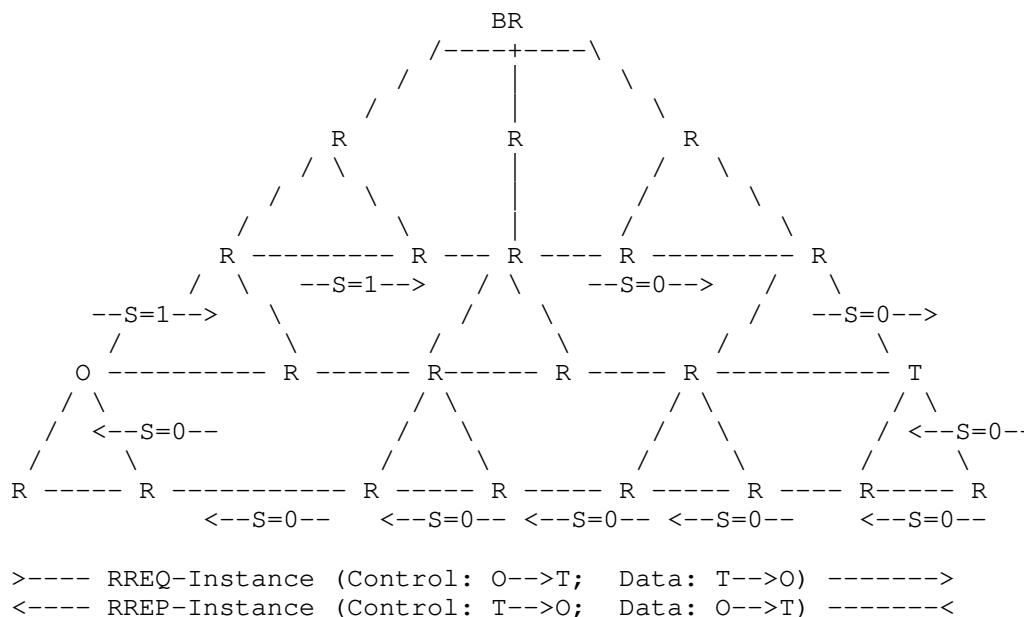




Figure 5: AODV-RPL with Asymmetric Paired Instances

As illustrated in Figure 5, an intermediate router determines the S bit value that the RREQ-DIO should carry using link asymmetry detection methods as discussed earlier in this section. In many cases the intermediate router has already made the link asymmetry decision by the time RREQ-DIO arrives.

## 6. AODV-RPL Operation

### 6.1. Route Request Generation

The route discovery process is initiated when an application at the OrigNode has data to be transmitted to the TargNode, but does not have a route that satisfies the Objective Function for the target of the application's data. In this case, the OrigNode builds a local RPLInstance and a DODAG rooted at itself. Then it transmits a DIO message containing exactly one RREQ option (see Section 4.1) to multicast group all-RPL-nodes. The DIO MUST contain at least one ART Option (see Section 4.3), which indicates the TargNode. The S bit in RREQ-DIO sent out by the OrigNode is set to 1.

Each node maintains a sequence number; the operation is specified in section 7.2 of [RFC6550]. When the OrigNode initiates a route discovery process, it MUST increase its own sequence number to avoid conflicts with previously established routes. The sequence number is carried in the Orig SeqNo field of the RREQ option.

The address in the ART Option can be a unicast IPv6 address or a prefix. The OrigNode can initiate the route discovery process for multiple targets simultaneously by including multiple ART Options. Within a RREQ-DIO the Objective Function for the routes to different TargNodes MUST be the same.

OrigNode can maintain different RPLInstances to discover routes with different requirements to the same targets. Using the RPLInstanceID pairing mechanism (see Section 6.3.3), route replies (RREP-DIOs) for different RPLInstances can be generated.

The transmission of RREQ-DIO obeys the Trickle timer [RFC6206]. If the length of time specified by the L field has elapsed, the OrigNode MUST leave the DODAG and stop sending RREQ-DIOs in the related RPLInstance.

### 6.2. Receiving and Forwarding RREQ messages

#### 6.2.1. Step 1: RREQ reception and evaluation

When a router X receives a RREQ message over a link from a neighbor Y, X first determines whether or not the RREQ is valid. If so, X then determines whether or not it has sufficient resources available to maintain the state needed to process an eventual RREP if the RREP were to be received. If not, then X MUST drop the packet and discontinue processing of the RREQ. Otherwise, X next determines whether the RREQ advertises a usable route to OrigNode, by checking whether the link to Y can be used to transmit packets to OrigNode.

When H=0 in the incoming RREQ, the router MUST drop the RREQ-DIO if one of its addresses is present in the Address Vector. When H=1 in the incoming RREQ, the router MUST drop the RREQ message if OrigSeqNo field of the RREQ is older than the SeqNo value that X has stored for a route to OrigNode. Otherwise, the router determines whether to propagate the RREQ-DIO. It does this by determining whether or not a route to OrigNode using the upstream direction of the incoming link satisfies the Objective Function (OF). In order to evaluate the OF, the router first determines the maximum useful rank (MaxUsefulRank). If the router has previously joined the RREQ-Instance associated with the RREQ-DIO, then MaxUsefulRank is set to be the Rank value that was stored when the router processed the best previous RREQ for the DODAG with the given RREQ-Instance. Otherwise, MaxUsefulRank is set to be RankLimit. If OF cannot be satisfied (i.e., the Rank evaluates to a value greater than MaxUsefulRank) the RREQ-DIO MUST be dropped, and the following steps are not processed. Otherwise, the router MUST join the RREQ-Instance and prepare to propagate the RREQ-DIO, as follows. The upstream neighbor router that transmitted the received RREQ-DIO is selected as the preferred parent.

#### 6.2.2. Step 2: TargNode and Intermediate Router determination

After determining that a received RREQ provides a usable route to OrigNode, a router determines whether it is a TargNode, or a possible intermediate router between OrigNode and a TargNode, or both. The router is a TargNode if it finds one of its own addresses in a Target Option in the RREQ. After possibly propagating the RREQ according to the procedures in Steps 3, 4, and 5, the TargNode generates a RREP as specified in Section 6.3.

If the OrigNode tries to reach multiple TargNodes in a single RREQ-Instance, one of the TargNodes can be an intermediate router to other TargNodes. In this case, before transmitting the RREQ-DIO to multicast group all-RPL-nodes, a TargNode MUST delete the Target Option encapsulating its own address, so that downstream routers with higher Rank values do not try to create a route to this TargNode.

An intermediate router could receive several RREQ-DIOs from routers with lower Rank values in the same RREQ-Instance with different lists of Target Options. For the purposes of determining the intersection with previous incoming RREQ-DIOs, the intermediate router maintains a record of the targets that have been requested for a given RREQ-Instance. An incoming RREQ-DIO message having multiple ART Options coming from a router with higher Rank than the Rank of the stored targets is ignored. When transmitting the RREQ-DIO, the intersection of all received lists MUST be included if it is nonempty after TargNode has deleted the Target Option encapsulating its own address. If the intersection is empty, it means that all the targets have been reached, and the router MUST NOT transmit any RREQ-DIO. Otherwise it proceeds to Section 6.2.3.

For example, suppose two RREQ-DIOs are received with the same RPLInstance and OrigNode. Suppose further that the first RREQ has (T1, T2) as the targets, and the second one has (T2, T4) as targets. Then only T2 needs to be included in the generated RREQ-DIO.

#### 6.2.3. Step 3: Intermediate Router RREQ processing

The intermediate router establishes itself as a viable node for a route to OrigNode as follows. If the H bit is set to 1, for hop-by-hop routing, then the router MUST build or update its upward route entry towards OrigNode, which includes at least the following items: Source Address, RPLInstanceID, Destination Address, Next Hop, Lifetime, and Sequence Number. The Destination Address and the RPLInstanceID respectively can be learned from the DODAGID and the RPLInstanceID of the RREQ-DIO. The Source Address is the address used by the router to send data to the Next Hop, i.e., the preferred parent. The lifetime is set according to DODAG configuration (not the L field) and can be extended when the route is actually used. The sequence number represents the freshness of the route entry; it is copied from the Orig SeqNo field of the RREQ option. A route entry with the same source and destination address, same RPLInstanceID, but stale sequence number, MUST be deleted.

#### 6.2.4. Step 4: Symmetric Route Processing at an Intermediate Router

If the S bit of the incoming RREQ-DIO is 0, then the route cannot be symmetric, and the S bit of the RREQ-DIO to be transmitted is set to 0. Otherwise, the router MUST determine whether the downward (i.e., towards the TargNode) direction of the incoming link satisfies the OF. If so, the S bit of the RREQ-DIO to be transmitted is set to 1. Otherwise the S bit of the RREQ-DIO to be transmitted is set to 0.

When a router joins the RREQ-Instance, it also associates within its data structure for the RREQ-Instance the information about whether or not the RREQ-DIO to be transmitted has the S-bit set to 1. This information associated to RREQ-Instance is known as the S-bit of the RREQ-Instance. It will be used later during the RREP-DIO message processing Section 6.3.2.

Suppose a router has joined the RREQ-Instance, and  $H=0$ , and the S-bit of the RREQ-Instance is set to 1. In this case, the router MAY optionally associate to the RREQ-Instance, the Address Vector of the symmetric route back to OrigNode. This is useful if the router later receives an RREP-DIO that is paired with the RREQ.

#### 6.2.5. Step 5: RREQ propagation at an Intermediate Router

If the router is an intermediate router, then it transmits the RREQ-DIO to the multicast group all-RPL-nodes; if the H bit is set to 0, the intermediate router MUST append the address of its interface receiving the RREQ-DIO into the address vector.

#### 6.2.6. Step 6: RREQ reception at TargNode

If the router is a TargNode and was already associated with the RREQ-Instance, it takes no further action and does not send an RREP-DIO. If TargNode is not already associated with the RREQ-Instance, it prepares and transmits a RREP-DIO, possibly after waiting for RREP\_WAIT\_TIME, as detailed in (Section 6.3).

#### 6.3. Generating Route Reply (RREP) at TargNode

When a TargNode receives a RREQ message over a link from a neighbor Y, TargNode first follows the procedures in Section 6.2. If the link to Y can be used to transmit packets to OrigNode, TargNode generates a RREP according to the steps below. Otherwise TargNode drops the RREQ and does not generate a RREP.

If the L field is not 0, the TargNode MAY delay transmitting the RREP-DIO for duration RREP\_WAIT\_TIME to await a route with a lower Rank. The value of RREP\_WAIT\_TIME is set by default to 1/4 of the duration determined by the L field. For  $L == 0$ , RREP\_WAIT\_TIME is set by default to 0. Depending upon the application, RREP\_WAIT\_TIME may be set to other values. Smaller values enable quicker formation for the P2P route. Larger values enable formation of P2P routes with better Rank values.

The address of the OrigNode MUST be encapsulated in the ART Option and included in this RREP-DIO message along with the SeqNo of TargNode.

### 6.3.1. RREP-DIO for Symmetric route

If the RREQ-Instance corresponding to the RREQ-DIO that arrived at TargNode has the S bit set to 1, there is a symmetric route both of whose directions satisfy the Objective Function. Other RREQ-DIOs might later provide better upward routes. The method of selection between a qualified symmetric route and an asymmetric route that might have better performance is implementation-specific and out of scope.

For a symmetric route, the RREP-DIO message is unicast to the next hop according to the Address Vector (H=0) or the route entry (H=1); the DODAG in RREP-Instance does not need to be built. The RPLInstanceID in the RREP-Instance is paired as defined in Section 6.3.3. In case the H bit is set to 0, the address vector from the RREQ-DIO MUST be included in the RREP-DIO.

### 6.3.2. RREP-DIO for Asymmetric Route

When a RREQ-DIO arrives at a TargNode with the S bit set to 0, the TargNode MUST build a DODAG in the RREP-Instance corresponding to the RREQ-DIO rooted at itself, in order to provide OrigNode with a downstream route to the TargNode. The RREP-DIO message is transmitted to multicast group all-RPL-nodes.

### 6.3.3. RPLInstanceID Pairing

Since the RPLInstanceID is assigned locally (i.e., there is no coordination between routers in the assignment of RPLInstanceID), the tuple (OrigNode, TargNode, RPLInstanceID) is needed to uniquely identify a discovered route. It is possible that multiple route discoveries with dissimilar Objective Functions are initiated simultaneously. Thus between the same pair of OrigNode and TargNode, there can be multiple AODV-RPL route discovery instances. So that OrigNode and Targnode can avoid any mismatch, they MUST pair the RREQ-Instance and the RREP-Instance in the same route discovery by using the RPLInstanceID.

When preparing the RREP-DIO, a TargNode could find the RPLInstanceID candidate for the RREP-Instance is already occupied by another RPL Instance from an earlier route discovery operation which is still active. This unlikely case might happen if two distinct OrigNodes need routes to the same TargNode, and they happen to use the same RPLInstanceID for RREQ-Instance. In such cases, the RPLInstanceID of an already active RREP-Instance MUST NOT be used again for assigning RPLInstanceID for the later RREP-Instance. Reusing the same RPLInstanceID for two distinct DODAGs originated with the same DODAGID (TargNode address) would prevent intermediate routers from

distinguishing between these DODAGs (and their associated Objective Functions). Instead, the RPLInstanceID MUST be replaced by another value so that the two RREP-instances can be distinguished. In the RREP-DIO option, the Delta field of the RREP-DIO message (Figure 2) indicates the increment to be applied to the pre-existing RPLInstanceID to obtain the value of the RPLInstanceID that is used in the RREP-DIO message. When the new RPLInstanceID after incrementation exceeds 255, it rolls over starting at 0. For example, if the RREQ-InstanceID is 252, and incremented by 6, the new RPLInstanceID will be 2. Related operations can be found in Section 6.4. RPLInstanceID collisions do not occur across RREQ-DIOs; the DODAGID equals the OrigNode address and is sufficient to disambiguate between DODAGs.

#### 6.4. Receiving and Forwarding Route Reply

Upon receiving a RREP-DIO, a router which already belongs to the RREP-Instance SHOULD drop the DIO. Otherwise the router performs the steps in the following subsections.

##### 6.4.1. Step 1: Receiving and Evaluation

If the Objective Function is not satisfied, the router MUST NOT join the DODAG; the router MUST discard the RREP-DIO, and does not execute the remaining steps in this section. An Intermediate Router MUST discard a RREP if one of its addresses is present in the Address Vector, and does not execute the remaining steps in this section.

If the S bit of the associated RREQ-Instance is set to 1, the router MUST proceed to Section 6.2.2.

If the S-bit of the RREQ-Instance is set to 0, the router MUST determine whether the downward direction of the link (towards the TargNode) over which the RREP-DIO is received satisfies the Objective Function, and the router's Rank would not exceed the RankLimit. If so, the router joins the DODAG of the RREP-Instance. The router that transmitted the received RREP-DIO is selected as the preferred parent. Afterwards, other RREP-DIO messages can be received.

##### 6.4.2. Step 2: OrigNode or Intermediate Router

The router updates its stored value of the TargNode's sequence number according to the value provided in the ART option. The router next checks if one of its addresses is included in the ART Option. If so, this router is the OrigNode of the route discovery. Otherwise, it is an intermediate router.

#### 6.4.3. Step 3: Build Route to TargNode

If the H bit is set to 1, then the router (OrigNode or intermediate) MUST build a downward route entry towards TargNode which includes at least the following items: OrigNode Address, RPLInstanceID, TargNode Address as destination, Next Hop, Lifetime and Sequence Number. For a symmetric route, the Next Hop in the route entry is the router from which the RREP-DIO is received. For an asymmetric route, the Next Hop is the preferred parent in the DODAG of RREP-Instance. The RPLInstanceID in the route entry MUST be the RREQ-InstanceID (i.e., after subtracting the Delta field value from the value of the RPLInstanceID). The source address is learned from the ART Option, and the destination address is learned from the DODAGID. The lifetime is set according to DODAG configuration (i.e., not the L field) and can be extended when the route is actually used. The sequence number represents the freshness of the route entry, and is copied from the Dest SeqNo field of the ART option of the RREP-DIO. A route entry with same source and destination address, same RPLInstanceID, but stale sequence number (i.e., incoming sequence number is less than the currently stored sequence number of the route entry), MUST be deleted.

#### 6.4.4. Step 4: RREP Propagation

If the receiver is the OrigNode, it can start transmitting the application data to TargNode along the path as provided in RREP-Instance, and processing for the RREP-DIO is complete. Otherwise, the RREP will be propagated towards OrigNode. If H=0, the intermediate router MUST include the address of the interface receiving the RREP-DIO into the address vector. If H=1, according to the last step the intermediate router has set up a route entry for TargNode. If the intermediate router has a route to OrigNode, it uses that route to unicast the RREP-DIO to OrigNode. Otherwise, in case of a symmetric route, the RREP-DIO message is unicast to the Next Hop according to the address vector in the RREP-DIO (H=0) or the local route entry (H=1). Otherwise, in case of an asymmetric route, the intermediate router transmits the RREP-DIO to multicast group all-RPL-nodes. The RPLInstanceID in the transmitted RREP-DIO is the same as the value in the received RREP-DIO.

### 7. Gratuitous RREP

In some cases, an Intermediate router that receives a RREQ-DIO message MAY transmit a "Gratuitous" RREP-DIO message back to OrigNode instead of continuing to multicast the RREQ-DIO towards TargNode. The intermediate router effectively builds the RREP-Instance on behalf of the actual TargNode. The G bit of the RREP option is provided to distinguish the Gratuitous RREP-DIO (G=1) sent by the

Intermediate router from the RREP-DIO sent by TargNode (G=0).

The gratuitous RREP-DIO MAY be sent out when an intermediate router receives a RREQ-DIO for a TargNode, and the router has a pair of downward and upward routes to the TargNode which also satisfy the Objective Function and for which the destination sequence number is at least as large as the sequence number in the RREQ-DIO message.

In case of source routing, the intermediate router MUST unicast the received RREQ-DIO to TargNode including the address vector between the OrigNode and the router. Thus the TargNode can have a complete upward route address vector from itself to the OrigNode. Then the router MUST include the address vector from the TargNode to the router itself in the gratuitous RREP-DIO to be transmitted.

In case of hop-by-hop routing, the intermediate router MUST unicast the received RREQ-DIO to the Next Hop on the route. The Next Hop router along the route MUST build new route entries with the related RPLInstanceID and DODAGID in the downward direction. The above process will happen recursively until the RREQ-DIO arrives at the TargNode. Then the TargNode MUST unicast recursively the RREP-DIO hop-by-hop to the intermediate router, and the routers along the route SHOULD build new route entries in the upward direction. Upon receiving the unicast RREP-DIO, the intermediate router sends the gratuitous RREP-DIO to the OrigNode as defined in Section 6.3.

## 8. Operation of Trickle Timer

RREQ-Instance/RREP-Instance multicast uses trickle timer operations [RFC6206] to control RREQ-DIO and RREP-DIO transmissions. The Trickle control of these DIO transmissions follows the procedures described in the Section 8.3 of [RFC6550] entitled "DIO Transmission". If the route is symmetric, the RREP DIO does not need the Trickle timer mechanism.

## 9. IANA Considerations

Note to RFC editor:

The sentence "The parenthesized numbers are only suggestions." is to be removed prior publication.

A Subregistry in this section refers to a named sub-registry of the "Routing Protocol for Low Power and Lossy Networks (RPL)" registry.

AODV-RPL uses the "P2P Route Discovery Mode of Operation" (MOP == 4) with new Options as specified in this document. Please cite AODV-RPL and this document as one of the protocols using MOP 4.



IANA is asked to assign three new AODV-RPL options "RREQ", "RREP" and "ART", as described in Figure 6 from the "RPL Control Message Options" Subregistry. The parenthesized numbers are only suggestions.

Value	Meaning	Reference
TBD2 (0x0B)	RREQ Option	This document
TBD3 (0x0C)	RREP Option	This document
TBD4 (0x0D)	ART Option	This document

Figure 6: AODV-RPL Options

## 10. Security Considerations

The security considerations for the operation of AODV-RPL are similar to those for the operation of RPL (as described in Section 19 of the RPL specification [RFC6550]). Sections 6.1 and 10 of [RFC6550] describe RPL's optional security framework, which AODV-RPL relies on to provide data confidentiality, authentication, replay protection, and delay protection services. Additional analysis for the security threats to RPL can be found in [RFC7416].

A router can join a temporary DAG created for a secure AODV-RPL route discovery only if it can support the security configuration in use (see Section 6.1 of [RFC6550]), which also specifies the key in use. It does not matter whether the key is preinstalled or dynamically acquired. The router must have the key in use before it can join the DAG being created for secure route discovery.

If a rogue router knows the key for the security configuration in use, it can join the secure AODV-RPL route discovery and cause various types of damage. Such a rogue router could advertise false information in its DIOs in order to include itself in the discovered route(s). It could generate bogus RREQ-DIO, and RREP-DIO messages carrying bad routes or maliciously modify genuine RREP-DIO messages it receives. A rogue router acting as the OrigNode could launch denial-of-service attacks against the LLN deployment by initiating fake AODV-RPL route discoveries. When rogue routers might be present, RPL's preinstalled mode of operation, where the key to use for route discovery is preinstalled, SHOULD be used.

When a RREQ-DIO message uses the source routing option by setting the H bit to 0, a rogue router may populate the Address Vector field with a set of addresses that may result in the RREP-DIO traveling in a routing loop.

If a rogue router is able to forge a gratuitous RREP, it could mount denial-of-service attacks.

## 11. Acknowledgements

The authors thank Pascal Thubert, Rahul Jadhav, and Lijo Thomas for their support and valuable inputs. The authors specially thank Lavanya H.M for implementing AODV-RPL in Contiki and conducting extensive simulation studies.

The authors would like to acknowledge the review, feedback and comments from the following people, in alphabetical order: Roman Danyliw, Lars Eggert, Benjamin Kaduk, Tero Kivinen, Erik Kline, Murray Kucherawy, Warren Kumari, Francesca Palombini, Alvaro Retana, Ines Robles, John Scudder, Meral Shirazipour, Peter Van der Stok, Eric Vyncke, and Robert Wilton.

## 12. References

### 12.1. 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, <<https://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, <<https://www.rfc-editor.org/info/rfc6206>>.
- [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>>.
- [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, <<https://www.rfc-editor.org/info/rfc6551>>.

- [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>>.

## 12.2. Informative References

- [co-ioam] Ballamajalu, Rashmi., S.V.R., Anand., and Malati Hegde, "Co-iOAM: In-situ Telemetry Metadata Transport for Resource Constrained Networks within IETF Standards Framework", 2018 10th International Conference on Communication Systems & Networks (COMSNETS) pp.573-576, January 2018.
- [contiki] Contiki contributors, "The Contiki Open Source OS for the Internet of Things (Contiki Version 2.7)", November 2013, <<https://github.com/contiki-os/contiki>>.
- [Contiki-ng] Contiki-NG contributors, "Contiki-NG: The OS for Next Generation IoT Devices (Contiki-NG Version 4.6)", December 2020, <<https://github.com/contiki-ng/contiki-ng>>.
- [cooja] Contiki/Cooja contributors, "Cooja Simulator for Wireless Sensor Networks (Contiki/Cooja Version 2.7)", November 2013, <<https://github.com/contiki-os/contiki/tree/master/tools/cooja>>.
- [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, <<https://www.rfc-editor.org/info/rfc3561>>.
- [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>>.
- [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, <<https://www.rfc-editor.org/info/rfc6998>>.

- [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <<https://www.rfc-editor.org/info/rfc7276>>.
- [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>>.
- [RFC7548] Ersue, M., Ed., Romascanu, D., Schoenwaelder, J., and A. Sehgal, "Management of Networks with Constrained Devices: Use Cases", RFC 7548, DOI 10.17487/RFC7548, May 2015, <<https://www.rfc-editor.org/info/rfc7548>>.
- [RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <<https://www.rfc-editor.org/info/rfc9030>>.

#### Appendix A. Example: Using ETX/RSSI Values to determine value of S bit

The combination of Received Signal Strength Indication(downstream) (RSSI) and Expected Number of Transmissions(upstream) (ETX) has been tested to determine whether a link is symmetric or asymmetric at intermediate routers. We present two methods to obtain an ETX value from RSSI measurement.

Method 1: In the first method, we constructed a table measuring RSSI

vs ETX using the Cooja simulation [cooja] setup in the Contiki OS environment[contiki]. We used Contiki-2.7 running 6LoWPAN/RPL protocol stack for the simulations. For approximating the number of packet drops based on the RSSI values, we implemented simple logic that drops transmitted packets with certain pre-defined ratios before handing over the packets to the receiver. The packet drop ratio is implemented as a table lookup of RSSI ranges mapping to different packet drop ratios with lower RSSI ranges resulting in higher values. While this table has been defined for the purpose of capturing the overall link behavior, it is highly recommended to conduct physical radio measurement experiments, in general. By keeping the receiving node at different distances, we let the packets experience different packet drops as per the described method. The ETX value computation is done by another module which is part of RPL Objective Function implementation. Since ETX value is reflective of the extent of packet drops, it allowed us to prepare a useful ETX vs RSSI table. ETX versus RSSI values obtained in this way may be used as explained below:

Source----->NodeA----->NodeB----->Destination

Figure 7: Communication link from Source to Destination

RSSI at NodeA for NodeB	Expected ETX at NodeA for NodeB->NodeA
> -60	150
-70 to -60	192
-80 to -70	226
-90 to -80	662
-100 to -90	3840

Table 1: Selection of S bit based on Expected ETX value

Method 2: One could also make use of the function `guess_etx_from_rssi()` defined in the 6LoWPAN/RPL protocol stack of Contiki-ng OS [Contiki-ng] to obtain RSSI-ETX mapping. This function outputs ETX value ranging between 128 and 3840 for  $-60 \leq \text{rssi} \leq -89$ . The function description is beyond the scope of this document.

We tested the operations in this specification by making the following experiment, using the above parameters. In our experiment, a communication link is considered as symmetric if the ETX value of NodeA->NodeB and NodeB->NodeA (see Figure 7) are within, say, a 1:3 ratio. This ratio should be understood as determining the link's symmetric/asymmetric nature. NodeA can typically know the ETX value in the direction of NodeA -> NodeB but it has no direct way of knowing the value of ETX from NodeB->NodeA. Using physical testbed experiments and realistic wireless channel propagation models, one can determine a relationship between RSSI and ETX representable as an expression or a mapping table. Such a relationship in turn can be used to estimate ETX value at nodeA for link NodeB--->NodeA from the received RSSI from NodeB. Whenever nodeA determines that the link towards the nodeB is bi-directional asymmetric then the S bit is set to 0. Afterwards, the link from NodeA to Destination remains designated as asymmetric and the S bit remains set to 0.

Determination of asymmetry versus bidirectionality remains a topic of lively discussion in the IETF.

## Appendix B. Changelog

Note to the RFC Editor: please remove this section before publication.

### B.1. Changes from version 12 to version 13

- \* Changed name of "Shift" field to be the "Delta" field.
- \* Specified that if a node does not have resources, it MUST drop the RREQ.
- \* Changed name of MaxUseRank to MaxUsefulRank.
- \* Revised a sentence that was not clear about when a TargNode can delay transmission of the RREP in response to a RREQ.
- \* Provided advice about running AODV-RPL at same time as P2P-RPL or native RPL.
- \* Small reorganization and enlargement of the description of Trickle time operation in Section 8.
- \* Added definition for "RREQ-InstanceID" to Terminology section.
- \* Specified that once a node leaves an RREQ-Instance, it MUST NOT rejoin the same RREQ-Instance.

## B.2. Changes from version 11 to version 12

- \* Defined RREP\_WAIT\_TIME for asymmetric as well as symmetric handling of RREP-DIO.
- \* Clarified link-local multicast transmission to use link-local multicast group all-RPL nodes.
- \* Identified some security threats more explicitly.
- \* Specified that the pairing between RREQ-DIO and RREP-DIO happens at OrigNode and TargNode. Intermediate routers do not necessarily maintain the pairing.
- \* When RREQ-DIO is received with H=0 and S=1, specified that intermediate routers MAY store symmetric Address Vector information for possible use when a machine RREP-DIO is received.
- \* Specified that AODV-RPL uses the "P2P Route Discovery Mode of Operation" (MOP == 4), instead of requesting the allocation of a new MOP. Clarified that there is no conflict with [RFC6997].
- \* Fixed several important typos and improved language in numerous places.
- \* Reorganized the steps in the specification for handling RREQ and RREP at an intermediate router, to more closely follow the order of processing actions to be taken by the router.

## B.3. Changes from version 10 to version 11

- \* Numerous editorial improvements.
- \* Replace  $\text{Floor}((7 + (\text{Prefix Length})) / 8)$  by  $\text{Ceil}(\text{Prefix Length} / 8)$  for simplicity and ease of understanding.
- \* Use "L field" instead of "L bit" since L is a two-bit field.
- \* Improved the procedures in section 6.2.1.
- \* Define the S bit of the data structure a router uses to represent whether or not the RREQ instance is for a symmetric or an asymmetric route. This replaces text in the document that was a holdover from earlier versions in which the RREP had an S bit for that purpose.

- \* Quote terminology from AODV that has been identified as possibly originating in language reflecting various kinds of bias against certain cultures.
- \* Clarified the relationship of AODV-RPL to RPL.
- \* Eliminated the "Point-to-Point" terminology to avoid suggesting only a single link.
- \* Modified certain passages to better reflect the possibility that a router might have multiple IP addresses.
- \* "Rsv" replaced by "X X" for reserved field.
- \* Added mandates for reserved fields, and replaces some ambiguous language phraseology by mandates.
- \* Replaced "retransmit" terminology by more correct "propagate" terminology.
- \* Added text about determining link symmetry near Figure 5.
- \* Mandated checking the Address Vector to avoid routing loops.
- \* Improved specification for use of the Delta value in Section 6.3.3.
- \* Corrected the wrong use of RREQ-Instance to be RREP-Instance.
- \* Referred to Subregistry values instead of Registry values in Section 9.
- \* Sharpened language in Section 10, eliminated misleading use of capitalization in the words "Security Configuration".
- \* Added acknowledgements and contributors.

#### B.4. Changes from version 09 to version 10

- \* Changed the title for brevity and to remove acronyms.
- \* Added "Note to the RFC Editor" in Section 9.
- \* Expanded DAO and P2MP in Section 1.
- \* Reclassified [RFC6998] and [RFC7416] as Informational.
- \* SHOULD changed to MUST in Section 4.1 and Section 4.2.



- \* Several editorial improvements and clarifications.

#### B.5. Changes from version 08 to version 09

- \* Removed section "Link State Determination" and put some of the relevant material into Section 5.
- \* Cited security section of [RFC6550] as part of the RREP-DIO message description in Section 2.
- \* SHOULD has been changed to MUST in Section 4.2.
- \* Expanded the terms ETX and RSSI in Section 5.
- \* Section 6.4 has been expanded to provide a more precise explanation of the handling of route reply.
- \* Added [RFC7416] in the Security Considerations (Section 10) for RPL security threats. Cited [RFC6550] for authenticated mode of operation.
- \* Appendix A has been mostly re-written to describe methods to determine whether or not the S bit should be set to 1.
- \* For consistency, adjusted several mandates from SHOULD to MUST and from SHOULD NOT to MUST NOT.
- \* Numerous editorial improvements and clarifications.

#### B.6. Changes from version 07 to version 08

- \* Instead of describing the need for routes to "fulfill the requirements", specify that routes need to "satisfy the Objective Function".
- \* Removed all normative dependencies on [RFC6997]
- \* Rewrote Section 10 to avoid duplication of language in cited specifications.
- \* Added a new section "Link State Determination" with text and citations to more fully describe how implementations determine whether links are symmetric.
- \* Modified text comparing AODV-RPL to other protocols to emphasize the need for AODV-RPL instead of the problems with the other protocols.

- \* Clarified that AODV-RPL uses some of the base RPL specification but does not require an instance of RPL to run.
- \* Improved capitalization, quotation, and spelling variations.
- \* Specified behavior upon reception of a RREQ-DIO or RREP-DIO message for an already existing DODAGID (e.g, Section 6.4).
- \* Fixed numerous language issues in IANA Considerations Section 9.
- \* For consistency, adjusted several mandates from SHOULD to MUST and from SHOULD NOT to MUST NOT.
- \* Numerous editorial improvements and clarifications.

#### B.7. Changes from version 06 to version 07

- \* Added definitions for all fields of the ART option (see Section 4.3). Modified definition of Prefix Length to prohibit Prefix Length values greater than 127.
- \* Modified the language from [RFC6550] Target Option definition so that the trailing zero bits of the Prefix Length are no longer described as "reserved".
- \* Reclassified [RFC3561] and [RFC6998] as Informative.
- \* Added citation for [RFC8174] to Terminology section.

#### B.8. Changes from version 05 to version 06

- \* Added Security Considerations based on the security mechanisms defined in [RFC6550].
- \* Clarified the nature of improvements due to P2P route discovery versus bidirectional asymmetric route discovery.
- \* Editorial improvements and corrections.

#### B.9. Changes from version 04 to version 05

- \* Add description for sequence number operations.
- \* Extend the residence duration L in section 4.1.
- \* Change AODV-RPL Target option to ART option.

## B.10. Changes from version 03 to version 04

- \* Updated RREP option format. Remove the T bit in RREP option.
- \* Using the same RPLInstanceID for RREQ and RREP, no need to update [RFC6550].
- \* Explanation of Delta field in RREP.
- \* Multiple target options handling during transmission.

## B.11. Changes from version 02 to version 03

- \* Include the support for source routing.
- \* Import some features from [RFC6997], e.g., choice between hop-by-hop and source routing, the L field which determines the duration of residence in the DAG, RankLimit, etc.
- \* Define new target option for AODV-RPL, including the Destination Sequence Number in it. Move the TargNode address in RREQ option and the OrigNode address in RREP option into ADOV-RPL Target Option.
- \* Support route discovery for multiple targets in one RREQ-DIO.
- \* New RPLInstanceID pairing mechanism.

## Appendix C. Contributors

Abdur Rashid Sangi

Huaiyin Institute of Technology

No.89 North Beijing Road, Qinghe District

Huaian 223001

P.R. China

Email: sangi\_bahrian@yahoo.com

Malati Hegde

Indian Institute of Science

Bangalore 560012

India

Email: malati@iisc.ac.in

Mingui Zhang

Huawei Technologies

No. 156 Beiqing Rd. Haidian District

Beijing 100095

P.R. China

Email: zhangmingui@huawei.com

#### Authors' Addresses

Charles E. Perkins

Lupin Lodge

Los Gatos, 95033

United States

Email: charliep@lupinlodge.com

S.V.R Anand

Indian Institute of Science

Bangalore 560012

India

Email: anandsvr@iisc.ac.in

Satish Anamalamudi

SRM University-AP

Amaravati Campus

Amaravati, Andhra Pradesh 522 502

India

Email: satishnaidu80@gmail.com

Bing Liu

Huawei Technologies

No. 156 Beiqing Rd. Haidian District

Beijing

100095

China

Email: remy.liubing@huawei.com

Roll Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 3, 2018

O. Bergmann  
C. Bormann  
S. Gerdes  
Universitaet Bremen TZI  
H. Chen  
Huawei Technologies  
October 30, 2017

Constrained-Cast: Source-Routed Multicast for RPL  
draft-ietf-roll-ccast-01

Abstract

This specification defines a protocol for forwarding multicast traffic in a constrained node network employing the RPL routing protocol in non-storing mode.

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, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	3
2. The BIER Approach . . . . .	3
3. The Constrained-Cast Approach . . . . .	3
4. False Positives . . . . .	4
5. Protocol . . . . .	4
5.1. Multicast Listener Advertisement Object (MLAO) . . . . .	4
5.2. Routing Header . . . . .	5
6. Implementation . . . . .	7
7. Benefits . . . . .	7
8. Security Considerations . . . . .	7
9. IANA Considerations . . . . .	8
9.1. ICMPv6 Parameter Registration . . . . .	8
9.2. Critical 6LowPAN Routing Header Type Registration . . . . .	8
10. Acknowledgments . . . . .	8
11. References . . . . .	8
11.1. Normative References . . . . .	8
11.2. Informative References . . . . .	9
Authors' Addresses . . . . .	9

## 1. Introduction

As defined in [RFC6550], RPL Multicast assumes that the RPL network operates in Storing Mode. Multicast DAOs are used to indicate subscription to multicast address to a parent; these DAOs percolate up and create bread-crumbs. This specification, although part of RFC 6550, appears to be incomplete and untested. More importantly, Storing Mode is not in use in constrained node networks outside research operating environments.

The present specification addresses multicast forwarding for RPL networks in the much more common Non-Storing Mode. Non-Storing is based on the root node adding source-routing information to downward packets. Evidently, to make this work, RPL multicast needs to source-route multicast packets. A source route here is a list of identifiers to instruct forwarders to relay the respective IP datagram.

As every forwarder in an IP-based constrained node network has at least one network interface, it is straight-forward to use the address of an outgoing interface as identifiers in this source-route. (Typically, this is a globally unique public address of the node's only network adapter.)

The source-route subsets the whole set of potential forwarders available in the RPL DODAG to those that need to forward in order to reach known multicast listeners.

Including an actual list of outgoing interfaces is rarely applicable, as this is likely to be a large list of 16-byte IPv6 addresses. Even with [RFC6554] style compression, the size of the list becomes prohibitively quickly.

### 1.1. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In this specification, the term "byte" is used in its now customary sense as a synonym for "octet".

All multi-byte integers in this protocol are interpreted in network byte order.

## 2. The BIER Approach

Bit-Indexed Explicit Replication [I-D.ietf-bier-architecture] lists all egress routers in a bitmap included in each multicast packet. This requires creating a mostly contiguous numbering of all egress routers; more importantly, BIER requires the presence of a network map in each forwarders to be able to interpret the bitmap and map it to a set of local outgoing interfaces.

## 3. The Constrained-Cast Approach

Constrained-Cast employs Bloom Filters [BLOOM] as a compact representation of a match or non-match for elements in a large set: Each element to be included is hashed with multiple hash functions; the result is used to index a bitmap and set the corresponding bit. To check for the presence of an element, the same hash functions are applied to obtain bit positions; if all corresponding bits are set, this is used to indicate a match. (Multiple hash functions are most easily obtained by adding a varying seed value to a single hash algorithm.)

By including a bloom filter in each packet that matches all outgoing interfaces that need to forward the packet, each forwarder can efficiently decide whether (and on which interfaces) to forward the packet.

#### 4. False Positives

Bloom filters are probabilistic. A false positive might be indicating a match where the bits are set by aliasing of the hash values. In case of Constrained-Cast, this causes spurious transmission and wastes some energy and radio bandwidth. However, there is no semantic damage (hosts still filter out unneeded multicasts). The total waste in energy and spectrum can be visualized as the false-positive-rate multiplied by the density of the RPL network. A network can easily live with a significant percentage of false positives. By changing the set of hash functions (i.e., seed) over time, the root can avoid a single node with a false positive to become an unnecessary hotspot for that multicast group.

#### 5. Protocol

The protocol uses DAO-like "MLAO" messages to announce membership to the root as specified in Section 5.1.

For downward messages, the root adds a new routing header that includes a hash function identifier and a seed value; another one of its fields gives the number of hash functions ( $k$ ) to ask for  $k$  instances of application of the hash function, with increasing seed. The format of the new routing header is specified in Section 5.2.

Typical sizes of the bloom filter bitmap that the root inserts into the packet can be 64, 128, or 256 bit, which may lead to acceptable false positive rates if the total number of forwarders in the 10s and 100s. (To do: write more about the math here. Note that this number tallies forwarding routers, not end hosts.)

A potential forwarder that receives a multicast packet adorned with a constrained-cast routing header first checks that the packet is marked with a RPL rank smaller than its own (loop prevention). If yes, it then forwards the packet to all outgoing interfaces that match the bloom filter in the packet.

##### 5.1. Multicast Listener Advertisement Object (MLAO)

The header format of the MLAO is depicted in Figure 1. The basic structure of the MLAO message is similar to the RPL Destination Advertisement Object (DAO). In particular, it starts with RPL ICMP base header with a type value of 155 and the code {IANA TBD1} (MLAO), followed by the Checksum, RPLInstanceID, parameters and flags as in a DAO.



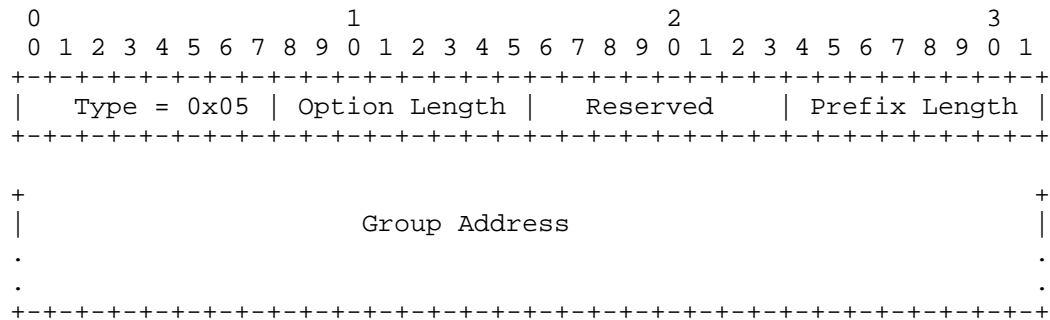


Figure 1: RPL Target Option for MLAO

The group address field indicates the group that the sender of the MLAO is interested in. This field usually contains a 128 bit IPv6 multicast group address. Shorter group identifiers could be used together with a protocol for explicit creation of groups. The MLAO message must have at least one RPL target option to specify the address of the listener that has generated the MLAO. The message is directed to the global unicast address of the DODAG root and travels upwards the routing tree.

Note: It has been suggested to use the RPL Transit Option (Type 0x06) instead as it is used in Non-Storing mode to inform the DODAG root of path attributes. Specifically, this option can be used to limit the subscription by providing a proper Path Lifetime.

## 5.2. Routing Header

This specification uses a new Source Routing 6LowPAN Routing Header (SRH-6LoRH) type [RFC8138] to convey the Bloom filter that describes the source route for the IPv6 multicast packet to take within the RPL routing tree. The 6LoRH Type for this Constrained Cast Routing Header (CCRH) is set to TBD7. Figure 2 depicts the format of this new routing header.

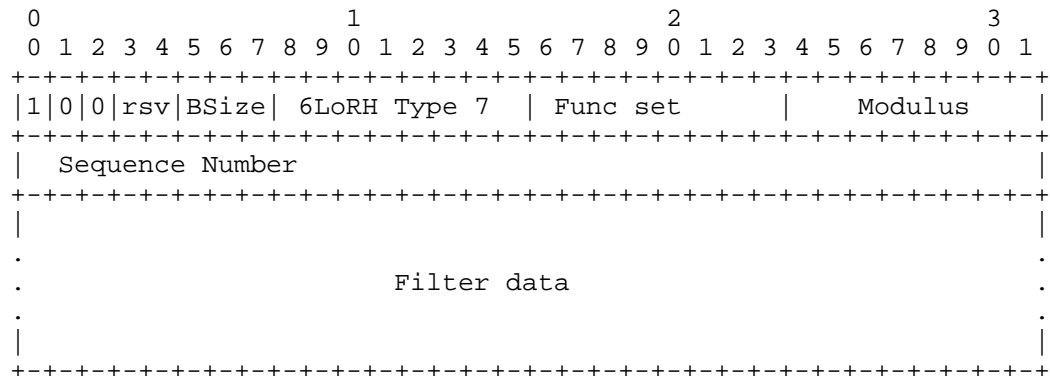


Figure 2: Routing header

rsv: This field is reserved for future use and MUST be set to 0 when sending a packet containing a CCRH. A receiver MUST ignore the value of the rsv field.

BSize: Specifies the size of the included Bloom filter. Currently, only 64 bits, 128 bits, or 256 bits are supported. The BSize field is encoded as specified in Table 1.

BSize value	Filter Size (Bits)
0	64
1	128
2	256

Table 1: Possible Bloom Filter Lengths

Func set: The set of hash functions used to generate the Filter data value.

Note: As the function set contains a combination of several distinct hash functions, it is currently unclear if 8 bits number space is large enough.

Modulus: The modulus that is used by the hash functions, minus 64 (the minimum filter data size that can be used). The DAG root chooses the modulus (and thus the filter data size) to achieve its objectives for false positive rates (Section 4).

Sequence Number: 32 bits sequence number. The number space is unique for a sequence of multicast datagrams for a specific group that arrive at the DAG root on their way up. The DAG root increments the number for each datagram it sends down the respective DODAG.

Filter data: A bit field that indicates which nodes should relay this multicast datagram. The length of this field is a multiple of 8 bytes ( $2^{(BSize + 3)}$  bytes). The actual length is derived from the contents of the field BSize.

## 6. Implementation

In 2013, Constrained-Cast was implemented in Contiki. It turns out that forwarders can compute the hash functions once for their outgoing interfaces and then cache them, simply bit-matching their outgoing interface hash bits against the bloom filter in the packet (a match is indicated when all bits in the outgoing interface hash are set in the bloom filter).

The Root computes the tree for each multicast group, computes the bloom filter for it, caches these values, and then simply adds the bloom filter routing header to each downward packet. For adding a new member, the relevant outgoing interfaces are simply added to the bloom filter. For removing a leaving member, however, the bloom filter needs to be recomputed (which can be sped up logarithmically if desired).

## 7. Benefits

Constrained-Cast:

- o operates in Non-Storing Mode, with the simple addition of a membership information service;
- o performs all routing decisions at the root.

Further optimizations might include using a similar kind of bloom filter routing header for unicast forwarding as well (representing, instead of the outgoing interface list, a list of children that forwarding parents need to forward to).

## 8. Security Considerations

TODO

## 9. IANA Considerations

The following registrations are done following the procedure specified in [RFC6838].

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and "TBD1" with the code selected for TBD1 below and "TBD7" with the code selected for TBD7.

### 9.1. ICMPv6 Parameter Registration

IANA is requested to add the following entry to the Code fields of the RPL Control Codes registry:

Code	Name	Reference
TBD1	MLAO	[RFC-XXXX]

### 9.2. Critical 6LowPAN Routing Header Type Registration

IANA is requested to add the following entries to the Critical 6LowPAN Routing Header Type Registration registry:

Value	Name	Reference
TBD7	CCast Routing Header	[RFC-XXXX]

## 10. Acknowledgments

Thanks to Yasuyuki Tanaka for valuable comments.

This work has been supported by Siemens Corporate Technology.

## 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, <<https://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, <<https://www.rfc-editor.org/info/rfc6550>>.
- [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>>.

## 11.2. Informative References

- [BLOOM] Bloom, B., "Space/time trade-offs in hash coding with allowable errors", ISSN 0001-0782, ACM Press Communications of the ACM vol 13 no 7 pp 422-426, 1970, <<http://doi.acm.org/10.1145/362686.362692>>.
- [I-D.ietf-bier-architecture] Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-08 (work in progress), September 2017.
- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.

## Authors' Addresses

Olaf Bergmann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63904  
Email: bergmann@tzi.org

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

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

Stefanie Gerdes  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63906  
Email: gerdes@tzi.org

Hao Chen  
Huawei Technologies  
12, E. Mozhou Rd  
Nanjing 211111  
China

Phone: +86-25-5662-7052  
Email: philips.chenhao@huawei.com

ROLL  
Internet-Draft  
Intended status: Standards Track  
Expires: 24 September 2022

P. Thubert, Ed.  
Cisco Systems  
R.A. Jadhav  
Huawei Tech  
M. Richardson  
Sandelman  
23 March 2022

Root initiated routing state in RPL  
draft-ietf-roll-dao-projection-25

Abstract

THIS RFC extends RFC 6550, RFC 6553, and RFC 8138 to enable a RPL Root to install and maintain Projected Routes within its DODAG, along a selected set of nodes that may or may not include self, for a chosen duration. This potentially enables routes that are more optimized or resilient than those obtained with the classical distributed operation of RPL, either in terms of the size of a Routing Header or in terms of path length, which impacts both the latency and the packet delivery ratio.

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	4
2.1. Requirements Language . . . . .	4
2.2. References . . . . .	5
2.3. Glossary . . . . .	5
2.4. Domain Terms . . . . .	5
2.4.1. Projected Route . . . . .	6
2.4.2. Projected DAO . . . . .	6
2.4.3. Path . . . . .	6
2.4.4. Routing Stretch . . . . .	6
2.4.5. Track . . . . .	7
3. Context and Goal . . . . .	9
3.1. RPL Applicability . . . . .	10
3.2. RPL Routing Modes . . . . .	11
3.3. Requirements . . . . .	12
3.3.1. Loose Source Routing . . . . .	12
3.3.2. East-West Routes . . . . .	13
3.4. On Tracks . . . . .	15
3.4.1. Building Tracks With RPL . . . . .	15
3.4.2. Tracks and RPL Instances . . . . .	16
3.5. Serial Track Signaling . . . . .	16
3.5.1. Using Storing Mode Segments . . . . .	18
3.5.2. Using Non-Storing Mode joining Tracks . . . . .	24
3.6. Complex Tracks . . . . .	31
3.7. Scope and Expectations . . . . .	33
3.7.1. External Dependencies . . . . .	33
3.7.2. Positioning vs. Related IETF Standards . . . . .	33
4. Extending existing RFCs . . . . .	35
4.1. Extending RFC 6550 . . . . .	35
4.1.1. Projected DAO . . . . .	36
4.1.2. Projected DAO-ACK . . . . .	38
4.1.3. Via Information Option . . . . .	39
4.1.4. Sibling Information Option . . . . .	39
4.1.5. P-DAO Request . . . . .	39
4.1.6. Amending the RPI . . . . .	40
4.1.7. Additional Flag in the RPL DODAG Configuration Option . . . . .	40
4.2. Extending RFC 6553 . . . . .	41
4.3. Extending RFC 8138 . . . . .	42
5. New RPL Control Messages and Options . . . . .	43



5.1.	New P-DAO Request Control Message . . . . .	43
5.2.	New PDR-ACK Control Message . . . . .	45
5.3.	Via Information Options . . . . .	46
5.4.	Sibling Information Option . . . . .	49
6.	Root Initiated Routing State . . . . .	51
6.1.	RPL Network Setup . . . . .	51
6.2.	Requesting a Track . . . . .	52
6.3.	Identifying a Track . . . . .	53
6.4.	Installing a Track . . . . .	54
6.4.1.	Signaling a Projected Route . . . . .	55
6.4.2.	Installing a Track Segment with a Storing Mode P-Route . . . . .	56
6.4.3.	Installing a Track Leg with a Non-Storing Mode P-Route . . . . .	58
6.5.	Tearing Down a P-Route . . . . .	60
6.6.	Maintaining a Track . . . . .	60
6.6.1.	Maintaining a Track Segment . . . . .	61
6.6.2.	Maintaining a Track Leg . . . . .	61
6.7.	Encapsulating and Forwarding Along a Track . . . . .	62
6.8.	Compression of the RPL Artifacts . . . . .	64
7.	Lesser Constrained Variations . . . . .	66
7.1.	Storing Mode Main DODAG . . . . .	66
7.2.	A Track as a Full DODAG . . . . .	68
8.	Profiles . . . . .	69
9.	Backwards Compatibility . . . . .	71
10.	Security Considerations . . . . .	72
11.	IANA Considerations . . . . .	72
11.1.	RPL DODAG Configuration Option Flag . . . . .	72
11.2.	Elective 6LoWPAN Routing Header Type . . . . .	73
11.3.	Critical 6LoWPAN Routing Header Type . . . . .	73
11.4.	Subregistry For The RPL Option Flags . . . . .	73
11.5.	RPL Control Codes . . . . .	74
11.6.	RPL Control Message Options . . . . .	74
11.7.	SubRegistry for the Projected DAO Request Flags . . . . .	75
11.8.	SubRegistry for the PDR-ACK Flags . . . . .	75
11.9.	Subregistry for the PDR-ACK Acceptance Status Values . . . . .	76
11.10.	Subregistry for the PDR-ACK Rejection Status Values . . . . .	76
11.11.	SubRegistry for the Via Information Options Flags . . . . .	77
11.12.	SubRegistry for the Sibling Information Option Flags . . . . .	77
11.13.	Destination Advertisement Object Flag . . . . .	77
11.14.	Destination Advertisement Object Acknowledgment Flag . . . . .	78
11.15.	New ICMPv6 Error Code . . . . .	78
11.16.	RPL Rejection Status values . . . . .	78
12.	Acknowledgments . . . . .	79
13.	Normative References . . . . .	79
14.	Informative References . . . . .	81
	Authors' Addresses . . . . .	83

## 1. Introduction

RPL, the "Routing Protocol for Low Power and Lossy Networks" [RPL] (LLNs), is an anisotropic Distance Vector protocol that is well-suited for application in a variety of low energy Internet of Things (IoT) networks where stretched P2P paths are acceptable vs. the signaling and state overhead involved in maintaining shortest paths across.

RPL forms destination Oriented Directed Acyclic Graphs (DODAGs) in which the Root often acts as the Border router to connect the RPL domain to the IP backbone and routes along that graph up, towards the Root, and down towards the nodes.

With this specification, an abstract routing function called a Path Computation Element [PCE] (e.g., located in an central controller or collocated with the Root) interacts with the RPL Root to compute Peer to Peer (P2P) paths within a pre-existing RPL Main DODAG. The topological information that is passed to the PCE is derived from the DODAG that is already available at the Root in RPL Non-Storing Mode. This specification introduces protocol extensions that enrich the topological information that is available at the Root and passed to the PCE.

Based on usage, path length, and knowledge of available resources such as battery levels and reservable buffers in the nodes, the PCE with a global visibility on the system can optimize the computed routes for the application needs, including the capability to provide path redundancy. This specification also introduces protocol extensions that enable the Root to translates the computed paths into RPL and install them as Projected Routes (aka P-Routes) inside the DODAG on behalf of a PCE.

## 2. Terminology

### 2.1. Requirements Language

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

In addition, the terms "Extends" and "Amends" are used as per [I-D.kuehlewind-update-tag] section 3.

## 2.2. References

In THIS RFC, readers will encounter terms and concepts that are discussed in the "Routing Protocol for Low Power and Lossy Networks" [RPL], the "6TiSCH Architecture" [RFC9030], the "Deterministic Networking Architecture" [RFC8655], the "Reliable and Available Wireless (RAW) Architecture" [RAW-ARCHI], and "Terminology in Low power And Lossy Networks" [RFC7102].

## 2.3. Glossary

THIS RFC often uses the following acronyms:

CMO: Control Message Option  
DAO: destination Advertisement Object  
DAG: Directed Acyclic Graph  
DODAG: destination-Oriented Directed Acyclic Graph; A DAG with only one vertex (i.e., node) that has no outgoing edge (i.e., link)  
GUA: IPv6 Global Unicast Address  
LLN: Low-Power and Lossy Network  
MOP: RPL Mode of Operation  
P-DAO: Projected DAO  
P-Route: Projected Route  
PDR: P-DAO Request  
RAN: RPL-Aware Node (either a RPL router or a RPL-Aware Leaf)  
RAL: RPL-Aware Leaf  
RH: Routing Header  
RPI: RPL Packet Information  
RTO: RPL Target Option  
RUL: RPL-Unaware Leaf  
SIO: RPL Sibling Information Option  
ULA: IPv6 Unique Local Address  
NSM-VIO: A Source-Routed Via Information Option, used in Non-Storing Mode P-DAO messages.  
SLO: Service Level Objective  
TIO: RPL Transit Information Option  
SM-VIO: A strict Via Information Option, used in Storing Mode P-DAO messages.  
VIO: A Via Information Option; it can be a SM-VIO or an NSM-VIO.

## 2.4. Domain Terms

This specification uses the following terminology:

#### 2.4.1. Projected Route

A RPL P-Route is a RPL route that is computed remotely by a PCE, and installed and maintained by a RPL Root on behalf of the PCE. It is installed as a state that signals that destinations (aka Targets) are reachable along a sequence of nodes.

#### 2.4.2. Projected DAO

A DAO message used to install a P-Route.

#### 2.4.3. Path

Quoting section 1.1.3 of [INT-ARCHI]:

At a given moment, all the IP datagrams from a particular source host to a particular destination host will typically traverse the same sequence of gateways. We use the term "path" for this sequence. Note that a path is uni-directional; it is not unusual to have different paths in the two directions between a given host pair.

Section 2 of [I-D.irtf-panrg-path-properties] points to a longer, more modern definition of path, which begins as follows:

A sequence of adjacent path elements over which a packet can be transmitted, starting and ending with a node. A path is unidirectional. Paths are time-dependent, i.e., the sequence of path elements over which packets are sent from one node to another may change. A path is defined between two nodes.

It follows that the general acceptance of a path is a linear sequence of nodes, as opposed to a multi-dimensional graph. In the context of this document, a path is observed by following one copy of a packet that is injected in a Track and possibly replicated within.

#### 2.4.4. Routing Stretch

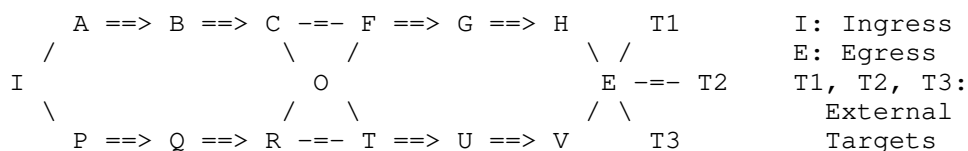
RPL is anisotropic, meaning that it is directional, or more exactly polar. RPL does not behave the same way "down" with multicast DIO messages that form the DODAG and "up" with unicast DAO messages that follow the DODAG. This is in contrast with traditional IGPs that operate the same in all directions and are thus called isotropic.

The term Routing Stretch denotes the length of a path, as compared with a shortest path, which can be an abstract concept in RPL when the metrics are statistical and dynamic, and the concept of short varies with the Objective Function.

The RPL DODAG optimizes the P2MP (from Root) and MP2P (to Root) paths, but the P2P (node to node) traffic has to follow the same DODAG. Following the DODAG, the RPL datapath passes via a common parent in Storing Mode and via the Root in Non-Storing Mode. This typically involves more hops and more latency than the minimum possible for a direct P2P path that an isotropic protocol would compute. We refer to this elongated path as stretched.

#### 2.4.5. Track

A networking graph that can be followed to transport packets with equivalent treatment; as opposed to the definition of a path above, a Track is not necessarily linear. It may contain multiple paths that may fork and rejoin, and may enable the RAW Packet ARQ, Replication, Elimination, and Overhearing (PAREO) operations.



I ==> A ==> B ==> C : a segment to targets F and O

I --> F --> E : a leg to targets T1, T2, T3

I, A, B, C, F, G, H, E : a path to T1, T2, T3

Figure 1: A Track and its Components

This specification builds Tracks that are DODAGs oriented towards a Track Ingress, and the forward direction for packets (aka East-West) is from the Track Ingress to one of the possibly multiple Track Egress Nodes, which is also down the DODAG.

The Track may be strictly connected, meaning that the vertices are adjacent, or loosely connected, meaning that the vertices are connected using Segments that are associated to the same Track.

##### 2.4.5.1. TrackID

A RPL Local InstanceID that identifies a Track using the namespace owned by the Track Ingress. The TrackID is associated with the IPv6 Address of the Track Ingress that is used as DODAGID, and together they form a unique identification of the Track (see the definition of DODAGID in section 2 of [RPL]).

#### 2.4.5.2. Namespace

The term namespace is used to refer to the scope of the TrackID. The TrackID is locally significant within its namespace. The namespace is identified by the DODAGID for the Track. The tuple (DODAGID, TrackID) is globally unique.

#### 2.4.5.3. Serial Track

A Track that has only one path.

#### 2.4.5.4. Stand-Alone

A single P-DAO that fully defines a Track, e.g., a Serial Track installed with a single Storing Mode Via Information option (SM-VIO).

#### 2.4.5.5. Stitching

This specification using the term stitching to indicate that a track is piped to another one, meaning that traffic out of the first is injected in the other.

#### 2.4.5.6. Leg

An end-to-end East-West serial path. A leg can be a serial Track by itself or a subTrack of a complex Track with the same Ingress and Egress Nodes. With this specification, a Leg is installed by the Root of the main DODAG using a Non-Storing Mode P-DAO message, and it is expressed as a loose sequence of nodes that are joined by Track Segments.

As the Non-Storing Mode Via Information option (NSM-VIO) can only signal sequences of nodes, it takes one Non-Storing Mode P-DAO message per Leg to signal the structure of a complex Track.

Each NSM-VIO for the same TrackId but a different Segment ID signals a different leg that the Track Ingress adds to the topology.

#### 2.4.5.7. subTrack

A Track within a Track, formed by a non-empty collection of Legs of the Track.

#### 2.4.5.8. Segment

A serial path formed by a strict sequence of nodes, along which a P-Route is installed. With this specification, a Segment is typically installed by the Root of the main DODAG using Storing Mode P-DAO messages. A Segment is used as the topological edge of a Track joining the loose steps along the Legs that form the structure of a complex Track. The same segment may be leveraged by more than one Leg where the Legs overlap.

Since this specification builds only DODAGs, all Segments are oriented from Ingress (East) to Egress (West), as opposed to the general Track model in the RAW Architecture [RAW-ARCHI], which allows North/South Segments that can be bidirectional as well.

##### 2.4.5.8.1. Section of a Segment

A continuous subset of a segment that may be replaced while the segment remains. for instance, in segment  $A=>B=>C=>D=>E=>F$ , say that the link C to D might be misbehaving. The section  $B=>C=>D=>E$  in the segment may be replaced by  $B=>C'=>D'=>E$  to route around the problem. The segment becomes  $A=>B=>C'=>D'=>E=>F$ .

##### 2.4.5.8.2. Segment Routing and SRH

The terms Segment Routing and SRH refer to using source-routing to hop over segments. In a Non-Storing mode RPL domain, the SRH is typically a RPL Source Route Header (the IPv6 RH of type 3) as defined in [RFC6554].

If the network is a 6LoWPAN Network, the expectation is that the SRH is compressed and encoded as a 6LoWPAN Routing Header (6LoRH), as specified in section 5 of [RFC8138].

On the other hand, if the RPL Network is less constrained and operated in Storing Mode, as discussed in Section 7.1, the Segment Routing operation and the SRH could be as specified in [RFC8754]. This specification applies equally to both forms of source routing and SRH.

### 3. Context and Goal

### 3.1. RPL Applicability

RPL is optimized for situations where the power is scarce, the bandwidth constrained and the transmissions unreliable. This matches the use case of an IoT LLN where RPL is typically used today, but also situations of high relative mobility between the nodes in the network (aka swarming), e.g., within a variable set of vehicles with a similar global motion, or a toon of drones.

To reach this goal, RPL is primarily designed to minimize the control plane activity, that is the relative amount of routing protocol exchanges vs. data traffic, and the amount of state that is maintained in each node. RPL does not need converge, and provides connectivity to most nodes most of the time.

RPL may form multiple topologies called instances. Instances can be created to enforce various optimizations through objective functions, or to reach out through different Root Nodes. The concept of objective function allows to adapt the activity of the routing protocol to the use case, e.g., type, speed, and quality of the LLN links.

RPL instances operate as ships passing in the night, unbeknownst of one another. The RPL Root is responsible to select the RPL Instance that is used to forward a packet coming from the Backbone into the RPL domain and set the related RPL information in the packets. 6TiSCH leverages RPL for its distributed routing operations.

To reduce the routing exchanges, RPL leverages an anisotropic Distance Vector approach, which does not need a global knowledge of the topology, and only optimizes the routes to and from the RPL Root, allowing P2P paths to be stretched. Although RPL installs its routes proactively, it only maintains them lazily, in reaction to actual traffic, or as a slow background activity.

This is simple and efficient in situations where the traffic is mostly directed from or to a central node, such as the control traffic between routers and a controller of a Software Defined Networking (SDN) infrastructure or an Autonomic Control Plane (ACP).

But stretch in P2P routing is counter-productive to both reliability and latency as it introduces additional delay and chances of loss. As a result, [RPL] is not a good fit for the use cases listed in the RAW use cases document [USE-CASES], which demand high availability and reliability, and as a consequence require both short and diverse paths.



### 3.2. RPL Routing Modes

RPL first forms a default route in each node towards the a Root, and those routes together coalesce as a Directed Acyclic Graph upwards. RPL then constructs routes to destinations signaled as Targets in the reverse direction, down the same DODAG. So do so, a RPL Instance can be operated either in RPL Storing or Non-Storing Mode of Operation (MOP). The default route towards the Root is maintained aggressively and may change while a packet progresses without causing loops, so the packet will still reach the Root.

In Non-Storing Mode, each node advertises itself as a Target directly to the Root, indicating the parents that may be used to reach self. Recursively, the Root builds and maintains an image of the whole DODAG in memory, and leverages that abstraction to compute source route paths for the packets to their destinations down the DODAG. When a node changes its point(s) of attachment to the DODAG, it takes single unicast packet to the Root along the default route to update it, and the connectivity is restored immediately; this mode is preferable for use cases where internet connectivity is dominant, or when, like here, the Root controls the network activity in the nodes.

In Storing Mode, the routing information percolates upwards, and each node maintains the routes to the subDAG of its descendants down the DODAG. The maintenance is lazy, either reactive upon traffic or as a slow background process. Packets flow via the common parent and the routing stretch is reduced vs. Non-Storing, for a better P2P connectivity. On the other hand, a new route takes a longer time to propagate to the Root, time for the Distance-Vector protocol to operate hop-by-hop, and the Internet connectivity is restored more slowly upon movement.

Either way, the RPL routes are injected by the Target nodes, in a distributed fashion. To complement RPL and eliminate routing stretch, this specification introduces an hybrid mode that combines Storing and Non-Storing operations to build and project routes onto the nodes where they should be installed. This specification uses the term Projected Route (P-Route) to refer to those routes.

A P-Route may be installed in either Storing and Non-Storing Mode, potentially resulting in hybrid situations where the Mode of the P-Route is different from that of the RPL Main DODAG. P-Routes can be used as stand-alone segments to reduce the size of the source routing headers with loose source routing operations down the main RPL DODAG. P-Routes can also be combined with other P-Routes to form a more complex forwarding graph called a Track.

### 3.3. Requirements

#### 3.3.1. Loose Source Routing

A RPL implementation operating in a very constrained LLN typically uses the Non-Storing Mode of Operation as represented in Figure 2. In that mode, 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 typically builds a source routed path to a destination down the DODAG by recursively concatenating this information.

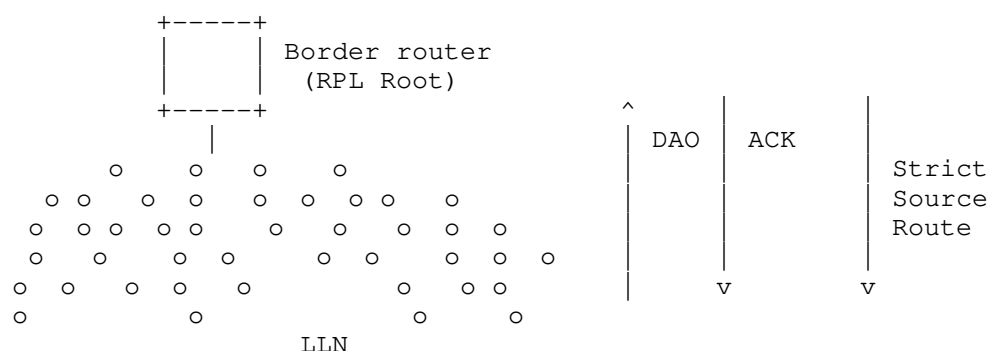


Figure 2: RPL Non-Storing Mode of operation

Based on the parent-children relationships expressed in the Non-Storing DAO messages, the Root possesses topological information about the whole network, though this information is limited to the structure of the DODAG for which it is the destination. A packet that is generated within the domain will always reach the Root, which can then apply a source routing information to reach the destination if the destination is also in the DODAG. 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. It results that the wireless bandwidth near the Root is the gating factor for all transmissions towards or within the domain, and that the Root is a single point of failure for all connectivity to nodes within its domain.

The RPL Root must add a source routing header to all downward packets. As a network grows, the size of the source routing header augments with the depth of the nodes. In some use cases, a RPL network forms long lines along physical structures such as streets for lighting. Limiting the packet size is directly beneficial to the energy budget, but, mostly, it reduces the chances of frame loss and packet fragmentation, which are highly detrimental to the LLN operation. A limited amount of well-targeted routing state would

allow the source routing operation to be loose as opposed to strict, and save packet size. 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 or an associated PCE may possess by means that are outside of the scope of this specification.

Being on path for all packets in Non-Storing mode, the Root may determine the number of P2P packets in its RPL domain per source and destination, the latency incurred, and the amount of energy and bandwidth that is consumed to reach the self and then down, including a possible fragmentation when encapsulating larger packets. Enabling a shorter path that would not traverse the Root for select P2P source/destinations may improve the latency, lower the consumption of constrained resources, free bandwidth at the bottleneck near the Root, improve the delivery ratio and reduce the latency for those P2P flows with a global benefit for all flows of reducing the load at the Root.

This requirement is to store a routing state associated with the Main DODAG in selected RPL routers, to limit the excursion of the source route headers in deep networks. The Root may elide the sequence of routers that is installed in the network from its source route header, which becomes loose while it is strict in [RPL].

### 3.3.2. East-West Routes

[RPL] optimizes Point-to-Multipoint (P2MP) routes from the Root, Multipoint-to-Point (MP2P) routes to the DODAG Root, and Internet access when the Root also serves as Border Router. All routes are installed North-South (aka up/down) along the RPL DODAG. Peer to Peer (P2P) East-West routes in a RPL network will generally suffer from some elongated (stretched) path versus a direct (optimized) path, since routing between two nodes always happens via a common parent, as illustrated in Figure 3:

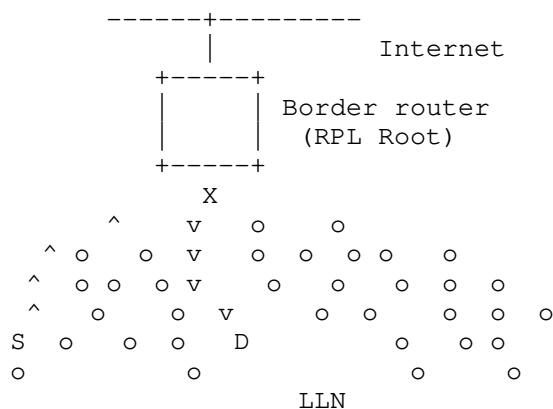


Figure 3: Routing Stretch between S and D via common parent X  
along North-South Paths

As described in [RFC9008], the amount of stretch depends on the Mode of Operation:

- \* 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 an RH 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 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 East-West P2P routes, either if the RPL route presents a stretch from shortest path, or if the new route is engineered with a different objective, and that it is even more critical in Non-Storing Mode than it is in Storing Mode, because the routing stretch is wider. 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.

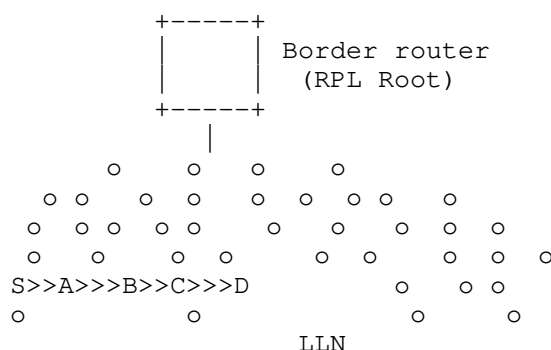


Figure 4: More direct East-West Route between S and D

The requirement is to install additional routes in the RPL routers, to reduce the stretch of some P2P routes and maintain the characteristics within a given SLO, e.g., in terms of latency and/or reliability.

### 3.4. On Tracks

#### 3.4.1. Building Tracks With RPL

The concept of a Track was introduced in the "6TiSCH Architecture" [RFC9030], as a collection of potential paths that leverage redundant forwarding solutions along the way. This can be a DODAG or a more complex structure that is only partially acyclic (e.g., per packet).

With this specification, a Track is shaped as a DODAG, and following the directed edges leads to a Track Ingress. Storing Mode P-DAO messages follow the direction of the edges to set up routes for traffic that flows the other way, towards the Track Egress(es). If there is a single Track Egress, then the Track is reversible to form another DODAG by reversing the direction of each edge. A node at the Ingress of more than one Segment in a Track may use one or more of these Segments to forward a packet inside the Track.

A RPL Track is a collection of (one or more) parallel loose source routed sequences of nodes ordered from Ingress to Egress, each forming a Track Leg. The nodes that are directly connected, reachable via existing Tracks as illustrated in Section 3.5.2.3 or joined with strict Segments of other nodes as shown in Section 3.5.1.3. The Legs are expressed in RPL Non-Storing Mode and require an encapsulation to add a Source Route Header, whereas the Segments are expressed in RPL Storing Mode.

A Serial Track comprises provides only one path between Ingress and Egress. It comprises at most one Leg. A Stand-Alone Segment implicitly defines a Serial Track from its Ingress to Egress.

A complex Track forms a graph that provides a collection of potential paths to provide redundancy for the packets, either as a collection of Legs that may be parallel or cross at certain points, or as a more generic DODAG.

### 3.4.2. Tracks and RPL Instances

Section 5.1. of [RPL] describes the RPL Instance and its encoding. There can be up to 128 global RPL Instances, for which there can be one or more DODAGs, and there can be 64 local RPL Instances, with a namespace that is indexed by a DODAGID, where the DODAGID is a Unique Local Address (ULA) or a Global Unicast Address (GUA) of the Root of the DODAG. Bit 0 (most significant) is set to 1 to signal a Local RPLInstanceID, as shown in Figure 5. By extension, this specification expresses the value of the RPLInstanceID as a single integer between 128 and 191, representing both the Local RPLInstanceID in 0..63 and Bit 0 set.

```

0 1 2 3 4 5 6 7
+---+---+---+---+---+---+
|1|D|   ID   |  Local RPLInstanceID in 0..63
+---+---+---+---+---+---+

```

Figure 5: Local RPLInstanceID Encoding

A Track is normally associated with a Local RPL Instance which RPLInstanceID is used as the TrackID, more in Section 6.3. A Track Leg may also be used as a subTrack that extends the RPL main DODAG. In that case, the TrackID is set to the global RPLInstanceID of the main DODAG, which suffices to identify the routing topology. As opposed to local RPL instances, the Track Ingress that encapsulates the packets over a subtrack is not Root, and that the source address of the encapsulated packet is not used to determine the Track.

### 3.5. Serial Track Signaling

This specification enables to set up a P-Route along either a Track Leg or a Segment. A P-Route is installed and maintained by the Root of the main DODAG using an extended RPL DAO message called a Projected DAO (P-DAO), and a Track is composed of the combination of one or more P-Routes.

A P-DAO message for a Track signals the TrackID in the RPLInstanceID field. In the case of a local RPL Instance, the address of the Track Ingress is used as source to encapsulate packets along the Track. The Track is signaled in the DODAGID field of the Projected DAO Base Object, see Figure 8.

This specification introduces the Via Information Option (VIO) to signal a sequence of hops in a Leg or a Segment in the P-DAO messages, either in Storing Mode (SM-VIO) or Non-Storing Mode (NSM-VIO). One P-DAO messages contains a single VIO, associated to one or more RPL Target Options that signal the destination IPv6 addresses that can be reached along the Track, more in Section 5.3.

Before diving deeper into Track Legs and Segments signaling and operation, this section provides examples of what how route projection works through variations of a simple example. This simple example illustrates the case of host routes, though RPL Targets can be prefixes.

Say we want to build a Serial Track from node A to E in Figure 6, so A can route packets to E's neighbors F and G along A, B, C, D and E as opposed to via the Root:

```

A ==> B ==> C ==> D ==> E < /==> F
                             \==> G

```

Figure 6: Reference Track

Conventionally we use ==> to represent a strict hop and --> for a loose hop. We use "-to-", such as in C==>D==>E-to-F to represent coma-separated Targets, e.g., F is a Target for Segment C==>D==>E. In this example, A is Track Ingress, E is Track Egress. C is a stitching point. F and G are "external" Targets for the Track, and become reachable from A via the Track A(ingress) to E (Egress and implicit Target in Non-Storing Mode) leading to F and G (explicit Targets).

Figure 5 depicts the format of the RPLInstanceID encoding for a local RPLInstanceID .

In a general manner the desired outcome is as follows:

- \* Targets are E, F, and G
- \* P-DAO 1 signals C==>D==>E

- \* P-DAO 2 signals  $A \Rightarrow B \Rightarrow C$

- \* P-DAO 3 signals F and G via the  $A \rightarrow E$  Track

P-DAO 3 may be omitted if P-DAO 1 and 2 signal F and G as Targets.

Loose sequences of hops must be expressed in Non-Storing Mode, so P-DAO 3 contains a NSM-VIO. With this specification, the DODAGID to be used by the Ingress as source address is signaled if needed in the DAO base object, the via list starts at the first loose hop and matches the source route header, and the Egress of a Non-Storing Mode P-DAO is an implicit Target that is not listed in the RTO.

### 3.5.1. Using Storing Mode Segments

$A \Rightarrow B \Rightarrow C$  and  $C \Rightarrow D \Rightarrow E$  are segments of a same Track. Note that the Storing Mode signaling imposes strict continuity in a segment, since the P-DAO is passed hop by hop, as a classical DAO is, along the reverse datapath that it signals. One benefit of strict routing is that loops are avoided along the Track.

#### 3.5.1.1. Stitched Segments

In this formulation:

- \* P-DAO 1 signals  $C \Rightarrow D \Rightarrow E$ -to-F,G

- \* P-DAO 2 signals  $A \Rightarrow B \Rightarrow C$ -to-F,G

Storing Mode P-DAO 1 is sent to E and when it is successfully acknowledged, Storing Mode P-DAO 2 is sent to C, as follows:



Field	P-DAO 1 to E	P-DAO 2 to C
Mode	Storing	Storing
Track Ingress	A	A
(DODAGID, TrackID)	(A, 129)	(A, 129)
SegmentID	1	2
VIO	C, D, E	A, B, C
Targets	F, G	F, G

Table 1: P-DAO Messages

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	P-DAO 1	Neighbor	(A, 129)
D	E	P-DAO 1	Neighbor	(A, 129)
"	F, G	P-DAO 1	E	(A, 129)
C	D	P-DAO 1	Neighbor	(A, 129)
"	F, G	P-DAO 1	D	(A, 129)
B	C	P-DAO 2	Neighbor	(A, 129)
"	F, G	P-DAO 2	C	(A, 129)
A	B	P-DAO 2	Neighbor	(A, 129)
"	F, G	P-DAO 2	B	(A, 129)

Table 2: RIB setting

Packets originated by A to F or G do not require an encapsulation as the RPI can be placed in the native header chain. For packets that it routes, A must encapsulate to add the RPI that signals the trackID; the outer headers of the packets that are forwarded along the Track have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	A	F or G	(A, 129)
Inner	X != A	F or G	N/A

Table 3: Packet Header Settings

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 2: A forwards to B and B forwards to C.
- \* From P-DAO 1: C forwards to D and D forwards to E.
- \* From Neighbor Cache Entry: E delivers the packet to F.

#### 3.5.1.2. External routes

In this example, we consider F and G as destinations that are external to the Track as a DODAG, as discussed in section 4.1.1. of [RFC9008]. We then apply the directives for encapsulating in that case, more in Section 6.7.

In this formulation, we set up the Track Leg explicitly, which creates less routing state in intermediate hops at the expense of larger packets to accommodate source routing:

- \* P-DAO 1 signals C==>D==>E-to-E
- \* P-DAO 2 signals A==>B==>C-to-E
- \* P-DAO 3 signals F and G via the A-->E-to-F,G Track

Storing Mode P-DAO 1 and 2, and Non-Storing Mode P-DAO 3, are sent to E, C and A, respectively, as follows:

	P-DAO 1 to E	P-DAO 2 to C	P-DAO 3 to A
Mode	Storing	Storing	Non-Storing
Track Ingress	A	A	A
(DODAGID, TrackID)	(A, 129)	(A, 129)	(A, 129)
SegmentID	1	2	3
VIO	C, D, E	A, B, C	E
Targets	E	E	F, G

Table 4: P-DAO Messages

Note in the above that E is not an implicit Target in Storing mode, so it must be added in the RTO.

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	P-DAO 1	Neighbor	(A, 129)
D	E	P-DAO 1	Neighbor	(A, 129)
C	D	P-DAO 1	Neighbor	(A, 129)
"	E	P-DAO 1	D	(A, 129)
B	C	P-DAO 2	Neighbor	(A, 129)
"	E	P-DAO 2	C	(A, 129)
A	B	P-DAO 2	Neighbor	(A, 129)
"	E	P-DAO 2	B	(A, 129)
"	F, G	P-DAO 3	E	(A, 129)

Table 5: RIB setting

Packets from A to E do not require an encapsulation. The outer headers of the packets that are forwarded along the Track have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	A	E	(A, 129)
Inner	X	E (X != A), F or G	N/A

Table 6: Packet Header Settings

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 3: A encapsulates the packet the Track signaled by P-DAO 3, with the outer header above. Now the packet destination is E.
- \* From P-DAO 2: A forwards to B and B forwards to C.
- \* From P-DAO 1: C forwards to D and D forwards to E; E decapsulates the packet.
- \* From Neighbor Cache Entry: E delivers packets to F or G.

#### 3.5.1.3. Segment Routing

In this formulation leverages Track Legs to combine Segments and form a Graph. The packets are source routed from a Segment to the next to adapt the path. As such, this can be seen as a form of Segment Routing [RFC8402]:

- \* P-DAO 1 signals C==>D==>E-to-E
- \* P-DAO 2 signals A==>B-to-B,C
- \* P-DAO 3 signals F and G via the A-->C-->E-to-F,G Track

Storing Mode P-DAO 1 and 2, and Non-Storing Mode P-DAO 3, are sent to E, B and A, respectively, as follows:

	P-DAO 1 to E	P-DAO 2 to B	P-DAO 3 to A
Mode	Storing	Storing	Non-Storing
Track Ingress	A	A	A
(DODAGID, TrackID)	(A, 129)	(A, 129)	(A, 129)
SegmentID	1	2	3
VIO	C, D, E	A, B	C, E
Targets	E	C	F, G

Table 7: P-DAO Messages

Note in the above that the Segment can terminate at the loose hop as used in the example of P-DAO 1 or at the previous hop as done with P-DAO 2. Both methods are possible on any Segment joined by a loose Track Leg. P-DAO 1 generates more signaling since E is the Segment Egress when D could be, but has the benefit that it validates that the connectivity between D and E still exists.

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	P-DAO 1	Neighbor	(A, 129)
D	E	P-DAO 1	Neighbor	(A, 129)
C	D	P-DAO 1	Neighbor	(A, 129)
"	E	P-DAO 1	D	(A, 129)
B	C	P-DAO 2	Neighbor	(A, 129)
A	B	P-DAO 2	Neighbor	(A, 129)
"	C	P-DAO 2	B	(A, 129)
"	E, F, G	P-DAO 3	C, E	(A, 129)

Table 8: RIB setting

Packets originated at A to E do not require an encapsulation, but carry a SRH via C. The outer headers of the packets that are forwarded along the Track have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	A	C till C then E	(A, 129)
Inner	X	E (X != A), F or G	N/A

Table 9: Packet Header Settings

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 3: A encapsulates the packet the Track signaled by P-DAO 3, with the outer header above. Now the destination in the IPv6 Header is C, and a SRH signals the final destination is E.
- \* From P-DAO 2: A forwards to B and B forwards to C.
- \* From P-DAO 3: C processes the SRH and sets the destination in the IPv6 Header to E.
- \* From P-DAO 1: C forwards to D and D forwards to E; E decapsulates the packet.
- \* From the Neighbor Cache Entry: E delivers packets to F or G.

### 3.5.2. Using Non-Storing Mode joining Tracks

In this formulation:

- \* P-DAO 1 signals C==>D==>E-to-F,G
- \* P-DAO 2 signals A==>B==>C-to-E,F,G

A==>B==>C and C==>D==>E are Tracks expressed as Non-Storing P-DAOs.

#### 3.5.2.1. Stitched Tracks

Non-Storing Mode P-DAO 1 and 2 are sent to C and A respectively, as follows:

	P-DAO 1 to C	P-DAO 2 to A
Mode	Non-Storing	Non-Storing
Track Ingress	C	A
(DODAGID, TrackID)	(C, 131)	(A, 131)
SegmentID	1	1
VIO	D, E	B, C
Targets	F, G	E, F, G

Table 10: P-DAO Messages

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	ND	Neighbor	Any
D	E	ND	Neighbor	Any
C	D	ND	Neighbor	Any
"	E, F, G	P-DAO 1	D, E	(C, 131)
B	C	ND	Neighbor	Any
A	B	ND	Neighbor	Any
"	C, E, F, G	P-DAO 2	B, C	(A, 131)

Table 11: RIB setting

Packets originated at A to E, F and G do not require an encapsulation, though it is preferred that A encapsulates and C decapsulates. Either way, they carry a SRH via B and C, and C needs to encapsulate to E, F, or G to add an SRH via D and E. The encapsulating headers of packets that are forwarded along the Track between C and E have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	C	D till D then E	(C, 131)
Inner	X	E, F, or G	N/A

Table 12: Packet Header Settings between C and E

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 2: A encapsulates the packet with destination of F in the Track signaled by P-DAO 2. The outer header has source A, destination B, an SRH that indicates C as the next loose hop, and a RPI indicating a TrackId of 131 from A's namespace, which is distinct from TrackId of 131 from C's.
- \* From the SRH: Packets forwarded by B have source A, destination C, a consumed SRH, and a RPI indicating a TrackId of 131 from A's namespace. C decapsulates.
- \* From P-DAO 1: C encapsulates the packet with destination of F in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E decapsulates.

#### 3.5.2.2. External routes

In this formulation:

- \* P-DAO 1 signals C==>D==>E-to-E
- \* P-DAO 2 signals A==>B==>C-to-C,E
- \* P-DAO 3 signals F and G via the A-->E-to-F,G Track

Non-Storing Mode P-DAO 1 is sent to C and Non-Storing Mode P-DAO 2 and 3 are sent A, as follows:



	P-DAO 1 to C	P-DAO 2 to A	P-DAO 3 to A
Mode	Non-Storing	Non-Storing	Non-Storing
Track Ingress	C	A	A
(DODAGID, TrackID)	(C, 131)	(A, 129)	(A, 141)
SegmentID	1	1	1
VIO	D, E	B, C	E
Targets	E	E	F, G

Table 13: P-DAO Messages

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	ND	Neighbor	Any
D	E	ND	Neighbor	Any
C	D	ND	Neighbor	Any
"	E	P-DAO 1	D, E	(C, 131)
B	C	ND	Neighbor	Any
A	B	ND	Neighbor	Any
"	C, E	P-DAO 2	B, C	(A, 129)
"	F, G	P-DAO 3	E	(A, 141)

Table 14: RIB setting

The encapsulating headers of packets that are forwarded along the Track between C and E have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	C	D till D then E	(C, 131)
Middle	A	E	(A, 141)
Inner	X	E, F or G	N/A

Table 15: Packet Header Settings

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 3: A encapsulates the packet with destination of F in the Track signaled by P-DAO 3. The outer header has source A, destination E, and a RPI indicating a TrackId of 141 from A's namespace. This recurses with:
- \* From P-DAO 2: A encapsulates the packet with destination of E in the Track signaled by P-DAO 2. The outer header has source A, destination B, an SRH that indicates C as the next loose hop, and a RPI indicating a TrackId of 129 from A's namespace.
- \* From the SRH: Packets forwarded by B have source A, destination C, a consumed SRH, and a RPI indicating a TrackId of 129 from A's namespace. C decapsulates.
- \* From P-DAO 1: C encapsulates the packet with destination of E in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E decapsulates.

### 3.5.2.3. Segment Routing

In this formulation:

- \* P-DAO 1 signals C==>D==>E-to-E
- \* P-DAO 2 signals A==>B-to-C
- \* P-DAO 3 signals F and G via the A-->C-->E-to-F,G Track

Non-Storing Mode P-DAO 1 is sent to C and Non-Storing Mode P-DAO 2 and 3 are sent A, as follows:

	P-DAO 1 to C	P-DAO 2 to A	P-DAO 3 to A
Mode	Non-Storing	Non-Storing	Non-Storing
Track Ingress	C	A	A
(DODAGID, TrackID)	(C, 131)	(A, 129)	(A, 141)
SegmentID	1	1	1
VIO	D, E	B	C, E
Targets		C	F, G

Table 16: P-DAO Messages

As a result the RIBs are set as follows:

Node	destination	Origin	Next Hop(s)	TrackID
E	F, G	ND	Neighbor	Any
D	E	ND	Neighbor	Any
C	D	ND	Neighbor	Any
"	E	P-DAO 1	D, E	(C, 131)
B	C	ND	Neighbor	Any
A	B	ND	Neighbor	Any
"	C	P-DAO 2	B, C	(A, 129)
"	E, F, G	P-DAO 3	C, E	(A, 141)

Table 17: RIB setting

The encapsulating headers of packets that are forwarded along the Track between A and B have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	A	B till D then E	(A, 129)
Middle	A	C	(A, 141)
Inner	X	E, F or G	N/A

Table 18: Packet Header Settings

The encapsulating headers of packets that are forwarded along the Track between B and C have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	A	C	(A, 141)
Inner	X	E, F or G	N/A

Table 19: Packet Header Settings

The encapsulating headers of packets that are forwarded along the Track between C and E have the following settings:

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
Outer	C	D till D then E	(C, 131)
Middle	A	E	(A, 141)
Inner	X	E, F or G	N/A

Table 20: Packet Header Settings

As an example, say that A has a packet for F. Using the RIB above:

- \* From P-DAO 3: A encapsulates the packet with destination of F in the Track signaled by P-DAO 3. The outer header has source A, destination C, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 141 from A's namespace. This recurses with:

- \* From P-DAO 2: A encapsulates the packet with destination of C in the Track signaled by P-DAO 2. The outer header has source A, destination B, and a RPI indicating a TrackId of 129 from A's namespace. B decapsulates forwards to C based on a sibling connected route.
- \* From the SRH: C consumes the SRH and makes the destination E.
- \* From P-DAO 1: C encapsulates the packet with destination of E in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E decapsulates.

### 3.6. Complex Tracks

To increase the reliability of the P2P transmission, this specification enables to build a collection of Legs between the same Ingress and Egress Nodes and combine them with the same TrackID, as shown in Figure 7. Legs may cross at the edges of loose hops or remain parallel.

The Segments that join the loose hops of a Leg are installed with the same TrackID as the Leg. But each individual Leg and Segment has its own P-RouteID which allows it to be managed separately. When Legs cross within respective Segment, the next loose hop (the current destination of the packet) indicates which Leg is being followed and a Segment that can reach that next loose hop is selected.

CPF

CPF

CPF

CPF

## Southbound API

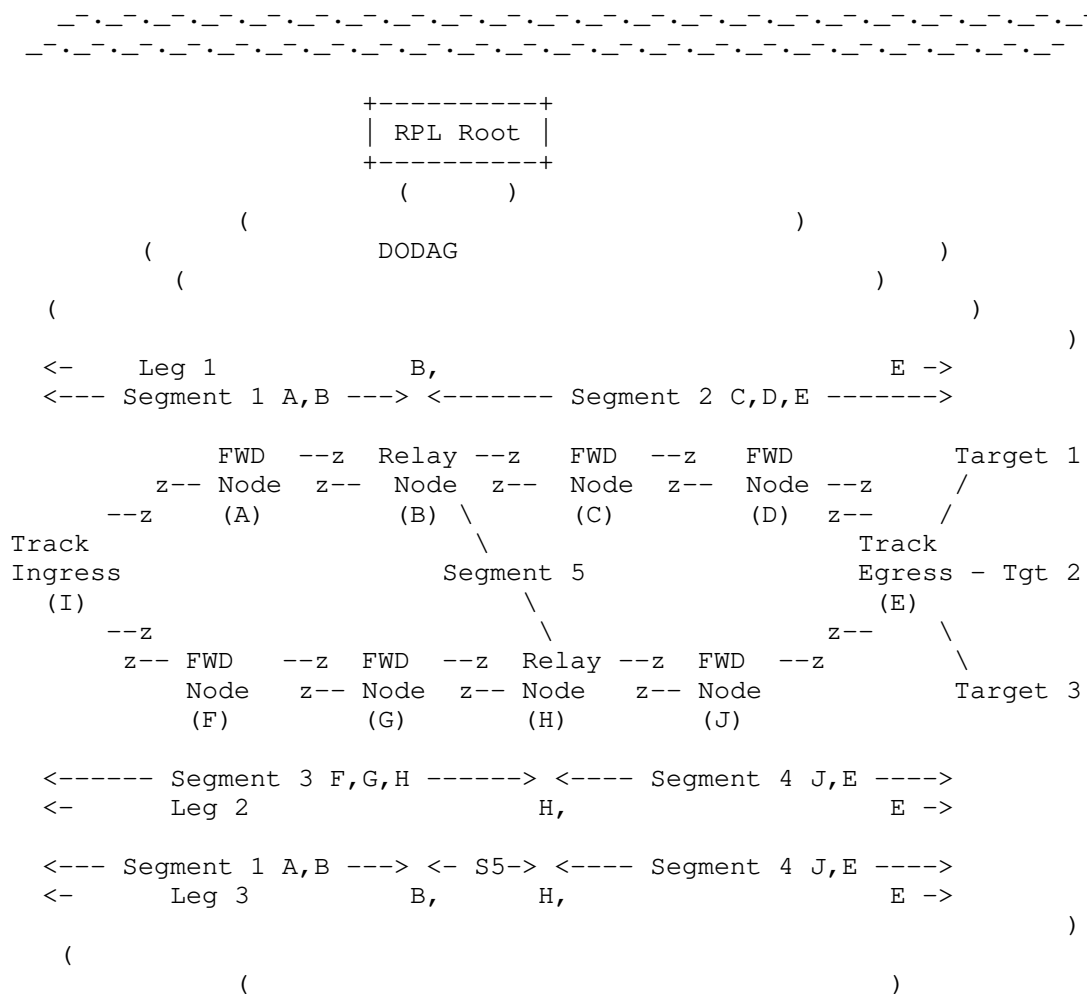


Figure 7: Segments and Tracks

Note that while this specification enables to build both Segments inside a Leg (aka East-West), such as Segment 2 above which is within Leg 1, and Inter-Leg Segments (aka North-South), such as Segment 2 above which joins Leg 1 and Leg 2, it does not signal to the Ingress which Inter-Leg Segments are available, so the use of North-South Segments and associated PAREO functions is currently limited. The only possibility available at this time is to define overlapping Legs

as illustrated in Figure 7, with Leg 3 that is congruent with Leg 1 till node B and congruent with Leg 2 from node H on, abstracting Segment 5 as an East-West Segment.

### 3.7. Scope and Expectations

#### 3.7.1. External Dependencies

This specification expects that the RPL Main DODAG is operated in RPL Non-Storing Mode to sustain the exchanges with the Root. Based on its comprehensive knowledge of the parent-child relationship, the Root can form an abstracted view of the whole DODAG topology. THIS RFC adds the capability for nodes to advertise additional sibling information to complement the topological awareness of the Root to be passed on to the PCE, and enable the PCE to build more / better paths that traverse those siblings.

P-Routes require resources such as routing table space in the routers and bandwidth on the links; the amount of state that is installed in each node must be computed to fit within the node's memory, and the amount of rerouted traffic must fit within the capabilities of the transmission links. The methods used to learn the node capabilities and the resources that are available in the devices and in the network are out of scope for THIS RFC. The method to capture and report the LLN link capacity and reliability statistics are also out of scope. They may be fetched from the nodes through network management functions or other forms of telemetry such as OAM.

#### 3.7.2. Positioning vs. Related IETF Standards

##### 3.7.2.1. Extending 6TiSCH

The "6TiSCH Architecture" [RFC9030] leverages a centralized model that is similar to that of "Deterministic Networking Architecture" [RFC8655], whereby the device resources and capabilities are exposed to an external controller which installs routing states into the network based on its own objective functions that reside in that external entity.

##### 3.7.2.2. Mapping to DetNet

DetNet Forwarding Nodes only understand the simple 1-to-1 forwarding sublayer transport operation along a segment whereas the more sophisticated Relay nodes can also provide service sublayer functions such as Replication and Elimination.

One possible mapping between DetNet and this specification is to signal the Relay Nodes as the hops of a Leg and the forwarding Nodes as the hops in a Segment that join the Relay nodes as illustrated in Figure 7.

#### 3.7.2.3. Leveraging PCE

With DetNet and 6TiSCH, the component of the controller that is responsible of computing routes is a PCE. The PCE computes its routes based on its own objective functions such as described in [RFC4655], and typically controls the routes using the PCE Protocol (PCEP) by [RFC5440]. While this specification expects a PCE and while PCEP might effectively be used between the Root and the PCE, the control protocol between the PCE and the Root is out of scope.

This specification also expects a single PCE with a full view of the network. Distributing the PCE function for a large network is out of scope. This specification uses the RPL Root as a proxy to the PCE. The PCE may be collocated with the Root, or may reside in an external Controller. In that case, the protocol between the Root and the PCE is out of scope and abstracted by / mapped to RPL inside the DODAG; one possibility is for the Root to transmit the RPL DAOs with the SIOs that detail the parent/child and sibling information.

The algorithm to compute the paths and the protocol used by the PCE and the metrics and link statistics involved in the computation are also out of scope. The effectiveness of the route computation by the PCE depends on the quality of the metrics that are reported from the RPL network. Which metrics are used and how they are reported is out of scope, but the expectation is that they are mostly of long-term, statistical nature, and provide visibility on link throughput, latency, stability and availability over relatively long periods.

#### 3.7.2.4. Providing for RAW

The RAW Architecture [RAW-ARCHI] extends the definition of Track, as being composed of East-West directional segments and North-South bidirectional segments, to enable additional path diversity, using Packet ARQ, Replication, Elimination, and Overhearing (PAREO) functions over the available paths, to provide a dynamic balance between the reliability and availability requirements of the flows and the need to conserve energy and spectrum. This specification prepares for RAW by setting up the Tracks, but only forms DODAGs, which are composed of aggregated end-to-end loose source routed Legs, joined by strict routed Segments, all oriented East-West.



The RAW Architecture defines a dataplane extension of the PCE called the Path Selection Engine (PSE), that adapts the use of the path redundancy within a Track to defeat the diverse causes of packet loss. The PSE controls the forwarding operation of the packets within a Track. This specification can use but does not impose a PSE and does not provide the policies that would select which packets are routed through which path within a Track, IOW, how the PSE may use the path redundancy within the Track. By default, the use of the available redundancy is limited to simple load balancing, and all the segments are East-West unidirectional only.

A Track may be set up to reduce the load around the Root, or to enable urgent traffic to flow more directly. This specification does not provide the policies that would decide which flows are routed through which Track. In a Non-Storing Mode RPL Instance, the Main DODAG provides a default route via the Root, and the Tracks provide more specific routes to the Track Targets.

#### 4. Extending existing RFCs

This section explains which changes are extensions to existing specifications, and which changes are amendments to existing specification. It is expected that extensions to existing specifications do not cause existing code on legacy 6LRs to malfunction, as the extensions will simply be ignored. New code is required for an extension. Those 6LRs will be unable to participate in the new mechanisms, but may also cause projected DAOs to be impossible to install. Amendments to existing specifications are situations where there are semantic changes required to existing code, and which may require new unit tests to confirm that legacy operations will continue unaffected.

##### 4.1. Extending RFC 6550

This specification Extends RPL [RPL] to enable the Root to install East-West routes inside a Main DODAG that is operated as Non-Storing Mode. The Root issues a Projected DAO (P-DAO) message (see Section 4.1.1) to the Track Ingress; the P-DAO message contains a new Via Information Option (VIO) that installs a strict or a loose sequence of hops to form respectively a Track Segment or a Track Leg.

The new P-DAO Request (PDR) is a new message detailed in Section 5.1. As per [RPL] section 6, if a node receives this message and it does not understand this new Code, then discards the message. When the root initiates to a node that it has not communicated with before, and to which it does not know if this specification has been implemented (by means such as capabilities), then the root SHOULD request a PDR-ACK.

A P-DAO Request (PDR) message enables a Track Ingress to request the Track from the Root. The resulting Track is also a DODAG for which the Track Ingress is the Root, the owner the address that serves as DODAGID and authoritative for the associated namespace from which the TrackID is selected. In the context of this specification, the installed route appears as a more specific route to the Track Targets, and the Track Ingress routes the packets towards the Targets via the Track using the longest match as usual.

To ensure that the PDR and P-DAO messages can flow at most times, it is RECOMMENDED that the nodes involved in a Track maintain multiple parents in the Main DODAG, advertise them all to the Root, and use them in turn to retry similar packets. It is also RECOMMENDED that the Root uses diverse source route paths to retry similar messages to the nodes in the Track.

#### 4.1.1. Projected DAO

Section 6 of [RPL] introduces the RPL Control Message Options (CMO), including the RPL Target Option (RTO) and Transit Information Option (TIO), which can be placed in RPL messages such as the destination Advertisement Object (DAO). A DAO message signals routing information to one or more Targets indicated in RTOs, providing one hop information at a time in the TIO.

THIS RFC Amends the specification of the DAO to create the P-DAO message. This Amended DAO is signaled with a new "Projected DAO" (P) flag, see Figure 8.

A Projected DAO (P-DAO) is a special DAO message generated by the Root to install a P-Route formed of multiple hops in its DODAG. This provides a RPL-based method to install the Tracks as expected by the 6TiSCH Architecture [RFC9030] as a collection of multiple P-Routes.

The Root MUST source the P-DAO message with its address that serves as DODAGID for the main DODAG. The receiver MUST NOT accept a P-DAO message that is not sent by the Root of its DODAG and MUST ignore such message silently.

The 'P' flag is encoded in bit position 2 (to be confirmed by IANA) of the Flags field in the DAO Base Object. The Root MUST set it to 1 in a Projected DAO message. Otherwise it MUST be set to 0. It is set to 0 in Legacy implementations as specified respectively in Sections 20.11 and 6.4 of [RPL].

The P-DAO is control plane signaling and should not be stuck behind high traffic levels. The expectation is that the P-DAO message is sent as high QoS level, above that of data traffic, typically with the Network Control precedence.

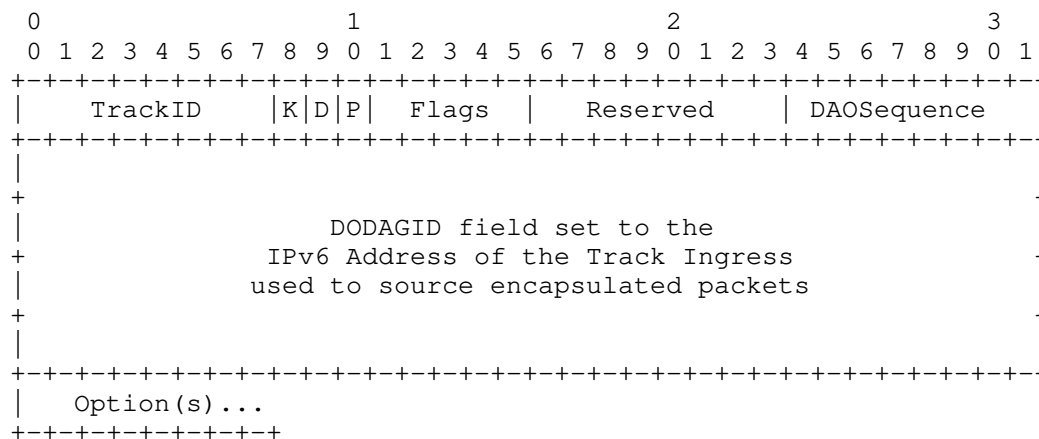


Figure 8: Projected DAO Base Object

New fields:

**TrackID:** The local or global RPLInstanceID of the DODAG that serves as Track, more in Section 6.3

**P:** 1-bit flag (position to be confirmed by IANA).

The 'P' flag is set to 1 by the Root to signal a Projected DAO, and it is set to 0 otherwise.

The D flag is set to one to signal that the DODAGID field is present. It may be set to zero if and only if the destination address of the P-DAO-ACK message is set to the IPv6 address that serves as DODAGID and it MUST be set to one otherwise, meaning that the DODAGID field MUST then be present.

In RPL Non-Storing Mode, the TIO and RTO are combined in a DAO message to inform the DODAG Root of all the edges in the DODAG, which are formed by the directed parent-child relationships. The DAO message signals to the Root that a given parent can be used to reach a given child. The P-DAO message generalizes the DAO to signal to the Track Ingress that a Track for which it is Root can be used to reach children and siblings of the Track Egress. In both cases, options may be factorized and multiple RTOs may be present to signal a collection of children that can be reached through the parent or the Track, respectively.

#### 4.1.2. Projected DAO-ACK

THIS RFC also Amends the DAO-ACK message. The new P flag signals the projected form.

The format of the P-DAO-ACK message is thus as illustrated in Figure 9:

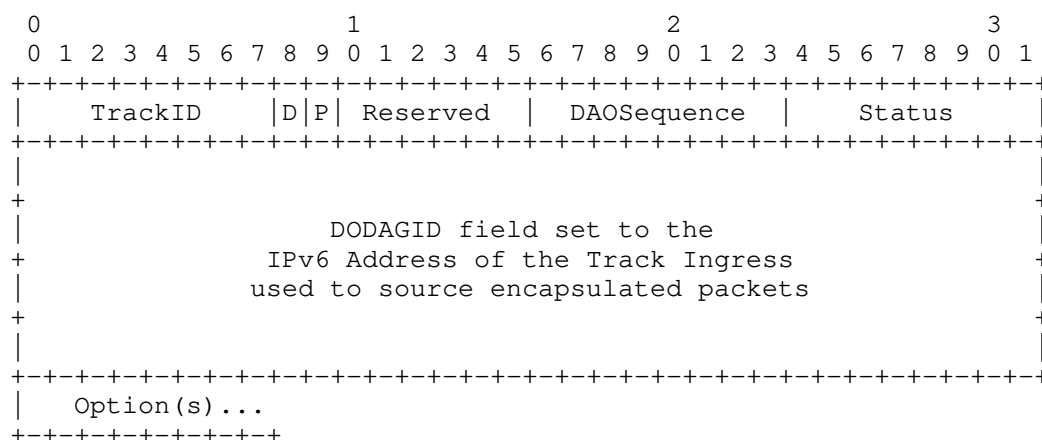


Figure 9: Projected DAO-ACK Base Object

New fields:

**TrackID:** The local or global RPLInstanceID of the DODAG that serves as Track, more in Section 6.3

**P:** 1-bit flag (position to be confirmed by IANA).

The 'P' flag is set to 1 by the Root to signal a Projected DAO, and it is set to 0 otherwise.

The D flag is set to one to signal that the DODAGID field is present. It may be set to zero if and only if the source address of the P-DAO-ACK message is set to the IPv6 address that serves as DODAGID and it MUST be set to one otherwise, meaning that the DODAGID field MUST then be present.

#### 4.1.3. Via Information Option

THIS RFC Extends the CMO to create new objects called the Via Information Options (VIO). The VIOs are the multihop alternative to the TIO, more in Section 5.3. One VIO is the stateful Storing Mode VIO (SM-VIO); an SM-VIO installs a strict hop-by-hop P-Route called a Track Segment. The other is the Non-Storing Mode VIO (NSM-VIO); the NSM-VIO installs a loose source-routed P-Route called a Track Leg at the Track Ingress, which uses that state to encapsulate a packet IPv6\_in\_IPv6 with a new Routing Header (RH) to the Track Egress, more in Section 6.7.

A P-DAO contains one or more RTOs to indicate the Target (destinations) that can be reached via the P-Route, followed by exactly one VIO that signals the sequence of nodes to be followed, more in Section 6. There are two modes of operation for the P-Routes, the Storing Mode and the Non-Storing Mode, see Section 6.4.2 and Section 6.4.3 respectively for more.

#### 4.1.4. Sibling Information Option

This specification Extends the CMO to create the Sibling Information Option (SIO). The SIO is used by a RPL Aware Node (RAN) to advertise a selection of its candidate neighbors as siblings to the Root, more in Section 5.4. The SIO is placed in DAO messages that are sent directly to the Root of the main DODAG.

#### 4.1.5. P-DAO Request

The set of RPL Control Messages is Extended to include the P-DAO Request (PDR) and P-DAO Request Acknowledgement (PDR-ACK). These two new RPL Control Messages enable an RPL-Aware Node to request the establishment of a Track between itself as the Track Ingress Node and a Track Egress. The node makes its request by sending a new P-DAO Request (PDR) Message to the Root. The Root confirms with a new PDR-ACK message back to the requester RAN, see Section 5.1 for more.

#### 4.1.6. Amending the RPI

Sending a Packet within a RPL Local Instance requires the presence of the abstract RPL Packet Information (RPI) described in section 11.2. of [RPL] in the outer IPv6 Header chain (see [RFC9008]). The RPI carries a local RPLInstanceID which, in association with either the source or the destination address in the IPv6 Header, indicates the RPL Instance that the packet follows.

This specification Amends [RPL] to create a new flag that signals that a packet is forwarded along a P-Route.

Projected-Route 'P': 1-bit flag. It is set to 1 in the RPI that is added in the encapsulation when a packet is sent over a Track. It is set to 0 when a packet is forwarded along the main Track, including when the packet follows a Segment that joins loose hops of the Main DODAG. The flag is not mutable en-route.

The encoding of the 'P' flag in native format is shown in Section 4.2 while the compressed format is indicated in Section 4.3.

#### 4.1.7. Additional Flag in the RPL DODAG Configuration Option

The DODAG Configuration Option is defined in Section 6.7.6 of [RPL]. Its purpose is extended to distribute configuration information affecting the construction and maintenance of the DODAG, as well as operational parameters for RPL on the DODAG, through the DODAG. This Option was originally designed with 4 bit positions reserved for future use as Flags.

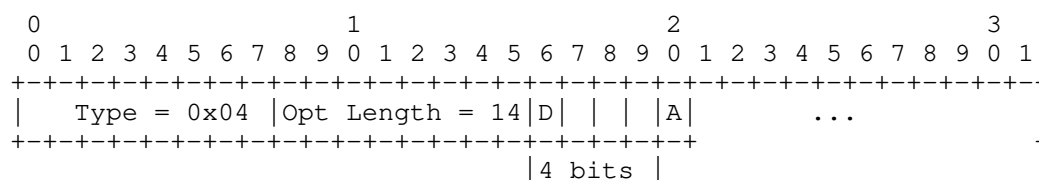


Figure 10: DODAG Configuration Option (Partial View)

This specification Amends the specification to define a new flag "Projected Routes Support" (D). The 'D' flag is encoded in bit position 0 of the reserved Flags in the DODAG Configuration Option (this is the most significant bit) (to be confirmed by IANA but there's little choice). It is set to 0 in legacy implementations as specified respectively in Sections 20.14 and 6.7.6 of [RPL].

The 'D' flag is set to 1 to indicate that this specification is enabled in the network and that the Root will install the requested Tracks when feasible upon a PDR message.

Section 4.1.2. of [RFC9008] updates [RPL] to indicate that the definition of the Flags applies to Mode of Operation values from zero (0) to six (6) only. For a MOP value of 7, the implementation MUST consider that the Root accepts PDR messages and will install Projected Routes.

The RPL DODAG Configuration option is typically placed in a DODAG Information Object (DIO) message. The DIO message propagates down the DODAG to form and then maintain its structure. The DODAG Configuration option is copied unmodified from parents to children.

[RPL] states that:

```
| Nodes other than the DODAG root MUST NOT modify this information
| when propagating the DODAG Configuration option.
```

Therefore, a legacy parent propagates the 'D' flag as set by the root, and when the 'D' flag is set to 1, it is transparently flooded to all the nodes in the DODAG.

#### 4.2. Extending RFC 6553

"The RPL Option for Carrying RPL Information in Data-Plane Datagrams" [RFC6553] describes the RPL Option for use among RPL routers to include the abstract RPL Packet Information (RPI) described in section 11.2. of [RPL] in data packets.

The RPL Option is commonly referred to as the RPI though the RPI is really the abstract information that is transported in the RPL Option. [RFC9008] updated the Option Type from 0x63 to 0x23.

This specification Amends the RPL Option to encode the 'P' flag as follows:

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                     +-----+-----+-----+
                                     | Option Type | Opt Data Len |
+-----+-----+-----+-----+-----+-----+-----+-----+
| O|R|F|P|0|0|0|0| RPLInstanceID | SenderRank |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     (sub-TLVs)                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 11: Amended RPL Option Format

Option Type: 0x23 or 0x63, see [RFC9008]

Opt Data Len: See [RFC6553]

'O', 'R' and 'F' flags: See [RFC6553]. Those flags MUST be set to 0 by the sender and ignored by the receiver if the 'P' flag is set.

Projected-Route 'P': 1-bit flag as defined in Section 4.1.6.

RPLInstanceID: See [RFC6553]. Indicates the TrackId if the 'P' flag is set, as discussed in Section 4.1.1.

SenderRank: See [RFC6553]. This field MUST be set to 0 by the sender and ignored by the receiver if the 'P' flag is set.

#### 4.3. Extending RFC 8138

The 6LoWPAN Routing Header [RFC8138] specification introduces a new IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) [RFC6282] dispatch type for use in 6LoWPAN route-over topologies, which initially covers the needs of RPL data packet compression.

Section 4 of [RFC8138] presents the generic formats of the 6LoWPAN Routing Header (6LoRH) with two forms, one Elective that can be ignored and skipped when the router does not understand it, and one Critical which causes the packet to be dropped when the router cannot process it. The 'E' Flag in the 6LoRH indicates its form. In order to skip the Elective 6LoRHs, their format imposes a fixed expression of the size, whereas the size of a Critical 6LoRH may be signaled in variable forms to enable additional optimizations.

When the [RFC8138] compression is used, the Root of the Main DODAG that sets up the Track also constructs the compressed routing header (SRH-6LoRH) on behalf of the Track Ingress, which saves the complexities of optimizing the SRH-6LoRH encoding in constrained code. The SRH-6LoRH is signaled in the NSM-VIO, in a fashion that it is ready to be placed as is in the packet encapsulation by the Track Ingress.

Section 6.3 of [RFC8138] presents the formats of the 6LoWPAN Routing Header of type 5 (RPI-6LoRH) that compresses the RPI for normal RPL operation. The format of the RPI-6LoRH is not suited for P-Routes since the O,R,F flags are not used and the Rank is unknown and ignored.



This specification extends [RFC8138] to introduce a new 6LoRH, the P-RPI-6LoRH that can be used in either Elective or Critical 6LoRH form, see Table 22 and Table 23 respectively. The new 6LoRH MUST be used as a Critical 6LoRH, unless an SRH-6LoRH is present and controls the routing decision, in which case it MAY be used in Elective form.

The P-RPI-6LoRH is designed to compress the RPI along RPL P-Routes. Its format is as follows:

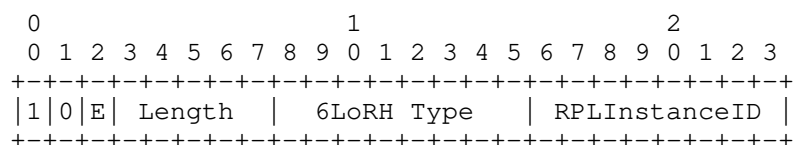


Figure 12: P-RPI-6LoRH Format

Type: IANA is requested to define the same value of the type for both Elective and Critical forms. A type of 8 is suggested.

Elective 'E': See [RFC8138]. The 'E' flag is set to 1 to indicate an Elective 6LoRH, meaning that it can be ignored when forwarding.

RPLInstanceID : In the context of this specification, the RPLInstanceID field signals the TrackID, see Section 3.4 and Section 6.3 .

Section 6.8 details how a a Track Ingress leverages the P-RPI-6LoRH Header as part of the encapsulation of a packet to place it into a Track.

## 5. New RPL Control Messages and Options

### 5.1. New P-DAO Request Control Message

The P-DAO Request (PDR) message is sent by a Node in the Main DODAG to the Root. It is a request to establish or refresh a Track where this node is Track Ingress, and signals whether an acknowledgment called PDR-ACK is requested or not. A positive PDR-ACK indicates that the Track was built and that the Roots commits to maintain the Track for the negotiated lifetime.

The main Root MAY indicate to the Track Ingress that the Track was terminated before its time and to do so, it MUST uses an asynchronous PDR-ACK with an negative status. A status of "Transient Failure" (see Section 11.10) is an indication that the PDR may be retried after a reasonable time that depends on the deployment. Other

negative status values indicate a permanent error; the tentative must be abandoned until a corrective action is taken at the application layer or through network management.

The source IPv6 address of the PDR signals the Track Ingress to-be of the requested Track, and the TrackID is indicated in the message itself. At least one RPL Target Option MUST be present in the message. If more than one RPL Target Option is present, the Root will provide a Track that reaches the first listed Target and a subset of the other Targets; the details of the subset selection are out of scope. The RTO signals the Track Egress, more in Section 6.2.

The RPL Control Code for the PDR is 0x09, to be confirmed by IANA. The format of PDR Base Object is as follows:

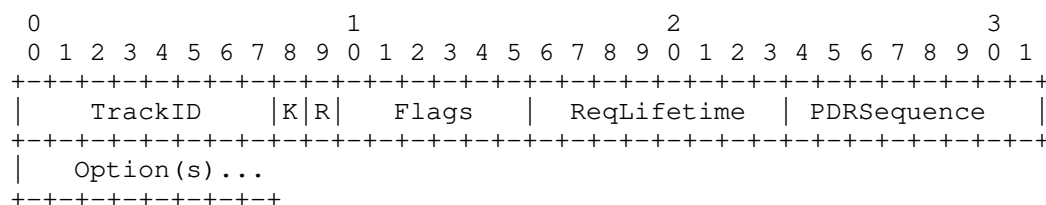


Figure 13: New P-DAO Request Format

**TrackID:** 8-bit field. In the context of this specification, the TrackID field signals the RPLInstanceID of the DODAG formed by the Track, see Section 3.4 and Section 6.3. To allocate a new Track, the Ingress Node must provide a value that is not in use at this time.

**K:** The 'K' flag is set to indicate that the recipient is expected to send a PDR-ACK back.

**R:** The 'R' flag is set to request a Complex Track for redundancy.

**Flags:** Reserved. The Flags field MUST be initialized to zero by the sender and MUST be ignored by the receiver

**ReqLifetime:** 8-bit unsigned integer. The requested lifetime for the Track expressed in Lifetime Units (obtained from the DODAG Configuration option).

A PDR with a fresher PDRSequence refreshes the lifetime, and a PDRLifetime of 0 indicates that the Track should be destroyed, e.g., when the application that requested the Track terminates.

**PDRSequence:** 8-bit wrapping sequence number, obeying the operation

in section 7.2 of [RPL]. The PDRSequence is used to correlate a PDR-ACK message with the PDR message that triggered it. It is incremented at each PDR message and echoed in the PDR-ACK by the Root.

## 5.2. New PDR-ACK Control Message

The new PDR-ACK is sent as a response to a PDR message with the 'K' flag set. The RPL Control Code for the PDR-ACK is 0x0A, to be confirmed by IANA. Its format is as follows:

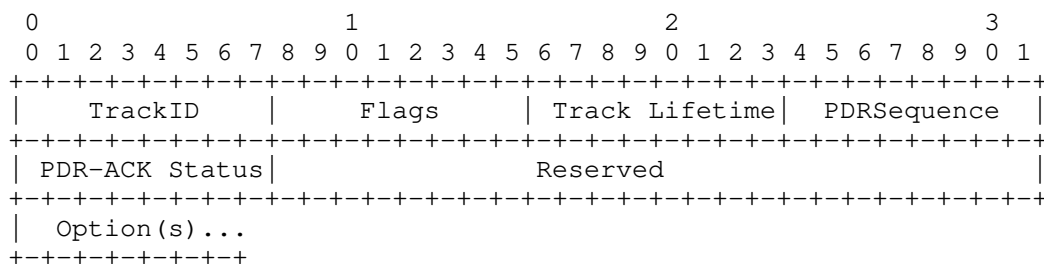


Figure 14: New PDR-ACK Control Message Format

**TrackID:** Set to the TrackID indicated in the TrackID field of the PDR messages that this replies to.

**Flags:** Reserved. The Flags field MUST initialized to zero by the sender and MUST be ignored by the receiver

**Track Lifetime:** Indicates that remaining Lifetime for the Track, expressed in Lifetime Units; the value of zero (0x00) indicates that the Track was destroyed or not created.

**PDRSequence:** 8-bit wrapping sequence number. It is incremented at each PDR message and echoed in the PDR-ACK.

**PDR-ACK Status:** 8-bit field indicating the completion. The PDR-ACK Status is substructured as indicated in Figure 15:

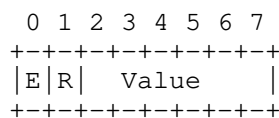


Figure 15: PDR-ACK status Format

**E:** 1-bit flag. Set to indicate a rejection. When not set, the

value of 0 indicates Success/Unqualified Acceptance and other values indicate "not an outright rejection".

R: 1-bit flag. Reserved, MUST be set to 0 by the sender and ignored by the receiver.

Status Value: 6-bit unsigned integer. Values depending on the setting of the 'E' flag, see Table 28 and Table 29.

Reserved: The Reserved field MUST be initialized to zero by the sender and MUST be ignored by the receiver

### 5.3. Via Information Options

A VIO signals the ordered list of IPv6 Via Addresses that constitutes the hops of either a Leg (using Non-Storing Mode) a Segment (using storing mode) of a Track. A Storing Mode P-DAO contains one Storing Mode VIO (SM-VIO) whereas a Non-Storing Mode P-DAO contains one Non-Storing Mode VIO (NSM-VIO)

The duration of the validity of a VIO is indicated in a Segment Lifetime field. A P-DAO message that contains a VIO with a Segment Lifetime of zero is referred as a No-Path P-DAO.

The VIO contains one or more SRH-6LoRH header(s), each formed of a SRH-6LoRH head and a collection of compressed Via Addresses, except in the case of a Non-Storing Mode No-Path P-DAO where the SRH-6LoRH header is not present.

In the case of a SM-VIO, or if [RFC8138] is not used in the data packets, then the Root MUST use only one SRH-6LoRH per Via Information Option, and the compression is the same for all the addresses, as shown in Figure 16, for simplicity.

In case of an NSM-VIO and if [RFC8138] is in use in the Main DODAG, the Root SHOULD optimize the size of the NSM-VIO if using different SRH-6LoRH Types make the VIO globally shorter; this means that more than one SRH-6LoRH may be present.

The format of the Via Information Options is as follows:

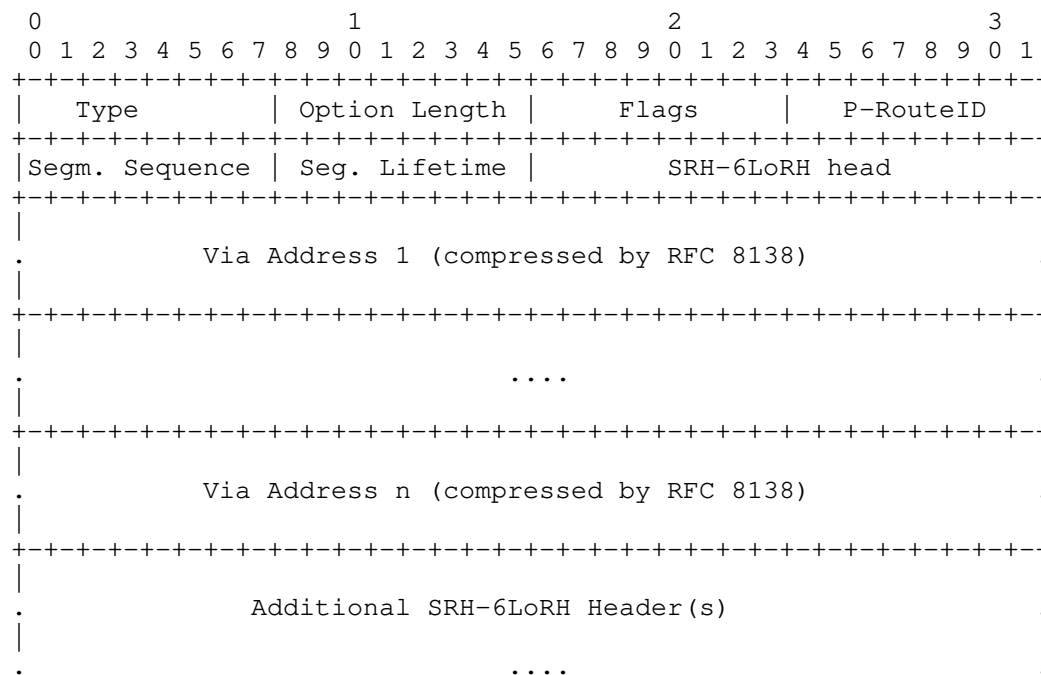


Figure 16: VIO format (uncompressed form)

Option Type: 0x0E for SM-VIO, 0x0F for NSM-VIO (to be confirmed by IANA), see Table 26

Option Length: 8-bit unsigned integer, representing the length in octets of the option, not including the Option Type and Length fields, see section 6.7.1. of [RPL]; the Option Length is variable, depending on the number of Via Addresses and the compression applied.

P-RouteID: 8-bit field that identifies a component of a Track or the Main DODAG as indicated by the TrackID field. The value of 0 is used to signal a Serial Track, i.e., made of a single segment/Leg. In an SM-VIO, the P-RouteID indicates an actual Segment. In an NSM-VIO, it indicates a Leg, that is a serial subTrack that is added to the overall topology of the Track.

Segment Sequence: 8-bit unsigned integer. The Segment Sequence obeys the operation in section 7.2 of [RPL] and the lollipop starts at 255.

When the Root of the DODAG needs to refresh or update a Segment in a Track, it increments the Segment Sequence individually for that Segment.

The Segment information indicated in the VIO deprecates any state for the Segment indicated by the P-RouteID within the indicated Track and sets up the new information.

A VIO with a Segment Sequence that is not as fresh as the current one is ignored.

A VIO for a given DODAGID with the same (TrackID, P-RouteID, Segment Sequence) indicates a retry; it MUST NOT change the Segment and MUST be propagated or answered as the first copy.

Segment Lifetime: 8-bit unsigned integer. The length of time in Lifetime Units (obtained from the Configuration option) that the Segment is usable.

The period starts when a new Segment Sequence is seen. The value of 255 (0xFF) represents infinity. The value of zero (0x00) indicates a loss of reachability.

SRH-6LoRH head: The first 2 bytes of the (first) SRH-6LoRH as shown in Figure 6 of [RFC8138]. As an example, a 6LoRH Type of 4 means that the VIA Addresses are provided in full with no compression.

Via Address: An IPv6 ULA or GUA of a node along the Segment. The VIO contains one or more IPv6 Via Addresses listed in the datapath order from Ingress to Egress. The list is expressed in a compressed form as signaled by the preceding SRH-6LoRH header.

In a Storing Mode P-DAO that updates or removes a section of an already existing Segment, the list in the SM-VIO may represent only the section of the Segment that is being updated; at the extreme, the SM-VIO updates only one node, in which case it contains only one IPv6 address. In all other cases, the list in the VIO MUST be complete.

In the case of an SM-VIO, the list indicates a sequential (strict) path through direct neighbors, the complete list starts at Ingress and ends at Egress, and the nodes listed in the VIO, including the Egress, MAY be considered as implicit Targets.

In the case of an NSM-VIO, the complete list can be loose and excludes the Ingress node, starting at the first loose hop and ending at a Track Egress; the Track Egress MUST be considered as an implicit Target, so it MUST NOT be signaled in a RPL Target Option.

#### 5.4. Sibling Information Option

The Sibling Information Option (SIO) provides indication on siblings that could be used by the Root to form P-Routes. One or more SIO(s) may be placed in the DAO messages that are sent to the Root in Non-Storing Mode.

To advertise a neighbor node, the router MUST have an active Address Registration from that sibling using [RFC8505], for an address (ULA or GUA) that serves as identifier for the node. If this router also registers an address to that sibling, and the link has similar properties in both directions, only the router with the lowest Interface ID in its registered address needs report the SIO, with the B flag set, and the Root will assume symmetry.

The SIO carries a flag (B) that is set when similar performances can be expected both directions, so the routing can consider that the information provided for one direction is valid for both. If the SIO is effectively received from both sides then the B flag MUST be ignored. The policy that describes the performance criteria, and how they are asserted is out of scope. In the absence of an external protocol to assert the link quality, the flag SHOULD NOT be set.

The format of the SIO is as follows:

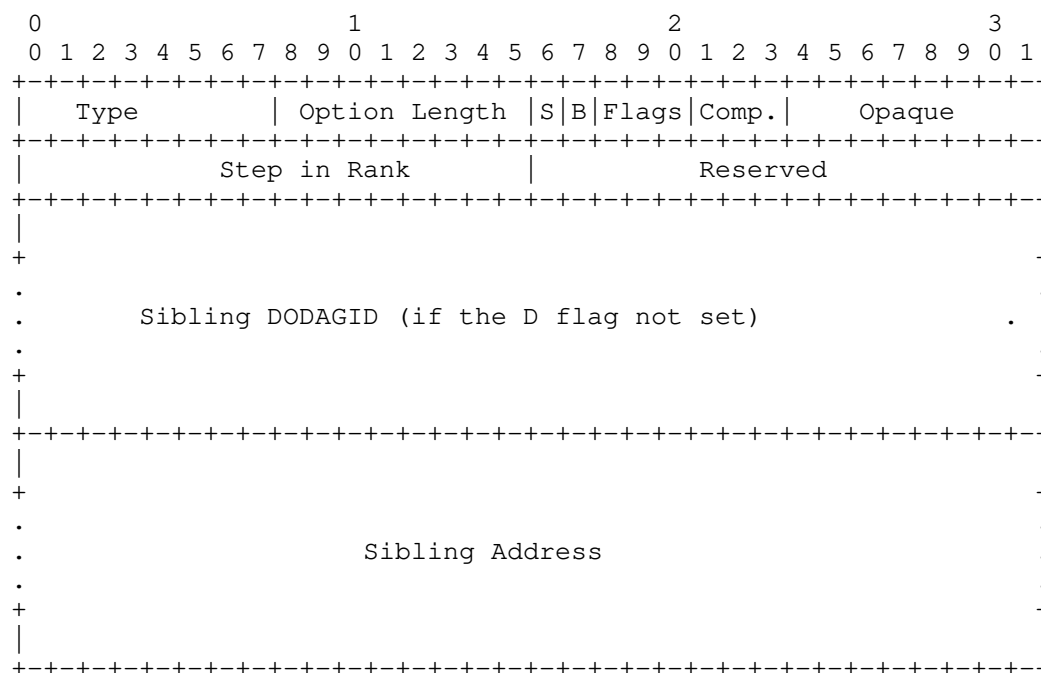


Figure 17: Sibling Information Option Format

Option Type: 0x10 for SIO (to be confirmed by IANA), see =Table 26

Option Length: 8-bit unsigned integer, representing the length in octets of the option, not including the Option Type and Length fields, see section 6.7.1. of [RPL].

Reserved for Flags: MUST be set to zero by the sender and MUST be ignored by the receiver.

B: 1-bit flag that is set to indicate that the connectivity to the sibling is bidirectional and roughly symmetrical. In that case, only one of the siblings may report the SIO for the hop. If 'B' is not set then the SIO only indicates connectivity from the sibling to this node, and does not provide information on the hop from this node to the sibling.

S: 1-bit flag that is set to indicate that sibling belongs to the same DODAG. When not set, the Sibling DODAGID is indicated.

Flags: Reserved. The Flags field MUST initialized to zero by the sender and MUST be ignored by the receiver



Opaque: MAY be used to carry information that the node and the Root understand, e.g., a particular representation of the Link properties such as a proprietary Link Quality Information for packets received from the sibling. An industrial Alliance that uses RPL for a particular use / environment MAY redefine the use of this field to fit its needs.

Compression Type: 3-bit unsigned integer. This is the SRH-6LoRH Type as defined in figure 7 in section 5.1 of [RFC8138] that corresponds to the compression used for the Sibling Address and its DODAGID if resent. The Compression reference is the Root of the Main DODAG.

Step in Rank: 16-bit unsigned integer. This is the Step in Rank [RPL] as computed by the Objective Function between this node and the sibling, that reflects the abstract Rank increment that would be computed by the OF if the sibling was the preferred parent.

Reserved: The Reserved field MUST be initialized to zero by the sender and MUST be ignored by the receiver

Sibling DODAGID: 2 to 16 bytes, the DODAGID of the sibling in a [RFC8138] compressed form as indicated by the Compression Type field. This field is present if and only if the D flag is not set.

Sibling Address: 2 to 16 bytes, an IPv6 Address of the sibling, with a scope that MUST be make it reachable from the Root, e.g., it cannot be a Link Local Address. The IPv6 address is encoded in the [RFC8138] compressed form indicated by the Compression Type field.

An SIO MAY be immediately followed by a DAG Metric Container. In that case the DAG Metric Container provides additional metrics for the hop from the Sibling to this node.

## 6. Root Initiated Routing State

### 6.1. RPL Network Setup

To avoid the need of Path MTU Discovery, 6LoWPAN links are normally defined with a MTU of 1280 (see section 4 of [6LoWPAN]). Injecting packets in a Track typically involves an IP-in-IP encapsulation and additional IPv6 Extension Headers. This may cause a fragmentation if the resulting packets exceeds the MTU that is defined for the RPL domain.

Though fragmentation is possible in a 6LoWPAN LLN, e.g., using [6LoWPAN], [RFC8930], and/or [RFC8931], it is RECOMMENDED to allow an MTU that is larger than 1280 in the main DODAG and allows for the additional headers while exposing only 1280 to the 6LoWPAN Nodes.

## 6.2. Requesting a Track

This specification introduces the PDR message, used by an LLN node to request the formation of a new Track for which this node is Ingress. Note that the namespace for the TrackID is owned by the Ingress node, and in the absence of a PDR, there must be some procedure for the Root to assign TrackIDs in that namespace while avoiding collisions, more in Section 6.3.

The PDR signals the desired TrackID and the duration for which the Track should be established. Upon a PDR, the Root MAY install the Track as requested, in which case it answers with a PDR-ACK indicating the granted Track Lifetime. All the Segments MUST be of a same mode, either Storing or Non-Storing. All the Segments MUST be created with the same TrackID and the same DODAGID signaled in the P-DAO.

The Root designs the Track as it sees best, and updates / changes the Segments overtime to serve the Track as needed. Note that there is no protocol element to notify to the requesting Track Ingress when changes happen deeper down the Track, so they are transparent to the Track Ingress. If the main Root cannot maintain an expected service level, then it needs to tear down the Track completely. The Segment Lifetime in the P-DAO messages does not need to be aligned to the Requested Lifetime in the PDR, or between P-DAO messages for different Segments. The Root may use shorter lifetimes for the Segments and renew them faster than the Track is, or longer lifetimes in which case it will need to tear down the Segments if the Track is not renewed.

When the Track Lifetime that was returned in the PDR-ACK is close to elapse - vs. the trip time from the node to the Root, the requesting node SHOULD resend a PDR using the TrackID in the PDR-ACK to extend the lifetime of the Track, else the Track will time out and the Root will tear down the whole structure.

If the Track fails and cannot be restored, the Root notifies the requesting node asynchronously with a PDR-ACK with a Track Lifetime of 0, indicating that the Track has failed, and a PDR-ACK Status indicating the reason of the fault.

### 6.3. Identifying a Track

RPL defines the concept of an Instance to signal an individual routing topology, and multiple topologies can coexist in the same network. The RPLInstanceID is tagged in the RPI of every packet to signal which topology the packet actually follows.

This draft leverages the RPL Instance model as follows:

- \* The Root MAY use P-DAO messages to add better routes in the main (Global) RPL Instance in conformance with the routing objectives in that Instance.

To achieve this, the Root MAY install a Segment along a path down the main Non-Storing Mode DODAG. This enables a loose source routing and reduces the size of the Routing Header, see Section 3.3.1. The Root MAY also install a Track Leg across the Main DODAG to complement the routing topology.

When adding a P-Route to the RPL Main DODAG, the Root MUST set the RPLInstanceID field of the P-DAO Base Object (see section 6.4.1. of [RPL]) to the RPLInstanceID of the Main DODAG, and MUST NOT use the DODAGID field. A P-Route provides a longer match to the Target Address than the default route via the Root, so it is preferred.

- \* The Root MAY also use P-DAO messages to install a Track as an independent routing topology (say, Traffic Engineered) to achieve particular routing characteristics from an Ingress to an Egress Endpoints. To achieve this, the Root MUST set up a local RPL Instance (see section 5 of [RPL]), and the Local RPLInstanceID serves as TrackID. The TrackID MUST be unique for the IPv6 ULA or GUA of the Track Ingress that serves as DODAGID for the Track.

This way, a Track is uniquely identified by the tuple (DODAGID, TrackID) where the TrackID is always represented with the D flag set to 0 (see also section 5.1. of [RPL]), indicating when used in an RPI that the source address of the IPv6 packet signals the DODAGID.

The P-DAO Base Object MUST indicate the tuple (DODAGID, TrackID) that identifies the Track as shown in Figure 8, and the P-RouteID that identifies the P-Route MUST be signaled in the VIO as shown in Figure 16.

The Track Ingress is the Root of the DODAG ID formed by the local RPL Instance. It owns the namespace of its TrackIDs, so it can pick any unused value to request a new Track with a PDR. In a

particular deployment where PDR are not used, a portion of the namespace can be administratively delegated to the main Root, meaning that the main Root is authoritative for assigning the TrackIDs for the Tracks it creates.

With this specification, the Root is aware of all the active Tracks, so it can also pick any unused value to form Tracks without a PDR. To avoid a collision of the Root and the Track Ingress picking the same value at the same time, it is RECOMMENDED that the Track Ingress starts allocating the ID value of the Local RPLInstanceID (see section 5.1. of [RPL]) used as TrackIDs with the value 0 incrementing, while the Root starts with 63 decrementing.

#### 6.4. Installing a Track

A Serial Track can be installed by a single P-Route that signals the sequence of consecutive nodes, either in Storing Mode as a single-Segment Track, or in Non-Storing Mode as a single-Leg Track. A single-Leg Track can be installed as a loose Non-Storing Mode P-Route, in which case the next loose entry must recursively be reached over a Serial Track.

A Complex Track can be installed as a collection of P-Routes with the same DODAGID and Track ID. The Ingress of a Non-Storing Mode P-Route is the owner and Root of the DODAGID. The Ingress of a Storing Mode P-Route must be either the owner of the DODAGID, or a hop of a Leg of the same Track. In the latter case, the Targets of the P-Route must include the next hop of the Leg if there is one, to ensure forwarding continuity. In the case of a Complex Track, each Segment is maintained independently and asynchronously by the Root, with its own lifetime that may be shorter, the same, or longer than that of the Track.

A route along a Track for which the TrackID is not the RPLInstanceID of the Main DODAG MUST be installed with a higher precedence than the routes along the Main DODAG, meaning that:

- \* Longest match MUST be the prime comparison for routing.
- \* In case of equal length match, the route along the Track MUST be preferred vs. the one along the Main DODAG.
- \* There SHOULD NOT be 2 different Tracks leading to the same Target from same Ingress node, unless there's a policy for selecting which packets use which Track; such policy is out of scope.

- \* A packet that was routed along a Track MUST NOT be routed along the main DODAG again; if the destination is not reachable as a neighbor by the node where the packet exits the Track then the packet MUST be dropped.

#### 6.4.1. Signaling a Projected Route

This draft adds a capability whereby the Root of a main RPL DODAG installs a Track as a collection of P-Routes, using a Projected-DAO (P-DAO) message for each individual Track Leg or Segment. The P-DAO signals a collection of Targets in the RPL Target Option(s) (RTO). Those Targets can be reached via a sequence of routers indicated in a VIO.

Like a classical DAO message, a P-DAO causes a change of state only if it is "new" per section 9.2.2. "Generation of DAO Messages" of the RPL specification [RPL]; this is determined using the Segment Sequence information from the VIO as opposed to the Path Sequence from a TIO. Also, a Segment Lifetime of 0 in a VIO indicates that the P-Route associated to the Segment is to be removed. There are two Modes of operation for the P-Routes, the Storing and the Non-Storing Modes.

A P-DAO message MUST be sent from the address of the Root that serves as DODAGID for the Main DODAG. It MUST contain either exactly one sequence of one or more RTOs followed one VIO, or any number of sequences of one or more RTOs followed by one or more TIOs. The former is the normal expression for this specification, where as the latter corresponds to the variation for lesser constrained environments described in Section 7.2.

A P-DAO that creates or updates a Track Leg MUST be sent to a GUA or a ULA of the Ingress of the Leg; it must contain the full list of hops in the Leg unless the Leg is being removed. A P-DAO that creates a new Track Segment MUST be sent to a GUA or a ULA of the Segment Egress and MUST signal the full list of hops in Segment; a P-DAO that updates (including deletes) a section of a Segment MUST be sent to the first node after the modified Segment and signal the full list of hops in the section starting at the node that immediately precedes the modified section.

In Non-Storing Mode, as discussed in Section 6.4.3, the Root sends the P-DAO to the Track Ingress where the source-routing state is applied, whereas in Storing Mode, the P-DAO is sent to the last node on the installed path and forwarded in the reverse direction, installing a Storing Mode state at each hop, as discussed in Section 6.4.2. In both cases the Track Ingress is the owner of the Track, and it generates the P-DAO-ACK when the installation is successful.

If the 'K' Flag is present in the P-DAO, the P-DAO must be acknowledged using a DAO-ACK that is sent back to the address of the Root from which the P-DAO was received. In most cases, the first node of the Leg, Segment, or updated section of the Segment is the node that sends the acknowledgment. The exception to the rule is when an intermediate node in a Segment fails to forward a Storing Mode P-DAO to the previous node in the SM-VIO.

In a No-Path Non-Storing Mode P-DAO, the SRH-6LoRH MUST NOT be present in the NSM-VIO; the state in the Ingress is erased regardless. In all other cases, a VIO MUST contain at least one Via Address, and a Via Address MUST NOT be present more than once, which would create a loop.

A node that processes a VIO MAY verify whether one of these conditions happen, and when so, it MUST ignore the P-DAO and reject it with a RPL Rejection Status of "Error in VIO" in the DAO-ACK, see Section 11.16.

Other errors than those discussed explicitly that prevent the installing the route are acknowledged with a RPL Rejection Status of "Unqualified Rejection" in the DAO-ACK.

#### 6.4.2. Installing a Track Segment with a Storing Mode P-Route

As illustrated in Figure 18, a Storing Mode P-DAO installs a route along the Segment signaled by the SM-VIO towards the Targets indicated in the Target Options. The Segment is to be included in a DODAG indicated by the P-DAO Base Object, that may be the one formed by the RPL Main DODAG, or a Track associated with a local RPL Instance.

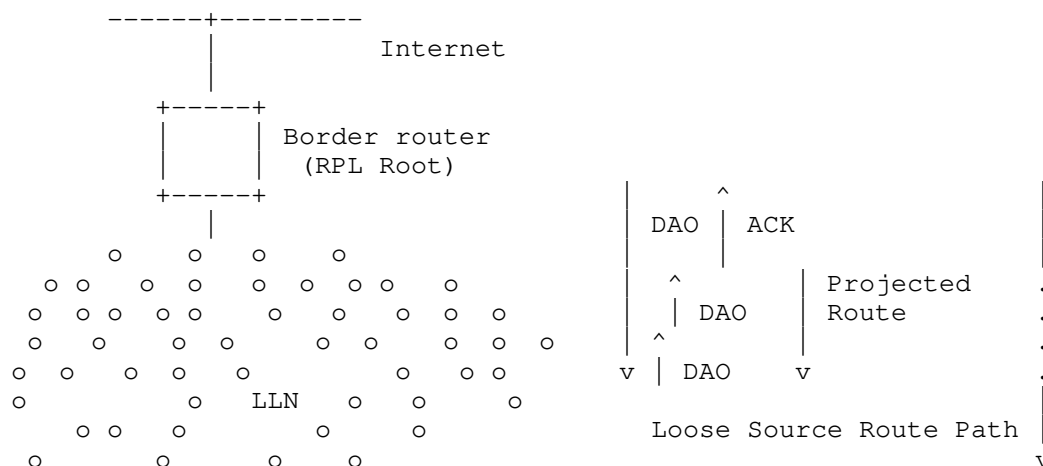


Figure 18: Projecting a route

In order to install the relevant routing state along the Segment , the Root sends a unicast P-DAO message to the Track Egress router of the routing Segment that is being installed. The P-DAO message contains a SM-VIO with the strict sequence of Via Addresses. The SM-VIO follows one or more RTOs indicating the Targets to which the Track leads. The SM-VIO contains a Segment Lifetime for which the state is to be maintained.

The Root sends the P-DAO directly to the Egress node of the Segment. In that P-DAO, the destination IP address matches the last Via Address in the SM-VIO. This is how the Egress recognizes its role. In a similar fashion, the Segment Ingress node recognizes its role as it matches first Via Address in the SM-VIO.

The Egress node of the Segment is the only node in the path that does not install a route in response to the P-DAO; it is expected to be already able to route to the Target(s) based on its existing tables. If one of the Targets is not known, the node MUST answer to the Root with a DAO-ACK listing the unreachable Target(s) in an RTO and a rejection status of "Unreachable Target".

If the Egress node can reach all the Targets, then it forwards the P-DAO with unchanged content to its predecessor in the Segment as indicated in the list of Via Information options, and recursively the message is propagated unchanged along the sequence of routers indicated in the P-DAO, but in the reverse order, from Egress to Ingress.

The address of the predecessor to be used as destination of the propagated DAO message is found in the Via Address the precedes the one that contain the address of the propagating node, which is used as source of the message.

Upon receiving a propagated DAO, all except the Egress router MUST install a route towards the DAO Target(s) via their successor in the SM-VIO. A router that cannot store the routes to all the Targets in a P-DAO MUST reject the P-DAO by sending a DAO-ACK to the Root with a Rejection Status of "Out of Resources" as opposed to forwarding the DAO to its predecessor in the list. The router MAY install additional routes towards the VIA Addresses that are the SM-VIO after self, if any, but in case of a conflict or a lack of resource, the route(s) to the Target(s) are the ones that must be installed in priority.

If a router cannot reach its predecessor in the SM-VIO, the router MUST send the DAO-ACK to the Root with a Rejection Status of "Predecessor Unreachable".

The process continues till the P-DAO is propagated to Ingress router of the Segment, which answers with a DAO-ACK to the Root. The Root always expects a DAO-ACK, either from the Track Ingress with a positive status or from any node along the segment with a negative status. If the DAO-ACK is not received, the Root may retry the DAO with the same TID, or tear down the route.

#### 6.4.3. Installing a Track Leg with a Non-Storing Mode P-Route

As illustrated in Figure 19, a Non-Storing Mode P-DAO installs a source-routed path within the Track indicated by the P-DAO Base Object, towards the Targets indicated in the Target Options. The source-routed path requires a Source-Routing header which implies an IP-in-IP encapsulation to add the SRH to an existing packet. It is sent to the Track Ingress which creates a tunnel associated with the Track, and connected routes over the tunnel to the Targets in the RTO. The tunnel encapsulation MUST incorporate a routing header via the list addresses listed in the VIO in the same order. The content of the NSM-VIO starting at the first SRH-6LoRH header MUST be used verbatim by the Track Ingress when it encapsulates a packet to forward it over the Track.



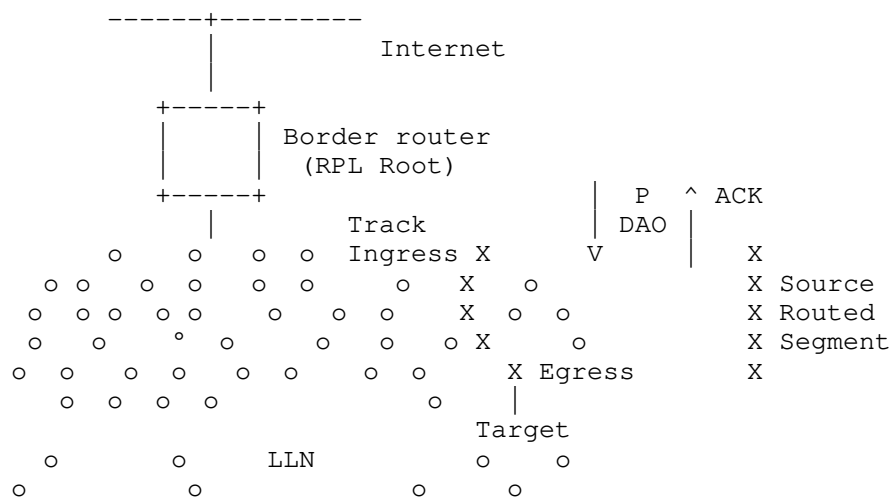


Figure 19: Projecting a Non-Storing Route

The next entry in the source-routed path must be either a neighbor of the previous entry, or reachable as a Target via another P-Route, either Storing or Non-Storing, which implies that the nested P-Route has to be installed before the loose sequence is, and that P-Routes must be installed from the last to the first along the datapath. For instance, a Segment of a Track must be installed before the Leg(s) of the same Track that use it, and stitched Segments must be installed in order from the last that reaches to the Targets to the first.

If the next entry in the loose sequence is reachable over a Storing Mode P-Route, it MUST be the Target of a Segment and the Ingress of a next segment, both already setup; the segments are associated with the same Track, which avoids the need of an additional encapsulation. For instance, in Section 3.5.1.3, Segments A==>B-to-C and C==>D==>E-to-F must be installed with Storing Mode P-DAO messages 1 and 2 before the Track A-->C-->E-to-F that joins them can be installed with Non-Storing Mode P-DAO 3.

Conversely, if it is reachable over a Non-Storing Mode P-Route, the next loose source-routed hop of the inner Track is a Target of a previously installed Track and the Ingress of a next Track, which requires a de- and a re-encapsulation when switching the outer Tracks that join the loose hops. This is exemplified in Section 3.5.2.3 where Non-Storing Mode P-DAO 1 and 2 install strict Tracks that Non-Storing Mode P-DAO 3 joins as a super Track. In such a case, packets are subject to double IP-in-IP encapsulation.

### 6.5. Tearing Down a P-Route

A P-DAO with a lifetime of 0 is interpreted as a No-Path DAO and results in cleaning up existing state as opposed to refreshing an existing one or installing a new one. To tear down a Track, the Root must tear down all the Track Segments and Legs that compose it one by one.

Since the state about a Leg of a Track is located only on the Ingress Node, the Root cleans up the Leg by sending an NSM-VIO to the Ingress indicating the TrackID and the P-RouteID of the Leg being removed, a Segment Lifetime of 0 and a newer Segment Sequence. The SRH-6LoRH with the Via Addresses in the NSM-VIO are not needed; it SHOULD not be placed in the message and MUST be ignored by the receiver. Upon that NSM-VIO, the Ingress node removes all state for that Track if any, and replies positively anyway.

The Root cleans up a section of a Segment by sending an SM-VIO to the last node of the Segment, with the TrackID and the P-RouteID of the Segment being updated, a Segment Lifetime of zero (0) and a newer Segment Sequence. The Via Addresses in the SM-VIO indicates the section of the Segment being modified, from the first to the last node that is impacted. This can be the whole Segment if it is totally removed, or a sequence of one or more nodes that have been bypassed by a Segment update.

The No-Path P-DAO is forwarded normally along the reverse list, even if the intermediate node does not find a Segment state to clean up. This results in cleaning up the existing Segment state if any, as opposed to refreshing an existing one or installing a new one.

### 6.6. Maintaining a Track

Repathing a Track Segment or Leg may cause jitter and packet misordering. For critical flows that require timely and/or in-order delivery, it might be necessary to deploy the PAREO functions [RAW-ARCHI] over a highly redundant Track. This specification allows to use more than one Leg for a Track, and 1+N packet redundancy.

This section provides the steps to ensure that no packet is lost due to the operation itself. This is ensured by installing the new section from its last node to the first, so when an intermediate node installs a route along the new section, all the downstream nodes in the section have already installed their own. The disabled section is removed when the packets in-flight are forwarded along the new section as well.

#### 6.6.1. Maintaining a Track Segment

To modify a section of a Segment between a first node and a second, downstream node (which can be the Ingress and Egress), while conserving those nodes in the Segment, the Root sends an SM-VIO to the second node indicating the sequence of nodes in the new section of the Segment. The SM-VIO indicates the TrackID and the P-RouteID of the Segment being updated, and a newer Segment Sequence. The P-DAO is propagated from the second to the first node and on the way, it updates the state on the nodes that are common to the old and the new section of the Segment and creates a state in the new nodes.

When the state is updated in an intermediate node, that node might still receive packets that were in flight from the Ingress to self over the old section of the Segment. Since the remainder of the Segment is already updated, the packets are forwarded along the new version of the Segment from that node on.

After a reasonable time to enable the deprecated sections to empty, the Root tears down the remaining section(s) of the old segments are torn down as described in Section 6.5.

#### 6.6.2. Maintaining a Track Leg

This specification allows the Root to add Legs to a Track by sending a Non-Storing Mode P-DAO to the Ingress associated to the same TrackID, and a new Segment ID. If the Leg is loose, then the Segments that join the hops must be created first. It makes sense to add a new Leg before removing one that is becoming excessively lossy, and switch to the new Leg before removing the old. Dropping a Track before the new one is installed would reroute the traffic via the root; this may augment the latency beyond acceptable thresholds, and load the network near the root. This may also cause loops in the case of stitched Tracks; the packets that cannot be injected in the second Track may be routed back at reinjected at the Ingress of the first.

It is also possible to update a Track Leg by sending a Non-Storing Mode P-DAO to the Ingress with the same Segment ID, an incremented Segment Sequence, and the new complete list of hops in the NSM-VIO. Updating a live Leg means changing one or more of the intermediate loose hops, and involves laying out new Segments from and to the new loose hops before the NSM-VIO for the new Leg is issued.

Packets that are in flight over the old version of the Track Leg still follow the old source route path over the old Segments. After a reasonable time to enable the deprecated Segments to empty, the Root tears down those Segments as described in Section 6.5.

## 6.7. Encapsulating and Forwarding Along a Track

When injecting a packet in a Track, the Ingress router must encapsulate the packet using IP-in-IP to add the Source Routing Header with the final destination set to the Track Egress.

All properties of a Track operations are inherited from the main RPL Instance that is used to install the Track. For instance, the use of compression per [RFC8138] is determined by whether it is used in the RPL Main DODAG, e.g., by setting the "T" flag [RFC9035] in the RPL configuration option.

The Track Ingress that places a packet in a Track encapsulates it with an IP-in-IP header, a Routing Header, and an IPv6 Hop-by-Hop Option Header that contains the RPL Packet Information (RPI) as follows:

- \* In the uncompressed form the source of the packet is the address that this router uses as DODAGID for the Track, the destination is the first Via Address in the NSM-VIO, and the RH is a Source Routing Header (SRH) [RFC6554] that contains the list of the remaining Via Addresses terminating by the Track Egress.
- \* The preferred alternate in a network where 6LoWPAN Header Compression [RFC6282] is used is to leverage "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch" [RFC8025] to compress the RPL artifacts as indicated in [RFC8138].

In that case, the source routed header is the exact copy of the (chain of) SRH-6LoRH found in the NSM-VIO, also terminating by the Track Egress. The RPI-6LoRH is appended next, followed by an IP-in-IP 6LoRH Header that indicates the Ingress router in the Encapsulator Address field, see as a similar case Figure 20 of [RFC9035].

To signal the Track in the packet, this specification leverages the RPL Forwarding model follows:

- \* In the data packets, the Track DODAGID and the TrackID MUST be respectively signaled as the IPv6 Source Address and the RPLInstanceID field of the RPI that MUST be placed in the outer chain of IPv6 Headers.

The RPI carries a local RPLInstanceID called the TrackID, which, in association with the DODAGID, indicates the Track along which the packet is forwarded.

The D flag in the RPLInstanceID MUST be set to 0 to indicate that the source address in the IPv6 header is set to the DODAGID, more in Section 6.3.

- \* This draft conforms to the principles of [RFC9008] with regards to packet forwarding and encapsulation along a Track, as follows:
  - With this draft, the Track is a RPL DODAG. From the perspective of that DODAG, the Track Ingress is the Root, the Track Egress is a RPL-Aware 6LR, and neighbors of the Track Egress that can be reached via the Track, but are external to it, are external destinations and treated as RPL-Unaware Leaves (RULs). The encapsulation rules in [RFC9008] apply.
  - If the Track Ingress is the originator of the packet and the Track Egress is the destination of the packet, there is no need for an encapsulation.
  - So the Track Ingress must encapsulate the traffic that it did not originate, and add an RPI.

A packet that is being routed over the RPL Instance associated to a first Non-Storing Mode Track MAY be placed (encapsulated) in a second Track to cover one loose hop of the first Track as discussed in more details Section 3.5.2.3. On the other hand, a Storing Mode Track must be strict and a packet that it placed in a Storing Mode Track MUST follow that Track till the Track Egress.

The forwarding of a packet along a track will fail if the Track continuity is broken, e.g.:

- \* In the case of a strict path along a Segment, if the next strict hop is not reachable, the packet is dropped.
- \* In the case of a loose source-routed path, when the loose next hop is not a neighbor, there must be a Segment of the same Track to that loose next hop. When that is the case the packet is forwarded to the next hop along that segment, or a common neighbor with the loose next hop, on which case the packet is forwarded to that neighbor, or another Track to the loose next hop for which this node or a neighbor is Ingress; in the last case, another encapsulation takes place and the process possibly recurses; otherwise the packet is dropped.

- \* When a Track Egress extracts a packet from a Track (decapsulates the packet), the destination of the inner packet must be either this node or a direct neighbor, or a Target of another Segment of the same Track for which this node is Ingress, otherwise the packet MUST be dropped.

In case of a failure forwarding a packet along a Segment, e.g., the next hop is unreachable, the node that discovers the fault MUST send an ICMPv6 Error message [RFC4443] to the Root, with a new Code "Error in P-Route" (See Section 11.15). The Root can then repair by updating the broken Segment and/or Tracks, and in the case of a broken Segment, remove the leftover sections of the segment using SM-VIOs with a lifetime of 0 indicating the section of one or more nodes being removed (See Section 6.6).

In case of a permanent forwarding error along a Source Route path, the node that fails to forward SHOULD send an ICMP error with a code "Error in Source Routing Header" back to the source of the packet, as described in section 11.2.2.3. of [RPL]. Upon this message, the encapsulating node SHOULD stop using the source route path for a reasonable period of time which duration depends on the deployment, and it SHOULD send an ICMP message with a Code "Error in P-Route" to the Root. Failure to follow these steps may result in packet loss and wasted resources along the source route path that is broken.

Either way, the ICMP message MUST be throttled in case of consecutive occurrences. It MUST be sourced at the ULA or a GUA that is used in this Track for the source node, so the Root can establish where the error happened.

The portion of the invoking packet that is sent back in the ICMP message SHOULD record at least up to the RH if one is present, and this hop of the RH SHOULD be consumed by this node so that the destination in the IPv6 header is the next hop that this node could not reach. If a 6LoWPAN Routing Header (6LoRH) [RFC8138] is used to carry the IPv6 routing information in the outer header then that whole 6LoRH information SHOULD be present in the ICMP message.

#### 6.8. Compression of the RPL Artifacts

When using [RFC8138] in the Main DODAG operated in Non-Storing Mode in a 6LoWPAN LLN, a typical packet that circulates in the Main DODAG is formatted as shown in Figure 20, representing the case where :

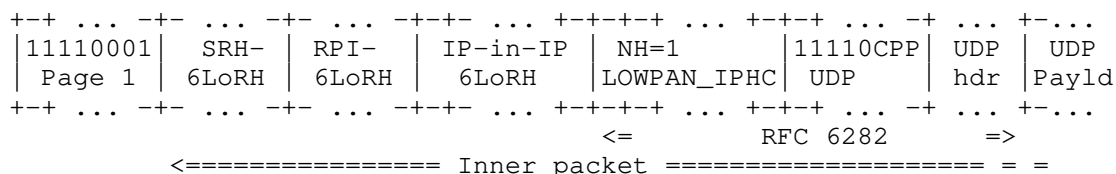


Figure 20: A Packet as Forwarded along the Main DODAG

Since there is no page switch between the encapsulated packet and the encapsulation, the first octet of the compressed packet that acts as page selector is actually removed at encapsulation, so the inner packet used in the descriptions below start with the SRH-6LoRH, and is verbatim the packet represented in Figure 20 from the second octet on.

When encapsulating that inner packet to place it in the Track, the first header that the Ingress appends at the head of the inner packet is an IP-in-IP 6LoRH Header; in that header, the encapsulator address, which maps to the IPv6 source address in the uncompressed form, contains a GUA or ULA IPv6 address of the Ingress node that serves as DODAG ID for the Track, expressed in the compressed form and using the DODAGID of the Main DODAG as compression reference. If the address is compressed to 2 bytes, the resulting value for the Length field shown in Figure 21 is 3, meaning that the SRH-6LoRH as a whole is 5-octets long.

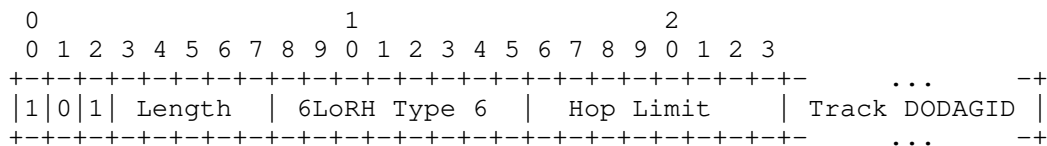


Figure 21: The IP-in-IP 6LoRH Header

At the head of the resulting sequence of bytes, the track Ingress then adds the RPI that carries the TrackID as RPLinstanceID as a P-RPI-6LoRH Header, as illustrated in Figure 12, using the TrackID as RPLInstanceID. Combined with the IP-in-IP 6LoRH Header, this allows to identify the Track without ambiguity.

The SRH-6LoRH is then added at the head of the resulting sequence of bytes as a verbatim copy of the content of the SR-VIO that signaled the selected Track Leg.

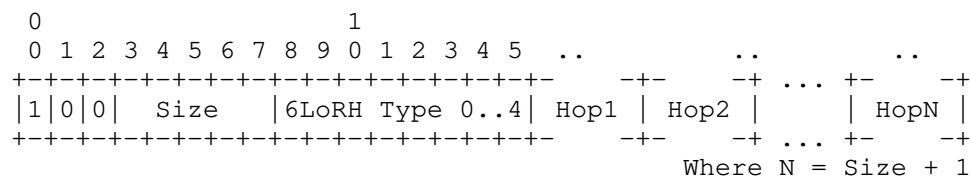
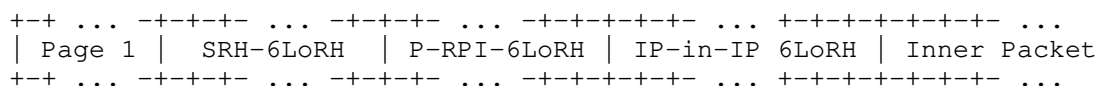


Figure 22: The SRH 6LoRH Header

The format of the resulting encapsulated packet in [RFC8138] compressed form is illustrated in Figure 23:



Signals :   Loose Hops :      TrackID    :    Track DODAGID :

Figure 23: A Packet as Forwarded along a Track

## 7. Lesser Constrained Variations

### 7.1. Storing Mode Main DODAG

This specification expects that the Main DODAG is operated in Non-Storing Mode. The reasons for that limitation are mostly related to LLN operations, power and spectrum conservation:

- \* In Non-Storing Mode The Root already possesses the DODAG topology, so the additional topological information is reduced to the siblings.
- \* The downwards routes are updated with unicast messages to the Root, which ensures that the Root can reach back to the LLN nodes after a repair faster than in the case of Storing Mode. Also the Root can control the use of the path diversity in the DODAG to reach to the LLN nodes. For both reasons, Non-Storing Mode provides better capabilities for the Root to maintain the P-Routes.
- \* When the Main DODAG is operated in Non-Storing Mode, P-Routes enable loose Source Routing, which is only an advantage in that mode. Storing Mode does not use Source Routing Headers, and does not derive the same benefits from this capability.

On the other hand, since RPL is a Layer-3 routing protocol, its applicability extends beyond LLNs to a generic IP network. RPL requires fewer resources than alternative IGPs like OSPF, ISIS,



EIGRP, BABEL or RIP at the expense of a route stretch vs. the shortest path routes to a destination that those protocols compute. P-Routes add the capability to install shortest and/or constrained routes to special destinations such as discussed in section A.9.4. of the ANIMA ACP [RFC8994].

In a powered and wired network, when enough memory to store the needed routes is available, the RPL Storing Mode proposes a better trade-off than the Non-Storing, as it reduces the route stretch and lowers the load on the Root. In that case, the control path between the Root and the LLN nodes is highly available compared to LLNs, and the nodes can be reached to maintain the P-Routes at most times.

This section specifies the additions that are needed to support Projected Routes when the Main DODAG is operated in Storing Mode. As long as the RPI can be processed adequately by the dataplane, the changes to this specification are limited to the DAO message. The Track structure, routes and forwarding operations remain the same. Since there is no capability negotiation, the expectation is that all the nodes in the network support this specification in the same fashion, or are configured the same way through management.

In Storing Mode, the Root misses the Child to Parent relationship that forms the Main DODAG, as well as the sibling information. To provide that knowledge the nodes in the network MUST send additional DAO messages that are unicast to the Root as Non-Storing DAO messages are.

In the DAO message, the originating router advertises a set of neighbor nodes using Sibling Information Options (SIO)s, regardless of the relative position in the DODAG of the advertised node vs. this router.

The DAO message MUST be formed as follows:

- \* The originating router is identified by the source address of the DAO. That address MUST be the one that this router registers to neighbor routers so the Root can correlate the DAOs from those routers when they advertise this router as their neighbor. The DAO contains one or more sequences of one Transit Information Option and one or more Sibling Information Options. There is no RPL Target Option so the Root is not confused into adding a Storing Mode route to the Target.

- \* The TIO is formed as in Storing Mode, and the Parent Address is not present. The Path Sequence and Path Lifetime fields are aligned with the values used in the Address Registration of the node(s) advertised in the SIO, as explained in Section 9.1. of [RFC9010]. Having similar values in all nodes allows to factorise the TIO for multiple SIOs as done with [RPL].
- \* The TIO is followed by one or more SIOs that provide an address (ULA or GUA) of the advertised neighbor node.

But the RPL routing information headers may not be supported on all type of routed network infrastructures, especially not in high-speed routers. When the RPI is not supported in the dataplane, there cannot be local RPL Instances and RPL can only operate as a single topology (the Main DODAG). The RPL Instance is that of the Main DODAG and the Ingress node that encapsulates is not the Root. The routes along the Tracks are alternate routes to those available along the Main DODAG. They MAY conflict with routes to children and MUST take precedence in the routing table. The Targets MUST be adjacent to the Track Egress to avoid loops that may form if the packet is reinjected in the Main DODAG.

## 7.2. A Track as a Full DODAG

This specification builds parallel or crossing Track Legs as opposed to a more complex DODAG with interconnections at any place desirable. The reason for that limitation is related to constrained node operations, and capability to store large amount of topological information and compute complex paths:

- \* With this specification, the node in the LLN has no topological awareness, and does not need to maintain dynamic information about the link quality and availability.
- \* The Root has a complete topological information and statistical metrics that allow it or its PCE to perform a global optimization of all Tracks in its DODAG. Based on that information, the Root computes the Track Leg and predigest the source route paths.
- \* The node merely selects one of the proposed paths and applies the associated pre-computed routing header in the encapsulation. This alleviates both the complexity of computing a path and the compressed form of the routing header.

The RAW Architecture [RAW-ARCHI] actually expects the PSE at the Track Ingress to react to changes in the forwarding conditions along the Track, and reroute packets to maintain the required degree of reliability. To achieve this, the PSE need the full richness of a DODAG to form any path that could make meet the Service Level Objective (SLO).

This section specifies the additions that are needed to turn the Track into a full DODAG and enable the main Root to provide the necessary topological information to the Track Ingress. The expectation is that the metrics that the PSE uses are of an order other than that of the PCE, because of the difference of time scale between routing and forwarding, more in [RAW-ARCHI]. It follows that the PSE will learn the metrics it needs from an alternate source, e.g., OAM frames.

To pass the topological information to the Ingress, the Root uses a P-DAO messages that contains sequences of Target and Transit Information options that collectively represent the Track, expressed in the same fashion as in classical Non-Storing Mode. The difference is that the Root is the source as opposed to the destination, and can report information on many Targets, possibly the full Track, with one P-DAO.

Note that the Path Sequence and Lifetime in the TIO are selected by the Root, and that the Target/Transit information tuples in the P-DAO are not those received by the Root in the DAO messages about the said Targets. The Track may follow sibling routes and does not need to be congruent with the Main DODAG.

## 8. Profiles

THIS RFC provides a set of tools that may or may not be needed by an implementation depending on the type of application it serves. This sections described profiles that can be implemented separately and can be used to discriminate what an implementation can and cannot do. This section describes profiles that enable to implement only a portion of this specification to meet a particular use case.

Profiles 0 to 2 operate in the Main RPL Instance and do not require the support of local RPL Instances or the indication of the RPL Instance in the data plane. Profile 3 and above leverage Local RPL Instances to build arbitrary Tracks Rooted at the Track Ingress and using its namespace for TrackID.

Profiles 0 and 1 are REQUIRED by all implementations that may be used in LLNs; Profiles 1 leverages Storing Mode to reduce the size of the Source Route Header in the most common LLN deployments. Profile 2 is

RECOMMENDED in high speed / wired environment to enable traffic Engineering and network automation. All the other profile / environment combinations are OPTIONAL.

**Profile 0** Profile 0 is the Legacy support of [RPL] Non-Storing Mode, with default routing Northwards (up) and strict source routing Southwards (down the main DODAG). It provides the minimal common functionality that must be implemented as a prerequisite to all the Track-supporting profiles. The other Profiles extend Profile 0 with selected capabilities that this specification introduces on top.

**Profile 1 (Storing Mode P-Route Segments along the Main DODAG)** Profile 1 does not create new paths; compared to Profile 0, it combines Storing and Non-Storing Modes to balance the size of the Routing Header in the packet and the amount of state in the intermediate routers in a Non-Storing Mode RPL DODAG.

**Profile 2 (Non-Storing Mode P-Route Segments along the Main DODAG)** Profile 2 extends Profile 0 with Strict Source-Routing Non-Storing Mode P-Routes along the Main DODAG, which is the same as Profile 1 but using NSM VIOs as opposed to SM VIOs. Profile 2 provides the same capability to compress the SRH in packets down the Main DODAG as Profile 1, but it requires an encapsulation, in order to insert an additional SRH between the loose source routing hops. In that case, the Tracks MUST be installed as subTracks of the Main DODAG, the main RPL Instance MUST be used as TrackID, and the Ingress node that encapsulates is not the Root as it does not own the DODAGID.

**Profile 3** In order to form the best path possible, those Profiles require the support of Sibling Information Option to inform the Root of additional possible hops. Profile 3 extends Profile 1 with additional Storing Mode P-Routes that install segments that do not follow the Main DODAG. If the Segment Ingress (in the SM-VIO) is the same as the IPv6 Address of the Track Ingress (in the projected DAO base Object), the P-DAO creates an implicit Track between the Segment Ingress and the Segment Egress.

**Profile 4** Profile 4 extends Profile 2 with Strict Source-Routing Non-Storing Mode P-Routes to form East-West Tracks that are inside the Main DODAG but do not necessarily follow it. A Track is formed as one or more strict source source routed paths between the Root that is the Track Ingress, and the Track Egress that is the last node.

**Profile 5** Profile 5 Combines Profile 4 with Profile 1 and enables to

loose source routing between the Ingress and the Egress of the Track. As in Profile 1, Storing Mode P-Routes connect the dots in the loose source route.

Profile 6 Profile 6 Combines Profile 4 with Profile 2 and also enables to loose source routing between the Ingress and the Egress of the Track.

Profile 7 Profile 7 implements profile 5 in a Main DODAG that is operated in Storing Mode as presented in Section 7.1. As in Profile 1 and 2, the TrackID is the RPLInstanceID of the Main DODAG. Longest match rules decide whether a packet is sent along the Main DODAG or rerouted in a track.

Profile 8 Profile 8 is offered in preparation of the RAW work, and for use cases where an arbitrary node in the network can afford the same code complexity as the RPL Root in a traditional deployment. It offers a full DODAG visibility to the Track Ingress as specified in Section 7.2 in a Non-Storing Mode Main DODAG.

Profile 9 Profile 9 combines profiles 7 and 8, operating the Track as a full DODAG within a Storing Mode Main DODAG, using only the Main DODAG RPLInstanceID as TrackID.

## 9. Backwards Compatibility

This specification can operate in a mixed network where some nodes support it and some do not. There are restrictions, though. All nodes that need to process a P-DAO MUST support this specification. As discussed in Section 3.7.1, how the root knows whether the nodes capabilities and whether it support this specification is out of scope.

This specification defines the 'D' flag in the RPL DODAG Configuration Option (see Section 4.1.7) to signal that the RPL nodes can request the creation of Tracks. The requester may not know whether the Track can effectively be constructed, and whether enough nodes along the preferred paths support this specification. Therefore it makes sense to only set the 'D' flags in DIO when the conditions of success are in place, in particular when all the nodes that could be on path of tracks are upgraded.

## 10. Security Considerations

It is worth noting that with [RPL], every node in the LLN is RPL-aware and can inject any RPL-based attack in the network. This draft uses messages that are already present in RPL [RPL] 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.

The LLN nodes depend on the 6LBR and the RPL participants for their operation. A trust model is necessary to ensure that the right devices are acting in these roles, so as to avoid threats such as black-holing, (see [RFC7416] section 7). This trust model could be at a minimum based on a Layer-2 Secure joining and the Link-Layer security. This is a generic 6LoWPAN requirement, see Req5.1 in Appendix B.5 of [RFC8505].

In a general manner, the Security Considerations in [RPL], and [RFC7416] apply to this specification as well. The Link-Layer security is needed in particular to prevent Denial-Of-Service attacks whereby a rogue router creates a high churn in the RPL network by constantly injected forged P-DAO messages and using up all the available storage in the attacked routers.

With this specification, only the Root may generate P-DAO messages. PDR messages may only be sent to the Root. This specification expects that the communication with the Root is authenticated but does enforce which method is used.

Additionally, the trust model could include a role validation (e.g., using a role-based authorization) to ensure that the node that claims to be a RPL Root is entitled to do so. That trust should propagate from Egress to Ingress in the case of a Storing Mode P-DAO.

This specification suggests some validation of the VIO to prevent basic loops by avoiding that a node appears twice. But that is only a minimal protection. Arguably, an attacker that can inject P-DAOs can reroute any traffic and deplete critical resources such as spectrum and battery in the LLN rapidly.

## 11. IANA Considerations

### 11.1. RPL DODAG Configuration Option Flag

IANA is requested to assign a flag from the "DODAG Configuration Option Flags for MOP 0..6" [RFC9010] registry as follows:

Bit Number	Capability Description	Reference
0 (suggested)	Projected Routes Support (D)	THIS RFC

Table 21: New DODAG Configuration Option Flag

IANA is requested to add [THIS RFC] as a reference for MOP 7 in the RPL Mode of Operation registry.

#### 11.2. Elective 6LoWPAN Routing Header Type

THIS RFC updates the IANA registry titled "Elective 6LoWPAN Routing Header Type" that was created for [RFC8138] and assigns the following value:

Value	Description	Reference
8 (Suggested)	P-RPI-6LoRH	THIS RFC

Table 22: New Elective 6LoWPAN Routing Header Type

#### 11.3. Critical 6LoWPAN Routing Header Type

THIS RFC updates the IANA registry titled "Critical 6LoWPAN Routing Header Type" that was created for [RFC8138] and assigns the following value:

Value	Description	Reference
8 (Suggested)	P-RPI-6LoRH	THIS RFC

Table 23: New Critical 6LoWPAN Routing Header Type

#### 11.4. Subregistry For The RPL Option Flags

IANA is required to create a subregistry for the 8-bit RPL Option Flags field, as detailed in Figure 11, under the "Routing Protocol for Low Power and Lossy Networks (RPL)" registry. The bits are indexed from 0 (leftmost) to 7. Each bit is Tracked with the following qualities:

- \* Bit number (counting from bit 0 as the most significant bit)
- \* Indication When Set
- \* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 27:

Bit number	Indication When Set	Reference
0	Down 'O'	[RFC6553]
1	Rank-Error (R)	[RFC6553]
2	Forwarding-Error (F)	[RFC6553]
3 (Suggested)	Projected-Route (P)	THIS RFC

Table 24: Initial PDR Flags

#### 11.5. RPL Control Codes

THIS RFC extends the IANA Subregistry created by RFC 6550 for RPL Control Codes as indicated in Table 25:

Code	Description	Reference
0x09 (Suggested)	Projected DAO Request (PDR)	THIS RFC
0x0A (Suggested)	PDR-ACK	THIS RFC

Table 25: New RPL Control Codes

#### 11.6. RPL Control Message Options

THIS RFC extends the IANA Subregistry created by RFC 6550 for RPL Control Message Options as indicated in Table 26:



Value	Meaning	Reference
0x0E (Suggested)	Stateful VIO (SM-VIO)	THIS RFC
0x0F (Suggested)	Source-Routed VIO (NSM-VIO)	THIS RFC
0x10 (Suggested)	Sibling Information option	THIS RFC

Table 26: RPL Control Message Options

### 11.7. SubRegistry for the Projected DAO Request Flags

IANA is required to create a registry for the 8-bit Projected DAO Request (PDR) Flags field. Each bit is Tracked with the following qualities:

- \* Bit number (counting from bit 0 as the most significant bit)
- \* Capability description
- \* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 27:

Bit number	Capability description	Reference
0	PDR-ACK request (K)	THIS RFC
1	Requested path should be redundant (R)	THIS RFC

Table 27: Initial PDR Flags

### 11.8. SubRegistry for the PDR-ACK Flags

IANA is required to create an subregistry for the 8-bit PDR-ACK Flags field. Each bit is Tracked with the following qualities:

- \* Bit number (counting from bit 0 as the most significant bit)
- \* Capability description
- \* Reference

Registration procedure is "Standards Action" [RFC8126]. No bit is currently defined for the PDR-ACK Flags.

#### 11.9. Subregistry for the PDR-ACK Acceptance Status Values

IANA is requested to create a Subregistry for the PDR-ACK Acceptance Status values.

- \* Possible values are 6-bit unsigned integers (0..63).
- \* Registration procedure is "Standards Action" [RFC8126].
- \* Initial allocation is as indicated in Table 28:

Value	Meaning	Reference
0	Unqualified Acceptance	THIS RFC

Table 28: Acceptance values of the PDR-ACK Status

#### 11.10. Subregistry for the PDR-ACK Rejection Status Values

IANA is requested to create a Subregistry for the PDR-ACK Rejection Status values.

- \* Possible values are 6-bit unsigned integers (0..63).
- \* Registration procedure is "Standards Action" [RFC8126].
- \* Initial allocation is as indicated in Table 29:

Value	Meaning	Reference
0	Unqualified Rejection	THIS RFC
1	Transient Failure	THIS RFC

Table 29: Rejection values of the PDR-ACK Status

## 11.11. SubRegistry for the Via Information Options Flags

IANA is requested to create a Subregistry for the 5-bit Via Information Options (Via Information Option) Flags field. Each bit is Tracked with the following qualities:

- \* Bit number (counting from bit 0 as the most significant bit)
- \* Capability description
- \* Reference

Registration procedure is "Standards Action" [RFC8126]. No bit is currently defined for the Via Information Options (Via Information Option) Flags.

## 11.12. SubRegistry for the Sibling Information Option Flags

IANA is required to create a registry for the 5-bit Sibling Information Option (SIO) Flags field. Each bit is Tracked with the following qualities:

- \* Bit number (counting from bit 0 as the most significant bit)
- \* Capability description
- \* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 30:

Bit number	Capability description	Reference
0 (Suggested)	"S" flag: Sibling in same DODAG as Self	THIS RFC

Table 30: Initial SIO Flags

## 11.13. Destination Advertisement Object Flag

THIS RFC modifies the "Destination Advertisement Object (DAO) Flags" registry initially created in Section 20.11 of [RPL] .

Section 4.1.1 also defines one new entry in the Registry as follows:

Bit Number	Capability Description	Reference
2 (Suggested)	Projected DAO (P)	THIS RFC

Table 31: New Destination Advertisement Object (DAO) Flag

#### 11.14. Destination Advertisement Object Acknowledgment Flag

THIS RFC modifies the "Destination Advertisement Object (DAO) Acknowledgment Flags" registry initially created in Section 20.12 of [RPL] .

Section 4.1.2 also defines one new entry in the Registry as follows:

Bit Number	Capability Description	Reference
1 (Suggested)	Projected DAO-ACK (P)	THIS RFC

Table 32: New Destination Advertisement Object Acknowledgment Flag

#### 11.15. New ICMPv6 Error Code

In some cases RPL will return an ICMPv6 error message when a message cannot be forwarded along a P-Route.

IANA has defined an ICMPv6 "Code" Fields Registry for ICMPv6 Message Types. ICMPv6 Message Type 1 describes "destination Unreachable" codes. This specification requires that a new code is allocated from the ICMPv6 Code Fields Registry for ICMPv6 Message Type 1, for "Error in P-Route", with a suggested code value of 8, to be confirmed by IANA.

#### 11.16. RPL Rejection Status values

This specification updates the Subregistry for the "RPL Rejection Status" values under the RPL registry, as follows:

Value	Meaning	Reference
2 (Suggested)	Out of Resources	THIS RFC
3 (Suggested)	Error in VIO	THIS RFC
4 (Suggested)	Predecessor Unreachable	THIS RFC
5 (Suggested)	Unreachable Target	THIS RFC
6..63	Unassigned	

Table 33: Rejection values of the RPL Status

## 12. Acknowledgments

The authors wish to acknowledge JP Vasseur, Remy Liubing, James Pylakutty, and Patrick Wetterwald for their contributions to the ideas developed here. Many thanks to Dominique Barthel and SVR Anand for their global contribution to 6TiSCH, RAW and this RFC, as well as text suggestions that were incorporated. Also special thanks Li Zhao and Toerless Eckert for their in-depth reviews, with many excellent suggestions that improved the readability and well as the content of the specification. Many thanks to Remous-Aris Koutsiamanis for his review during WGLC.

## 13. Normative References

### [INT-ARCHI]

Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.

[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>>.

[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>>.

- [RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.
- [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>>.
- [RPL] 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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

- [RFC9008] Robles, M.I., Richardson, M., and P. Thubert, "Using RPI Option Type, Routing Header for Source Routes, and IPv6-in-IPv6 Encapsulation in the RPL Data Plane", RFC 9008, DOI 10.17487/RFC9008, April 2021, <<https://www.rfc-editor.org/info/rfc9008>>.

#### 14. Informative References

- [6LoWPAN] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [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>>.
- [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>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

- [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>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <<https://www.rfc-editor.org/info/rfc8655>>.
- [RFC8930] Watteyne, T., Ed., Thubert, P., Ed., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multi-Hop IPv6 Network", RFC 8930, DOI 10.17487/RFC8930, November 2020, <<https://www.rfc-editor.org/info/rfc8930>>.
- [RFC8931] Thubert, P., Ed., "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery", RFC 8931, DOI 10.17487/RFC8931, November 2020, <<https://www.rfc-editor.org/info/rfc8931>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.
- [RFC9010] Thubert, P., Ed. and M. Richardson, "Routing for RPL (Routing Protocol for Low-Power and Lossy Networks) Leaves", RFC 9010, DOI 10.17487/RFC9010, April 2021, <<https://www.rfc-editor.org/info/rfc9010>>.
- [RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <<https://www.rfc-editor.org/info/rfc9030>>.
- [RFC9035] Thubert, P., Ed. and L. Zhao, "A Routing Protocol for Low-Power and Lossy Networks (RPL) Destination-Oriented Directed Acyclic Graph (DODAG) Configuration Option for the 6LoWPAN Routing Header", RFC 9035, DOI 10.17487/RFC9035, April 2021, <<https://www.rfc-editor.org/info/rfc9035>>.



## [RAW-ARCHI]

Thubert, P. and G. Z. Papadopoulos, "Reliable and Available Wireless Architecture", Work in Progress, Internet-Draft, draft-ietf-raw-architecture-04, 4 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-raw-architecture-04>>.

## [USE-CASES]

Bernardos, C. J., Papadopoulos, G. Z., Thubert, P., and F. Theoleyre, "RAW use-cases", Work in Progress, Internet-Draft, draft-ietf-raw-use-cases-05, 23 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-raw-use-cases-05>>.

## [I-D.kuehlewind-update-tag]

Kuehlewind, M. and S. Krishnan, "Definition of new tags for relations between RFCs", Work in Progress, Internet-Draft, draft-kuehlewind-update-tag-04, 12 July 2021, <<https://datatracker.ietf.org/doc/html/draft-kuehlewind-update-tag-04>>.

## [I-D.irtf-panrg-path-properties]

Enghardt, T. and C. Krähenbühl, "A Vocabulary of Path Properties", Work in Progress, Internet-Draft, draft-irtf-panrg-path-properties-05, 7 March 2022, <<https://datatracker.ietf.org/doc/html/draft-irtf-panrg-path-properties-05>>.

## [PCE]

IETF, "Path Computation Element", <<https://dataTracker.ietf.org/doc/charter-ietf-pce/>>.

## Authors' Addresses

Pascal Thubert (editor)  
Cisco Systems, Inc  
Building D  
45 Allee des Ormes - BP1200  
06254 Mougins - Sophia Antipolis  
France  
Phone: +33 497 23 26 34  
Email: [pthubert@cisco.com](mailto:pthubert@cisco.com)

Rahul Arvind Jadhav  
Huawei Tech  
Kundalahalli Village, Whitefield,  
Bangalore 560037  
Karnataka  
India  
Phone: +91-080-49160700  
Email: rahul.ietf@gmail.com

Michael C. Richardson  
Sandelman Software Works  
Email: mcr+ietf@sandelman.ca  
URI: <http://www.sandelman.ca/>

ROLL  
Internet-Draft  
Intended status: Standards Track  
Expires: October 17, 2020

R. Jadhav, Ed.  
Huawei  
P. Thubert  
Cisco  
R. Sahoo  
Z. Cao  
Huawei  
April 15, 2020

Efficient Route Invalidation  
draft-ietf-roll-efficient-npdao-18

Abstract

This document explains the problems associated with the current use of NPDAO messaging and also discusses the requirements for an optimized route invalidation messaging scheme. Further a new proactive route invalidation message called as "Destination Cleanup Object" (DCO) is specified which fulfills requirements of an optimized route invalidation messaging.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 17, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language and Terminology . . . . .	3
1.2. Current NPDAO messaging . . . . .	4
1.3. Why Is NPDAO Important? . . . . .	5
2. Problems with current NPDAO messaging . . . . .	6
2.1. Lost NPDAO due to link break to the previous parent . . . . .	6
2.2. Invalidate Routes of Dependent Nodes . . . . .	6
2.3. Possible route downtime caused by asynchronous operation of NPDAO and DAO . . . . .	6
3. Requirements for the NPDAO Optimization . . . . .	6
3.1. Req#1: Remove messaging dependency on link to the previous parent . . . . .	6
3.2. Req#2: Dependent nodes route invalidation on parent switching . . . . .	7
3.3. Req#3: Route invalidation should not impact data traffic . . . . .	7
4. Changes to RPL signaling . . . . .	7
4.1. Change in RPL route invalidation semantics . . . . .	7
4.2. Transit Information Option changes . . . . .	8
4.3. Destination Cleanup Object (DCO) . . . . .	9
4.3.1. Secure DCO . . . . .	10
4.3.2. DCO Options . . . . .	10
4.3.3. Path Sequence number in the DCO . . . . .	11
4.3.4. Destination Cleanup Option Acknowledgment (DCO-ACK) . . . . .	11
4.3.5. Secure DCO-ACK . . . . .	12
4.4. DCO Base Rules . . . . .	12
4.5. Unsolicited DCO . . . . .	13
4.6. Other considerations . . . . .	13
4.6.1. Dependent Nodes invalidation . . . . .	13
4.6.2. NPDAO and DCO in the same network . . . . .	14
4.6.3. Considerations for DCO retry . . . . .	14
4.6.4. DCO with multiple preferred parents . . . . .	15
5. Acknowledgments . . . . .	16
6. IANA Considerations . . . . .	16
6.1. New Registry for the Destination Cleanup Object (DCO) Flags . . . . .	16
6.2. New Registry for the Destination Cleanup Object Acknowledgment (DCO-ACK) Status field . . . . .	17
6.3. New Registry for the Destination Cleanup Object (DCO) Acknowledgment Flags . . . . .	17
7. Security Considerations . . . . .	18

8. Normative References . . . . .	19
Appendix A. Example Messaging . . . . .	20
A.1. Example DCO Messaging . . . . .	20
A.2. Example DCO Messaging with multiple preferred parents . .	21
Authors' Addresses . . . . .	22

## 1. Introduction

RPL [RFC6550] (Routing Protocol for Low power and lossy networks) specifies a proactive distance-vector based routing scheme. RPL has optional messaging in the form of DAO (Destination Advertisement Object) messages, which the 6LBR (6Lo Border Router) and 6LR (6Lo Router) can use to learn a route towards the downstream nodes. In storing mode, DAO messages would result in routing entries being created on all intermediate 6LRs from the node's parent all the way towards the 6LBR.

RPL allows the use of No-Path DAO (NPDAO) messaging to invalidate a routing path corresponding to the given target, thus releasing resources utilized on that path. A NPDAO is a DAO message with route lifetime of zero, originates at the target node and always flows upstream towards the 6LBR. This document explains the problems associated with the current use of NPDAO messaging and also discusses the requirements for an optimized route invalidation messaging scheme. Further a new proactive route invalidation message called as "Destination Cleanup Object" (DCO) is specified which fulfills requirements of an optimized route invalidation messaging.

The document only caters to the RPL's storing mode of operation (MOP). The non-storing MOP does not require use of NPDAO for route invalidation since routing entries are not maintained on 6LRs.

### 1.1. Requirements Language and 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification requires readers to be familiar with all the terms and concepts that are discussed in "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550].

Low Power and Lossy Networks (LLN):

Network in which both the routers and their interconnect are constrained. LLN routers typically operate with constraints on processing power, memory, and energy (battery power). Their

interconnects are characterized by high loss rates, low data rates, and instability.

**6LoWPAN Router (6LR):**

An intermediate router that is able to send and receive Router Advertisements (RAs) and Router Solicitations (RSs) as well as forward and route IPv6 packets.

**Directed Acyclic Graph (DAG):**

A directed graph having the property that all edges are oriented in such a way that no cycles exist.

**Destination-Oriented DAG (DODAG):**

A DAG rooted at a single destination, i.e., at a single DAG root with no outgoing edges.

**6LoWPAN Border Router (6LBR):**

A border router which is a DODAG root and is the edge node for traffic flowing in and out of the 6LoWPAN network.

**Destination Advertisement Object (DAO):**

DAO messaging allows downstream routes to the nodes to be established.

**DODAG Information Object (DIO):**

DIO messaging allows upstream routes to the 6LBR to be established. DIO messaging is initiated at the DAO root.

**Common Ancestor node**

6LR/6LBR node which is the first common node between two paths of a target node.

**No-Path DAO (NPDAO):**

A DAO message which has target with lifetime 0 used for the purpose of route invalidation.

**Destination Cleanup Object (DCO):**

A new RPL control message code defined by this document. DCO messaging improves proactive route invalidation in RPL.

**Regular DAO:**

A DAO message with non-zero lifetime. Routing adjacencies are created or updated based on this message.

**Target node:**

The node switching its parent whose routing adjacencies are updated (created/removed).

## 1.2. Current NPDAO messaging

RPL uses NPDAO messaging in the storing mode so that the node changing its routing adjacencies can invalidate the previous route. This is needed so that nodes along the previous path can release any resources (such as the routing entry) they maintain on behalf of target node.

For the rest of this document consider the following topology:

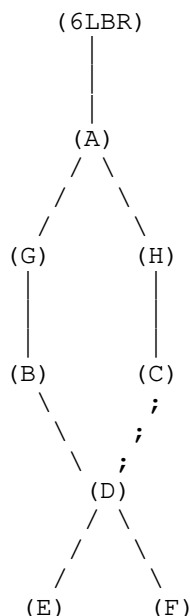


Figure 1: Sample topology

Node (D) is connected via preferred parent (B). (D) has an alternate path via (C) towards the 6LBR. Node (A) is the common ancestor for (D) for paths through (B)-(G) and (C)-(H). When (D) switches from (B) to (C), RPL allows sending NPDAO to (B) and regular DAO to (C).

### 1.3. Why Is NPDAO Important?

Nodes in LLNs may be resource constrained. There is limited memory available and routing entry records are 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 optimize resource utilization. Thus it becomes necessary to have an 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 updates needs to be done for the routing tables entries of (C), (H), (A), (G), and (B) with destination (D), (E) and (F). Without efficient route invalidation, a 6LR may have to hold a lot of stale route entries.

## 2. Problems with current NPDAO messaging

### 2.1. Lost NPDAO due to link break to the previous parent

When a node switches its parent, the NPDAO is to be sent to its previous parent and a regular DAO to 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.

### 2.2. Invalidate Routes of Dependent Nodes

RPL does not specify how route invalidation will work for dependent nodes rooted at the switching node, resulting in stale routing entries of the dependent nodes. 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. There is no NPDAO generated by the dependent child nodes (E) and (F), through the previous path via (D) to (B) and (G), resulting in stale entries on nodes (B) and (G) for nodes (E) and (F).

### 2.3. Possible 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 impacting downward traffic for the switching node.

In the example topology, consider Node (D) switches from parent (B) to (C). An NPDAO sent via the previous route may invalidate the previous route whereas there is no way to determine whether the new DAO has successfully updated the route entries on the new path.

## 3. Requirements for the NPDAO Optimization

### 3.1. Req#1: Remove messaging dependency on link to the previous parent

When the switching node sends the NPDAO message to the previous parent, it is normal that the link to the previous parent is prone to failure (that's why the node decided to switch). Therefore, it is required that the route invalidation does not depend on the previous link which is prone to failure. The previous link referred here represents the link between the node and its previous parent (from whom the node is now disassociating).



### 3.2. Req#2: Dependent nodes route invalidation on parent switching

It should be possible to do route invalidation for dependent nodes rooted at the switching node.

### 3.3. Req#3: Route invalidation should not impact data traffic

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

## 4. Changes to RPL signaling

### 4.1. Change in RPL route invalidation semantics

As described in Section 1.2, the NPDAO originates at the node changing to a new parent and traverses upstream towards the root. In order to solve the problems as mentioned in Section 2, the document adds a new proactive route invalidation message called "Destination Cleanup Object" (DCO) that originates at a common ancestor node and flows downstream between the new and old path. The common ancestor node generates a DCO in response to the change in the next-hop on receiving a regular DAO with updated Path Sequence for the target.

The 6LRs in the path for DCO take action such as route invalidation based on the DCO information and subsequently send another DCO with the same information downstream to the next hop. This operation is similar to how the DAOs are handled on intermediate 6LRs in storing MOP in [RFC6550]. Just like DAO in storing MOP, the DCO is sent using link-local unicast source and destination IPv6 address. Unlike DAO, which always travels upstream, the DCO always travels downstream.

In Figure 1, when node D decides to switch the path from B to C, it sends a regular DAO to node C with reachability information containing the address of D as the target and an incremented Path Sequence. Node C will update the routing table based on the reachability information in the DAO and in turn generate another DAO with the same reachability information and forward it to H. Node H also follows the same procedure as Node C and forwards it to node A. When node A receives the regular DAO, it finds that it already has a routing table entry on behalf of the target address of node D. It finds however that the next hop information for reaching node D has changed i.e., node D has decided to change the paths. In this case, Node A which is the common ancestor node for node D along the two

paths (previous and new), should generate a DCO which traverses downwards in the network. Node A handles normal DAO forwarding to 6LBR as required by [RFC6550].

#### 4.2. Transit Information Option changes

Every RPL message is divided into base message fields and additional Options as described in Section 6 of [RFC6550]. The base fields apply to the message as a whole and options are appended to add message/use-case specific attributes. As an example, a DAO message may be attributed by one or more "RPL Target" options which specify the reachability information for the given targets. Similarly, a Transit Information option may be associated with a set of RPL Target options.

This document specifies a change in the Transit Information Option to contain the "Invalidate previous route" (I) flag. This 'I' flag signals the common ancestor node to generate a DCO on behalf of the target node with a RPL Status of 195 indicating that the address has moved. The 'I' flag is carried in the Transit Information Option which augments the reachability information for a given set of RPL Target(s). Transit Information Option with 'I' flag set should be carried in the DAO message when route invalidation is sought for the corresponding target(s).

Value 195 represents 'E' and 'A' bit in RPL Status to be set as per Figure 3 of [I-D.ietf-roll-unaware-leaves] with the lower 6 bits with value 3 indicating 'Moved' as per Table 1 of [RFC8505].

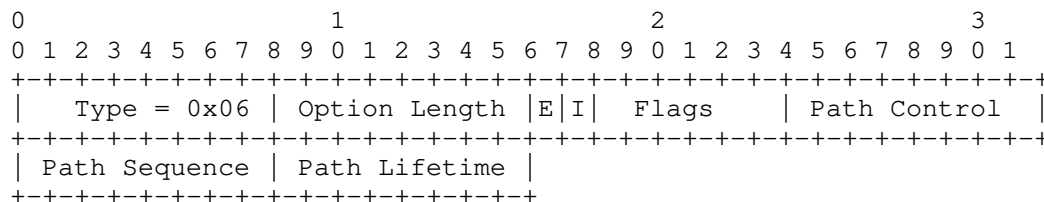


Figure 2: Updated Transit Information Option (New I flag added)

I (Invalidate previous route) flag: The 'I' flag is set by the target node to indicate to the common ancestor node that it wishes to invalidate any previous route between the two paths.

[RFC6550] allows the parent address to be sent in the Transit Information Option depending on the mode of operation. In case of storing mode of operation the field is usually not needed. In case of DCO, the parent address field MUST NOT be included.



D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: The 6 bits remaining unused in the Flags field are reserved for future use. These bits MUST be initialized to zero by the sender and MUST be ignored by the receiver.

RPL Status: As defined in [RFC6550] and updated in [I-D.ietf-roll-unaware-leaves]. The root or common parent that generates a DCO is authoritative for setting the status information and the information is unchanged as propagated down the DODAG. This document does not specify a differentiated action based on the RPL status.

DCOSequence: 8-bit field incremented at each unique DCO message from a node and echoed in the DCO-ACK message. The initial DCOSequence can be chosen randomly by the node. Section 4.4 explains the handling of the DCOSequence.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present if 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

#### 4.3.1. Secure DCO

A Secure DCO message follows the format in [RFC6550] Figure 7, where the base message format is the DCO message shown in Figure 3.

#### 4.3.2. DCO Options

The DCO message MUST carry at least one RPL Target and the Transit Information Option and MAY carry other valid options. This specification allows for the DCO message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x06 Transit Information
- 0x09 RPL Target Descriptor

Section 6.7 of [RFC6550] defines all the above mentioned options. The DCO carries an RPL Target Option and an associated Transit Information Option with a lifetime of 0x00000000 to indicate a loss of reachability to that Target.

#### 4.3.3. Path Sequence number in the DCO

A DCO message may contain a Path Sequence in the Transit Information Option to identify the freshness of the DCO message. The Path Sequence in the DCO MUST use the same Path Sequence number present in the regular DAO message when the DCO is generated in response to a DAO message. Thus if a DCO is received by a 6LR and subsequently a DAO is received with an old sequence number, then the DAO MUST be ignored. When the DCO is generated in response to a DCO from upstream parent, the Path Sequence MUST be copied from the received DCO.

#### 4.3.4. Destination Cleanup Option Acknowledgment (DCO-ACK)

The DCO-ACK message SHOULD be sent as a unicast packet by a DCO recipient in response to a unicast DCO message with 'K' flag set. If 'K' flag is not set then the receiver of the DCO message MAY send a DCO-ACK, especially to report an error condition.

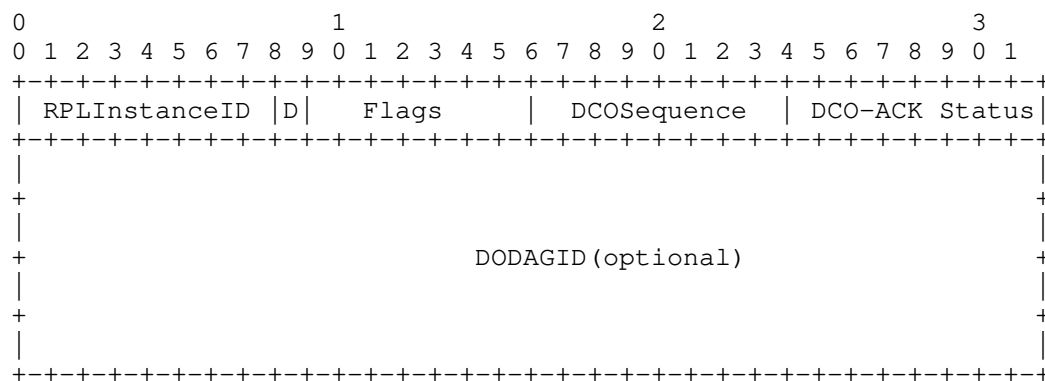


Figure 4: DCO-ACK base object

**RPLInstanceID:** 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

**D:** The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

**Flags:** 7-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

**DCOSequence:** 8-bit field. The DCOSequence in DCO-ACK is copied from the DCOSequence received in the DCO message.

DCO-ACK Status: Indicates the completion. A value of 0 is defined as unqualified acceptance in this specification. A value of 1 is defined as "No routing-entry for the Target found". The remaining status values are reserved as rejection codes.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present when 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

#### 4.3.5. Secure DCO-ACK

A Secure DCO-ACK message follows the format in [RFC6550] Figure 7, where the base message format is the DCO-ACK message shown in Figure 4.

#### 4.4. DCO Base Rules

1. If a node sends a DCO message with newer or different information than the prior DCO message transmission, it MUST increment the DCOSequence field by at least one. A DCO message transmission that is identical to the prior DCO message transmission MAY increment the DCOSequence field. The DCOSequence counter follows the sequence counter operation as defined in Section 7.2 of [RFC6550].
2. The RPLInstanceID and DODAGID fields of a DCO message MUST be the same value as that of the DAO message in response to which the DCO is generated on the common ancestor node.
3. A node MAY set the 'K' flag in a unicast DCO message to solicit a unicast DCO-ACK in response in order to confirm the attempt.
4. A node receiving a unicast DCO message with the 'K' flag set SHOULD respond with a DCO-ACK. A node receiving a DCO message without the 'K' flag set MAY respond with a DCO-ACK, especially to report an error condition.
5. A node receiving a unicast DCO message MUST verify the stored Path Sequence in context to the given target. If the stored Path Sequence is more fresh, newer than the Path Sequence received in the DCO, then the DCO MUST be dropped.
6. A node that sets the 'K' flag in a unicast DCO message but does not receive DCO-ACK in response MAY reschedule the DCO message transmission for another attempt, up until an implementation specific number of retries.
7. A node receiving a unicast DCO message with its own address in the RPL Target Option MUST strip-off that Target Option. If this Target Option is the only one in the DCO message then the DCO message MUST be dropped.

The scope of DCOSequence values is unique to the node which generates it.

#### 4.5. Unsolicited DCO

A 6LR may generate an unsolicited DCO to unilaterally cleanup the path on behalf of the target entry. The 6LR has all the state information, namely, the Target address and the Path Sequence, required for generating DCO in its routing table. The conditions why 6LR may generate an unsolicited DCO are beyond the scope of this document but some possible reasons could be:

1. On route expiry of an entry, a 6LR may decide to graciously cleanup the entry by initiating DCO.
2. 6LR needs to entertain higher priority entries in case the routing table is full, thus resulting in eviction of an existing routing entry. In this case the eviction can be handled graciously using DCO.

Note that if the 6LR initiates a unilateral path cleanup using DCO and if it has the latest state for the target then the DCO would finally reach the target node. Thus the target node would be informed of its invalidation.

#### 4.6. Other considerations

##### 4.6.1. Dependent Nodes invalidation

Current RPL [RFC6550] does not provide a mechanism for route invalidation for dependent nodes. This document allows the dependent nodes invalidation. Dependent nodes will generate their respective DAOs to update their paths, and the previous route invalidation for those nodes should work in the similar manner described for switching node. The dependent node may set the 'I' flag in the Transit Information Option as part of regular DAO so as to request invalidation of previous route from the common ancestor node.

Dependent nodes do not have any indication regarding if any of their parents in turn have decided to switch their parent. Thus for route invalidation the dependent nodes may choose to always set the 'I' flag in all its DAO message's Transit Information Option. Note that setting the 'I' flag is not counterproductive even if there is no previous route to be invalidated.

#### 4.6.2. NPDAO and DCO in the same network

The current NPDAO mechanism in [RFC6550] can still be used in the same network where DCO is used. The NPDAO messaging can be used, for example, on route lifetime expiry of the target or when the node simply decides to gracefully terminate the RPL session on graceful node shutdown. Moreover, a deployment can have a mix of nodes supporting the DCO and the existing NPDAO mechanism. It is also possible that the same node supports both the NPDAO and DCO signaling for route invalidation.

Section 9.8 of [RFC6550] states, "When a node removes a node from its DAO parent set, it SHOULD send a No-Path DAO message to that removed DAO parent to invalidate the existing router". This document introduces an alternative and more optimized way of route invalidation but it also allows existing NPDAO messaging to work. Thus an implementation has two choices to make when a route invalidation is to be initiated:

1. Use NPDAO to invalidate the previous route and send regular DAO on the new path.
2. Send regular DAO on the new path with the 'I' flag set in the Transit Information Option such that the common ancestor node initiates the DCO message downstream to invalidate the previous route.

This document recommends using option 2 for reasons specified in Section 3 in this document.

This document assumes that all the 6LRs in the network support this specification. If there are 6LRs en-route DCO message path which do not support this document, then the route invalidation for corresponding targets may not work or may work partially i.e., only part of the path supporting DCO may be invalidated. Alternatively, a node could generate an NPDAO if it does not receive a DCO with itself as target within specified time limit. The specified time limit is deployment specific and depends upon the maximum depth of the network and per hop average latency. Note that sending NPDAO and DCO for the same operation would not result in unwanted side-effects because the acceptability of NPDAO or DCO depends upon the Path Sequence freshness.

#### 4.6.3. Considerations for DCO retry

A DCO message could be retried by a sender if it sets the 'K' flag and does not receive a DCO-ACK. The DCO retry time could be dependent on the maximum depth of the network and average per hop latency. This could range from 2 seconds to 120 seconds depending on



the deployment. In case the latency limits are not known, an implementation MUST NOT retry more than once in 3 seconds and MUST NOT retry more than 3 times.

The number of retries could also be set depending on how critical the route invalidation could be for the deployment and the link layer retry configuration. For networks supporting only MP2P and P2MP flows, such as in AMI and telemetry applications, the 6LRs may not be very keen to invalidate routes, unless they are highly memory-constrained. For home and building automation networks which may have substantial P2P traffic, the 6LRs might be keen to invalidate efficiently because it may additionally impact the forwarding efficiency.

#### 4.6.4. DCO with multiple preferred parents

[RFC6550] allows a node to select multiple preferred parents for route establishment. Section 9.2.1 of [RFC6550] specifies, "All DAOs generated at the same time for the same Target MUST be sent with the same Path Sequence in the Transit Information". Subsequently when route invalidation has to be initiated, RPL mentions use of NPDAO which can be initiated with an updated Path Sequence to all the parent nodes through which the route is to be invalidated.

With DCO, the Target node itself does not initiate the route invalidation and it is left to the common ancestor node. A common ancestor node when it discovers an updated DAO from a new next-hop, it initiates a DCO. With multiple preferred parents, this handling does not change. But in this case it is recommended that an implementation initiates a DCO after a time period (DelayDCO) such that the common ancestor node may receive updated DAOs from all possible next-hops. This will help to reduce DCO control overhead i.e., the common ancestor can wait for updated DAOs from all possible directions before initiating a DCO for route invalidation. After timeout, the DCO needs to be generated for all the next-hops for whom the route invalidation needs to be done.

This document recommends using a DelayDCO timer value of 1sec. This value is inspired by the default DelayDAO value of 1sec in [RFC6550]. Here the hypothesis is that the DAOs from all possible parent sets would be received on the common ancestor within this time period.

It is still possible that a DCO is generated before all the updated DAOs from all the paths are received. In this case, the ancestor node would start the invalidation procedure for paths from which the updated DAO is not received. The DCO generated in this case would start invalidating the segments along these paths on which the updated DAOs are not received. But once the DAO reaches these

segments, the routing state would be updated along these segments and should not lead to any inconsistent routing state.

Note that there is no requirement for synchronization between DCO and DAOs. The DelayDCO timer simply ensures that the DCO control overhead can be reduced and is only needed when the network contains nodes using multiple preferred parent.

## 5. Acknowledgments

Many thanks to Alvaro Retana, Cenk Gundogan, Simon Duquennoy, Georgios Papadopoulos, Peter Van Der Stok for their review and comments. Alvaro Retana helped shape this document's final version with critical review comments.

## 6. IANA Considerations

IANA is requested to allocate new codes for the DCO and DCO-ACK messages from the RPL Control Codes registry.

Code	Description	Reference
TBD1	Destination Cleanup Object	This document
TBD2	Destination Cleanup Object Acknowledgment	This document
TBD3	Secure Destination Cleanup Object	This document
TBD4	Secure Destination Cleanup Object Acknowledgment	This document

IANA is requested to allocate bit 1 from the Transit Information Option Flags registry for the 'I' flag (Section 4.2)

### 6.1. New Registry for the Destination Cleanup Object (DCO) Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Flags field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description

- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DCO-ACK request (K)	This document
1	DODAGID field is present (D)	This document

#### DCO Base Flags

### 6.2. New Registry for the Destination Cleanup Object Acknowledgment (DCO-ACK) Status field

IANA is requested to create a registry for the 8-bit Destination Cleanup Object Acknowledgment (DCO-ACK) Status field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New Status values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- o Status Code
- o Description
- o Defining RFC

The following values are currently defined:

Status Code	Description	Reference
0	Unqualified acceptance	This document
1	No routing-entry for the indicated Target found	This document

#### DCO-ACK Status Codes

### 6.3. New Registry for the Destination Cleanup Object (DCO) Acknowledgment Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Acknowledgment Flags field. This registry

should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description
- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DODAGID field is present (D)	This document

#### DCO-ACK Base Flags

## 7. Security Considerations

This document introduces the ability for a common ancestor node to invalidate a route on behalf of the target node. The common ancestor node could be directed to do so by the target node using the 'I' flag in DCO's Transit Information Option. However, the common ancestor node is in a position to unilaterally initiate the route invalidation since it possesses all the required state information, namely, the Target address and the corresponding Path Sequence. Thus a rogue common ancestor node could initiate such an invalidation and impact the traffic to the target node.

The DCO carries a RPL Status value, which is informative. New Status values may be created over time and a node will ignore an unknown Status value. This enables RPL Status field to be used as a cover channel. But the channel only works once since the message destroys its own medium, that is the existing route that it is removing.

This document also introduces an 'I' flag which is set by the target node and used by the ancestor node to initiate a DCO if the ancestor sees an update in the route adjacency. However, this flag could be spoofed by a malicious 6LR in the path and can cause invalidation of an existing active path. Note that invalidation will happen only if the other conditions such as Path Sequence condition is also met. Having said that, such a malicious 6LR may spoof a DAO on behalf of the (sub) child with the 'I' flag set and can cause route invalidation on behalf of the (sub) child node. Note that, using existing mechanisms offered by [RFC6550], a malicious 6LR might also

spoof a DAO with lifetime of zero or otherwise cause denial of service by dropping traffic entirely, so the new mechanism described in this document does not present a substantially increased risk of disruption.

This document assumes that the security mechanisms as defined in [RFC6550] are followed, which means that the common ancestor node and all the 6LRs are part of the RPL network because they have the required credentials. A non-secure RPL network needs to take into consideration the risks highlighted in this section as well as those highlighted in [RFC6550].

All RPL messages support a secure version of messages which allows integrity protection using either a MAC or a signature. Optionally, secured RPL messages also have encryption protection for confidentiality.

The document adds new messages (DCO, DCO-ACK) which are syntactically similar to existing RPL messages such as DAO, DAO-ACK. Secure versions of DCO and DCO-ACK are added similar to other RPL messages (such as DAO, DAO-ACK).

RPL supports three security modes as mentioned in Section 10.1 of [RFC6550]:

1. Unsecured: In this mode, it is expected that the RPL control messages are secured by other security mechanisms, such as link-layer security. In this mode, the RPL control messages, including DCO, DCO-ACK, do not have Security sections. Also note that unsecured mode does not imply that all messages are sent without any protection.
2. Preinstalled: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.
3. Authenticated: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.

## 8. Normative References

[I-D.ietf-roll-unaware-leaves]

Thubert, P. and M. Richardson, "Routing for RPL Leaves", draft-ietf-roll-unaware-leaves-14 (work in progress), April 2020.

[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>>.

- [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>>.
- [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>>.

## Appendix A. Example Messaging

### A.1. Example DCO Messaging

In Figure 1, node (D) switches its parent from (B) to (C). This example assumes that Node D has already established its own route via Node B-G-A-6LBR using pathseq=x. The example uses DAO and DCO messaging convention and specifies only the required parameters to explain the example namely, the parameter 'tgt', which stands for Target Option and value of this parameter specifies the address of the target node. The parameter 'pathseq', which specifies the Path Sequence value carried in the Transit Information Option. The parameter 'I\_flag' specifies the 'I' flag in the Transit Information Option. sequence of actions is as follows:

1. Node D switches its parent from node B to node C
2. D sends a regular DAO(tgt=D,pathseq=x+1,I\_flag=1) in the updated path to C
3. C checks for a routing entry on behalf of D, since it cannot find an entry on behalf of D it creates a new routing entry and forwards the reachability information of the target D to H in a DAO(tgt=D,pathseq=x+1,I\_flag=1).
4. Similar to C, node H checks for a routing entry on behalf of D, cannot find an entry and hence creates a new routing entry and forwards the reachability information of the target D to A in a DAO(tgt=D,pathseq=x+1,I\_flag=1).
5. Node A receives the DAO(tgt=D,pathseq=x+1,I\_flag=1), and checks for a routing entry on behalf of D. It finds a routing entry but checks that the next hop for target D is different (i.e., Node G). Node A checks the I\_flag and generates DCO(tgt=D,pathseq=x+1) to previous next hop for target D which is G. Subsequently, Node A updates the routing entry and forwards the reachability information of target D upstream DAO(tgt=D,pathseq=x+1,I\_flag=1).
6. Node G receives the DCO(tgt=D,pathseq=x+1). It checks if the received path sequence is later than the stored path sequence. If it is later, Node G invalidates the routing entry of target D

and forwards the (un)reachability information downstream to B in DCO(tgt=D,pathseq=x+1).

7. Similarly, B processes the DCO(tgt=D,pathseq=x+1) by invalidating the routing entry of target D and forwards the (un)reachability information downstream to D.
8. D ignores the DCO(tgt=D,pathseq=x+1) since the target is itself.
9. The propagation of the DCO will stop at any node where the node does not have an routing information associated with the target. If cached routing information is present and the cached Path Sequence is higher than the value in the DCO, then the DCO is dropped.

#### A.2. Example DCO Messaging with multiple preferred parents

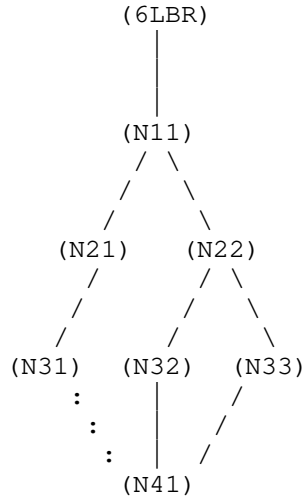


Figure 5: Sample topology 2

In Figure 5, node (N41) selects multiple preferred parents (N32) and (N33). The sequence of actions is as follows:

1. (N41) sends DAO(tgt=N41,PS=x,I\_flag=1) to (N32) and (N33). Here I\_flag refers to the Invalidation flag and PS refers to Path Sequence in Transit Information option.
2. (N32) sends DAO(tgt=N41,PS=x,I\_flag=1) to (N22). (N33) also sends DAO(tgt=N41,PS=x,I\_flag=1) to (N22). (N22) learns multiple routes for the same destination (N41) through multiple next-hops. (N22) may receive the DAOs from (N32) and (N33) in any order with the I\_flag set. The implementation should use the DelayDCO timer to wait to initiate the DCO. If (N22) receives an updated DAO from all the paths then the DCO need not

- be initiated in this case. Thus the route table at N22 should contain (Dst,NextHop,PS): { (N41,N32,x), (N41,N33,x) }.
3. (N22) sends DAO(tgt=N41,PS=x,I\_flag=1) to (N11).
  4. (N11) sends DAO(tgt=N41,PS=x,I\_flag=1) to (6LBR). Thus the complete path is established.
  5. (N41) decides to change preferred parent set from { N32, N33 } to { N31, N32 }.
  6. (N41) sends DAO(tgt=N41,PS=x+1,I\_flag=1) to (N32). (N41) sends DAO(tgt=N41,PS=x+1,I\_flag=1) to (N31).
  7. (N32) sends DAO(tgt=N41,PS=x+1,I\_flag=1) to (N22). (N22) has multiple routes to destination (N41). It sees that a new Path Sequence for Target=N41 is received and thus it waits for pre-determined time period (DelayDCO time period) to invalidate another route {(N41),(N33),x}. After time period, (N22) sends DCO(tgt=N41,PS=x+1) to (N33). Also (N22) sends the regular DAO(tgt=N41,PS=x+1,I\_flag=1) to (N11).
  8. (N33) receives DCO(tgt=N41,PS=x+1). The received Path Sequence is latest and thus it invalidates the entry associated with target (N41). (N33) then sends the DCO(tgt=N41,PS=x+1) to (N41). (N41) sees itself as the target and drops the DCO.
  9. From Step 6 above, (N31) receives the DAO(tgt=N41,PS=x+1,I\_flag=1). It creates a routing entry and sends the DAO(tgt=N41,PS=x+1,I\_flag=1) to (N21). Similarly (N21) receives the DAO and subsequently sends the DAO(tgt=N41,PS=x+1,I\_flag=1) to (N11).
  10. (N11) receives DAO(tgt=N41,PS=x+1,I\_flag=1) from (N21). It waits for DelayDCO timer since it has multiple routes to (N41). (N41) will receive DAO(tgt=N41,PS=x+1,I\_flag=1) from (N22) from Step 7 above. Thus (N11) has received regular DAO(tgt=N41,PS=x+1,I\_flag=1) from all paths and thus does not initiate DCO.
  11. (N11) forwards the DAO(tgt=N41,PS=x+1,I\_flag=1) to 6LBR and the full path is established.

#### Authors' Addresses

Rahul Arvind Jadhav (editor)  
Huawei  
Kundalahalli Village, Whitefield,  
Bangalore, Karnataka 560037  
India

Phone: +91-080-49160700  
Email: rahul.ietf@gmail.com



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

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

Rabi Narayan Sahoo  
Huawei  
Kundalahalli Village, Whitefield,  
Bangalore, Karnataka 560037  
India

Phone: +91-080-49160700  
Email: rabinarayans@huawei.com

Zhen Cao  
Huawei  
W Chang'an Ave  
Beijing  
P.R. China

Email: zhencao.ietf@gmail.com

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

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

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

## Abstract

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

## Status of This Memo

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

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

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

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

## Copyright Notice

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

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

## Table of Contents

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

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

## 1. Introduction

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

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

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

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

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

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

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

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

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

### 1.1. Overview

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

## 2. Terminology and Requirements Language

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3. RPL Overview

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

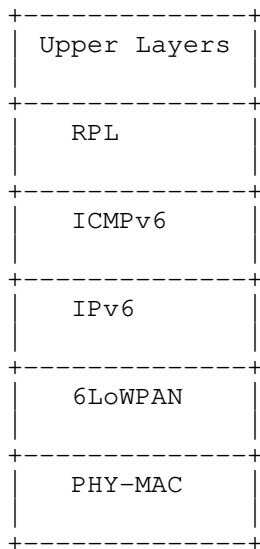


Figure 1: RPL Stack.

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

#### 4. Updates to RFC6550, RFC6553 and RFC8138

##### 4.1. Updates to RFC6550

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

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



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

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

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

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

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

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

#### 4.1.2. Configuration Options and Mode of Operation

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

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

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

See Sections 11.2 and 11.3.

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

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

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

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

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

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

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

Bit number	Description	Reference
3	RPI 0x23 enable	This document

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

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

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

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

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

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

Figure 3: Option Type in RPL Option.

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

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

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

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

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

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

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

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

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

This is a significant update to [RFC6553].

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

Figure 5: RPI Inserted by the Root in Storing Mode

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

## 5. Sample/reference topology

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

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

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

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

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

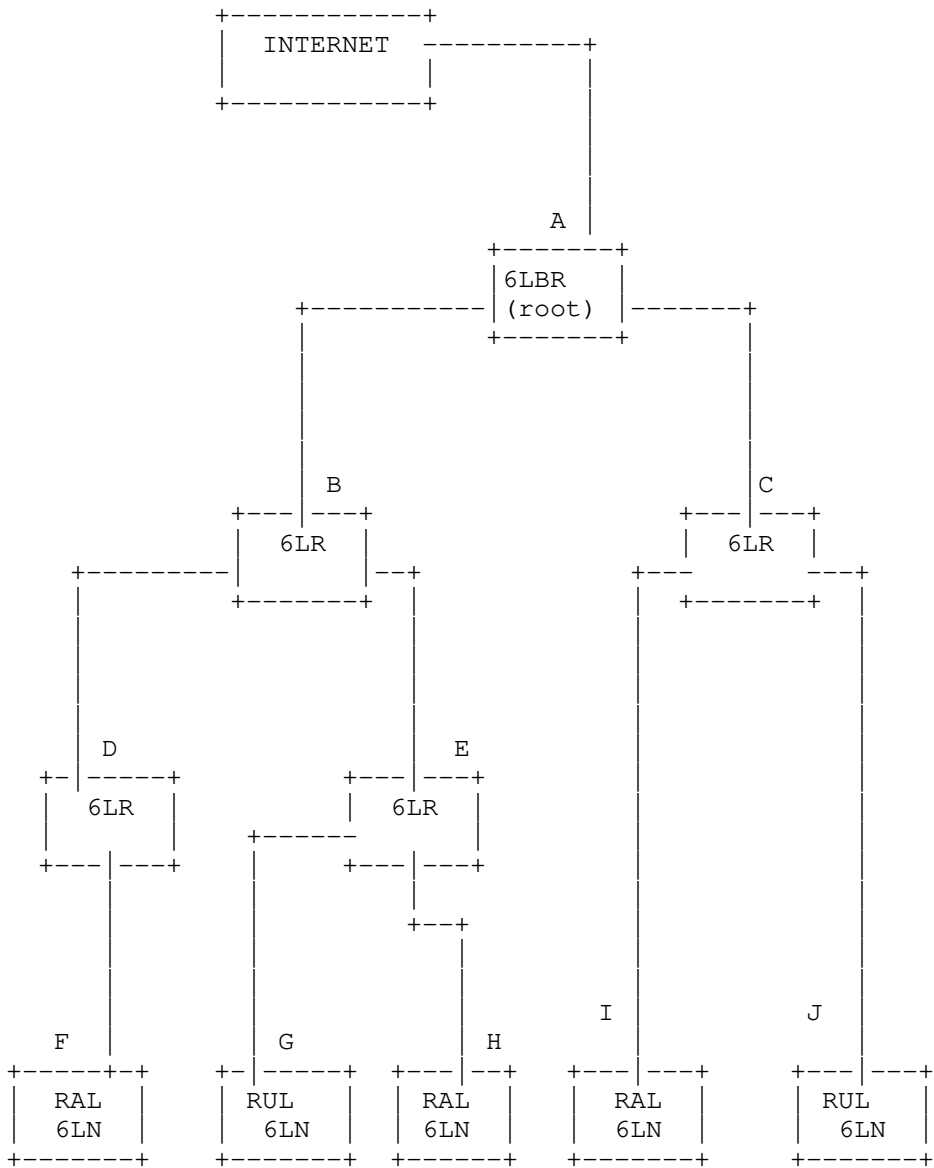


Figure 6: A reference RPL Topology.



## 6. Use cases

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

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

The use cases are as follows:

Interaction between Leaf and Root:

RAL to root

root to RAL

RUL to root

root to RUL

Interaction between Leaf and Internet:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

Interaction between leaves:

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

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

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

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

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

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

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

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

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

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

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

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

The uses cases are delineated based on the following assumptions:

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

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

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

## 7. Storing mode

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

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

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

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

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

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

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

### 7.1. Storing Mode: Interaction between Leaf and Root

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

RAL to root

root to RAL

RUL to root

root to RUL

### 7.1.1. SM: Example of Flow from RAL to Root

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

In this case the flow comprises:

RAL (6LN) --> 6LR\_i --> root(6LBR)

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

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

No IPv6-in-IPv6 header is required.

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

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

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

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

### 7.1.2. SM: Example of Flow from Root to RAL

In this case the flow comprises:

root (6LBR) --> 6LR\_i --> RAL (6LN)

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

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

No IPv6-in-IPv6 header is required.

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

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

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

### 7.1.3. SM: Example of Flow from Root to RUL

In this case the flow comprises:

root (6LBR) --> 6LR\_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Node A (6LBR) --> Node B (6LR\_i) --> Node E (6LR\_n) --> Node G (RUL)

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

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

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

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

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

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

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

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



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

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

#### 7.1.4. SM: Example of Flow from RUL to Root

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR<sub>1</sub> --> 6LR<sub>i</sub> --> root (6LBR)

For example, a communication flow could be: Node G (RUL) --> Node E (6LR<sub>1</sub>) --> Node B (6LR<sub>i</sub>) --> Node A root (6LBR)

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

When the packet arrives from the RUL (Node G) to 6LR<sub>1</sub> (Node E), the 6LR<sub>1</sub> will encapsulate the packet in an IPv6-in-IPv6 header with an RPI. The IPv6-in-IPv6 header is addressed to the root (Node A). The root removes the header and processes the packet.

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

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

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

## 7.2. SM: Interaction between Leaf and Internet.

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

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

### 7.2.1. SM: Example of Flow from RAL to Internet

In this case the flow comprises:

RAL (6LN) --> 6LR\_i --> root (6LBR) --> Internet

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

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

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

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

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

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

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

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

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

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

#### 7.2.2. SM: Example of Flow from Internet to RAL

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR\_i --> RAL (6LN)

For example, a communication flow could be: Internet --> Node A  
root(6LBR) --> Node B (6LR\_1) --> Node D (6LR\_n) --> Node F (RAL)

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

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

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

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

### 7.2.3. SM: Example of Flow from RUL to Internet

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR\_1 --> 6LR\_i --> root (6LBR) --> Internet

For example, a communication flow could be: Node G (RUL) --> Node E (6LR\_1) --> Node B (6LR\_i) --> Node A root (6LBR) --> Internet

The node 6LR\_1 (i=1) will add an IPv6-in-IPv6(RPI) header addressed to the root such that the root can remove the RPI before passing upwards. In the intermediate 6LR, the rank in the RPI is modified.

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

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

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

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

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

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR\_i --> RUL (IPv6 dst node)

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

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

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

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

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

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

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

### 7.3. SM: Interaction between Leaf and Leaf

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

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

#### 7.3.1. SM: Example of Flow from RAL to RAL

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

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

RAL src (6LN) --> 6LR\_ia --> common parent (6LR\_x) --> 6LR\_id --> RAL dst (6LN)

For example, a communication flow could be: Node F (RAL src) --> Node D (6LR\_ia) --> Node B (6LR\_x) --> Node E (6LR\_id) --> Node H (RAL dst)

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

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

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

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

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

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

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

### 7.3.2. SM: Example of Flow from RAL to RUL

In this case the flow comprises:



RAL src (6LN) --> 6LR\_ia --> common parent (6LBR - The root-) -->  
6LR\_id --> RUL (IPv6 dst node)

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

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

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

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

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

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

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

### 7.3.3. SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR\_ia --> 6LBR --> 6LR\_id --> RAL dst (6LN)

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

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

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

The 6LR\_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI1) encapsulated in a IPv6-in-IPv6 header to the root. The root removes the outer header including the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the destination RAL (Node F).

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

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

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

## 7.3.4. SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node)--> 6LR\_1--> 6LR\_ia --> 6LBR --> 6LR\_id --> RUL  
(IPv6 dst node)

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

Internal nodes 6LR\_ia (e.g: Node E or Node B) is the intermediate router from the RUL source (Node G) to the root (6LBR) (Node A). In this case,  $1 \leq ia \leq n$ , where  $n$  is the total number of routers (6LR) that the packet goes through from the RUL to the root. 6LR\_1 refers when  $ia=1$ .

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

The 6LR\_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI), encapsulated in an IPv6-in-IPv6 header directed to the root. The root removes the outer header including

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

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

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

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

## 8. Non Storing mode

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

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

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

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

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

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

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

RAL to root

root to RAL

RUL to root

root to RUL

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

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

RAL (6LN) --> 6LR\_i --> root(6LBR)

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

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

This situation is the same case as storing mode.

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

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

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

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

In this case the flow comprises:

root (6LBR) --> 6LR\_i --> RAL (6LN)

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

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

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

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

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

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

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

In this case the flow comprises:

root (6LBR) --> 6LR\_i --> RUL (IPv6 dst node)

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

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

In the 6LBR, the RH3 is added; it is then modified at each intermediate 6LR (6LR\_1 and so on), and it is fully consumed in the last 6LR (6LR\_n) but is left in place. When the RPI is added, the RUL, which does not understand the RPI, will ignore it (per [RFC8200]); thus, encapsulation is not necessary.

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



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

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

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

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR<sub>1</sub> --> 6LR<sub>i</sub> --> root (6LBR) dst

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

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

In this case, the RPI is added by the first 6LR (6LR<sub>1</sub>) (Node E), encapsulated in an IPv6-in-IPv6 header, and modified in the subsequent 6LRs in the flow. The RPI and the entire packet are consumed by the root.

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

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

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

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

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

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

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

In this case the flow comprises:

RAL (6LN) src --> 6LR\_i --> root (6LBR) --> Internet dst

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

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

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

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

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

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

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

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

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

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

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR\_i --> RAL dst (6LN)

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

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

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

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

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

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

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

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR\_1 --> 6LR\_i --> root (6LBR) --> Internet  
dst

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

6LR\_i represents the intermediate routers from source to destination,  
 $1 \leq i \leq n$ , where  $n$  is the total number of routers (6LRs) that the  
 packet goes through from the source (RUL) to the 6LBR, e.g., 6LR\_1  
 ( $i=1$ ).

In this case the flow label is recommended to be zero in the RUL. As  
 the RUL parent adds RPL headers in the RUL packet, the first 6LR  
 (6LR\_1) will add an RPI inside a new IPv6-in-IPv6 header. The IPv6-  
 in-IPv6 header will be addressed to the root. This case is identical  
 to the storing-mode case (see Section 7.2.3).

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

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

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

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

In this case the flow comprises:

Internet src --> root (6LBR) --> 6LR\_i --> RUL (IPv6 dst node)

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

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

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

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

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

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

### 8.3. Non-SM: Interaction between leaves

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

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

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

In this case the flow comprises:

RAL src --> 6LR\_ia --> root (6LBR) --> 6LR\_id --> RAL dst

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

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

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

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

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

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

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

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

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



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

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

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

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

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

In this case the flow comprises:

RAL --> 6LR\_ia --> root (6LBR) --> 6LR\_id --> RUL (IPv6 dst node)

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

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

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

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

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

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

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

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

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

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

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

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR\_1 --> 6LR\_ia --> root (6LBR) --> 6LR\_id  
--> RAL dst (6LN)

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

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

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

In this scenario the RPI (RPI1) is added by the first 6LR (6LR\_1) inside an IPv6-in-IPv6 header addressed to the root. The 6LBR will remove this RPI, and add its own IPv6-in-IPv6 header containing an RH3 header and an RPI (RPI2).

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

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

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

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

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR\_1 --> 6LR\_ia --> root (6LBR) --> 6LR\_id  
--> RUL (IPv6 dst node)

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

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

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

This scenario is the combination of the previous two cases.

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

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

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

## 9. Operational Considerations of supporting RUL-leaves

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

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

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

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

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

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

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

## 10. Operational considerations of introducing 0x23

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

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

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

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

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

## 11. IANA Considerations

### 11.1. Option Type in RPL Option

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

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

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

Bit number	Description	Reference
3	RPI 0x23 enable	This document

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

#### 11.2. Change to the DODAG Configuration Options Flags registry

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

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

#### 11.3. Change MOP value 7 to Reserved

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

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



## 12. Security Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 13. Acknowledgments

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

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

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

### 14. References

## 14.1. Normative References

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

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

#### 14.2. Informative References

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

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

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

#### Authors' Addresses

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

Email: [mariainesrobles@gmail.com](mailto:mariainesrobles@gmail.com)

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

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)  
URI: <http://www.sandelman.ca/mcr/>



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

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