

Workgroup: ROLL

Updates: [6550](#), [6553](#), [8138](#) (if approved)

Published: 30 November 2023

Intended Status: Standards Track

Expires: 2 June 2024

Authors: P. Thubert, Ed.    R.A. Jadhav    M. Richardson  
                                 Huawei Tech    Sandelman

## **Root initiated routing state in RPL**

### **Abstract**

This document 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 itself, 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 2 June 2024.

### **Copyright Notice**

Copyright (c) 2023 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](#)
2. [Terminology](#)
  - 2.1. [Requirements Language](#)
  - 2.2. [References](#)
  - 2.3. [Glossary](#)
  - 2.4. [Domain Terms](#)
    - 2.4.1. [Projected Route](#)
    - 2.4.2. [Projected DAO](#)
    - 2.4.3. [Path](#)
    - 2.4.4. [Routing Stretch](#)
    - 2.4.5. [Track](#)
3. [Context and Goal](#)
  - 3.1. [RPL Applicability](#)
  - 3.2. [Multi-Topology Routing and Loop Avoidance](#)
  - 3.3. [Requirements](#)
    - 3.3.1. [Loose Source Routing](#)
    - 3.3.2. [forward Routes](#)
  - 3.4. [On Tracks](#)
    - 3.4.1. [Building Tracks With RPL](#)
    - 3.4.2. [Tracks and RPL Instances](#)
  - 3.5. [path Signaling](#)
    - 3.5.1. [Using Storing Mode Segments](#)
    - 3.5.2. [Using Non-Storing Mode joining Tracks](#)
  - 3.6. [Complex Tracks](#)
  - 3.7. [Scope and Expectations](#)
    - 3.7.1. [External Dependencies](#)
    - 3.7.2. [Positioning vs. Related IETF Standards](#)
4. [Extending existing RFCs](#)
  - 4.1. [Extending RFC 6550](#)
    - 4.1.1. [Projected DAO](#)
    - 4.1.2. [Projected DAO-ACK](#)
    - 4.1.3. [Via Information Option](#)
    - 4.1.4. [Sibling Information Option](#)
    - 4.1.5. [P-DAO Request](#)
    - 4.1.6. [Amending the RPI](#)
    - 4.1.7. [Additional Flag in the RPL DODAG Configuration Option](#)
  - 4.2. [Extending RFC 6553](#)
  - 4.3. [Extending RFC 8138](#)
5. [New RPL Control Messages and Options](#)
  - 5.1. [New P-DAO Request Control Message](#)
  - 5.2. [New PDR-ACK Control Message](#)
  - 5.3. [Via Information Options](#)
  - 5.4. [Sibling Information Option](#)

6.	<a href="#">Root Initiated Routing State</a>
6.1.	<a href="#">RPL Network Setup</a>
6.2.	<a href="#">Requesting a Track</a>
6.3.	<a href="#">Identifying a Track</a>
6.4.	<a href="#">Installing a Track</a>
6.4.1.	<a href="#">Signaling a Projected Route</a>
6.4.2.	<a href="#">Installing a Track Segment with a Storing Mode P-Route</a>
6.4.3.	<a href="#">Installing a lane with a Non-Storing Mode P-Route</a>
6.5.	<a href="#">Tearing Down a P-Route</a>
6.6.	<a href="#">Maintaining a Track</a>
6.6.1.	<a href="#">Maintaining a Track Segment</a>
6.6.2.	<a href="#">Maintaining a lane</a>
6.7.	<a href="#">Encapsulating and Forwarding Along a Track</a>
6.8.	<a href="#">Compression of the RPL Artifacts</a>
7.	<a href="#">Lesser Constrained Variations</a>
7.1.	<a href="#">Storing Mode main DODAG</a>
7.2.	<a href="#">A Track as a Full DODAG</a>
8.	<a href="#">Profiles</a>
9.	<a href="#">Backwards Compatibility</a>
10.	<a href="#">Security Considerations</a>
11.	<a href="#">IANA Considerations</a>
11.1.	<a href="#">RPL DODAG Configuration Option Flag</a>
11.2.	<a href="#">Elective 6LoWPAN Routing Header Type</a>
11.3.	<a href="#">Critical 6LoWPAN Routing Header Type</a>
11.4.	<a href="#">Registry For The RPL Option Flags</a>
11.5.	<a href="#">RPL Control Codes</a>
11.6.	<a href="#">RPL Control Message Options</a>
11.7.	<a href="#">SubRegistry for the Projected DAO Request Flags</a>
11.8.	<a href="#">SubRegistry for the PDR-ACK Flags</a>
11.9.	<a href="#">Registry for the PDR-ACK Acceptance Status Values</a>
11.10.	<a href="#">Registry for the PDR-ACK Rejection Status Values</a>
11.11.	<a href="#">SubRegistry for the Via Information Options Flags</a>
11.12.	<a href="#">SubRegistry for the Sibling Information Option Flags</a>
11.13.	<a href="#">Destination Advertisement Object Flag</a>
11.14.	<a href="#">Destination Advertisement Object Acknowledgment Flag</a>
11.15.	<a href="#">New ICMPv6 Error Code</a>
11.16.	<a href="#">RPL Rejection Status values</a>
12.	<a href="#">Acknowledgments</a>
13.	<a href="#">Normative References</a>
14.	<a href="#">Informative References</a>
	<a href="#">Authors' Addresses</a>

## 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 the 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. Routers inside the DODAG route along that graph up towards the Root for the default route and down towards destinations in the RPL domain for more specific routes. This specification expects as a pre-requisite a pre-existing RPL Instance with an associated DODAG and RPL Root, which are referred to as main Instance, main DODAG and main Root respectively. The main Instance is operated in RPL Non-Storing Mode of Operation (MOP).

With this specification, an abstract routing function called a Path Computation Element [[PCE](#)] (e.g., located in an central controller or collocated with the main Root) interacts with the main Root to compute Peer-to-Peer (P2P) paths within the main Instance. In Non-Storing Mode, the base topological information to be passed to the PCE, that is the knowledge of the main DODAG, is already available at the Root. This specification introduces protocol extensions that enrich the topological information available to the Root with sibling relationships that are usable but not leveraged to form the main DODAG.

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 of 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 translate the computed paths into RPL and install them as Projected Routes (aka P-Routes) inside the DODAG on behalf of a PCE.

A P-Route may be installed in either Storing and Non-Storing Mode, potentially resulting in hybrid situations where the Mode in which the P-Route operates is different from that of the RPL main Instance. P-Routes can be used as stand-alone Segments meant to reduce the size of the source routing headers, leveraging loose source routing operations down the main RPL DODAG. P-Routes can also be combined with other P-Routes to form a protection Path called a Track and signaled as a RPL Instance. A Track provides underlay shortcuts in an existing main Instance, each with its own RIB.

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

## 2.2. References

In this document, 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 "[Using RPI Option Type, Routing Header for Source Routes, and IPv6-in-IPv6 Encapsulation in the RPL Data Plane](#)" [[RFC9008](#)], the "[Reliable and Available Wireless \(RAW\) Architecture](#)" [[RAW-ARCHI](#)], and "[Terminology in Low power And Lossy Networks](#)" [[RFC7102](#)]. Both architecture documents define the concept of Track in a compatible fashion. This documents only builds Tracks that are DODAGs, meaning that all links are oriented From Ingress to Egress. This specification also utilizes the terms Segment and Lane that are also defined in the RAW Architecture.

As opposed to routing trees, RPL DODAGs are typically constructed to provide redundancy and dynamically adapt the forwarding operation to the state of the LLN links. Note that the plain forwarding operation over DODAGs does not provide redundancy for all nodes, since at least the node nearest to the Root does not have an alternate feasible successor.

RAW solves that problem by defining Protection Paths that can be fully non-congruent and can be activated dynamically upon failures. This requires additional control to take the routing decision early enough along the Track to route around the failure.

RAW only uses single-ended DODAGs, meaning that they can be reversed in another DODAG by reversing all the links. The Ingress of the Track is the Root of the DODAG, whereas the Egress is the Root of the reversed DODAG. From the RAW perspective, single-ended DODAGs are special Tracks that only have forward links, and that can be leveraged to provide Protection services by defining destination-oriented Protection Paths within the DODAG.

## 2.3. Glossary

This document often uses the following acronyms:

**ARQ:** Automatic Repeat Request, in other words retries

**FEC:** Forward Error Correction

**HARQ:** Hybrid Automatic Repeat Request, combining FEC and ARQ

**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  
**PCE:** Path Computation Element  
**PLR:** Point of Local Repair  
**RAN:** RPL-Aware Node (either a RPL router or a RPL-Aware Leaf)  
**RAL:** RPL-Aware Leaf  
**RH:** Routing Header  
**RIB:** Routing Information Base, aka the routing table.  
**RPI:** RPL Packet Information  
**RPL:** IPv6 Routing Protocol for Low-Power and Lossy Networks  
**RT0:** 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 an SM-VIO or a 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 "downwards" (root towards leaves) with *multicast* DIO messages that form the DODAG and "upwards" (leaves towards root) with *unicast* DAO messages that follow the DODAG. This is in contrast with traditional IGP that operate the same way in all directions and are thus called isotropic.

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

The RPL DODAG optimizes the P2MP (Point-to-MultiPoint) (from the Root) and MP2P (MultiPoint-to-Point) (towards the Root) paths, but the P2P (Point-to-Point) 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

The concept of Track is inherited from the "[6TiSCH Architecture](#)" [[RFC9030](#)] and matches that of a Protection Path in the [RAW Architecture](#)" [[RAW-ARCHI](#)]. A Track is 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.

[Figure 1](#) illustrates the mapping of the DODAG with the generic concept of a Track, with the DODAG Root acting as Ingress for the Track, and the mapping of Lanes and Segments, and only forward Segments, meaning that they are directional and progressing towards the destination.

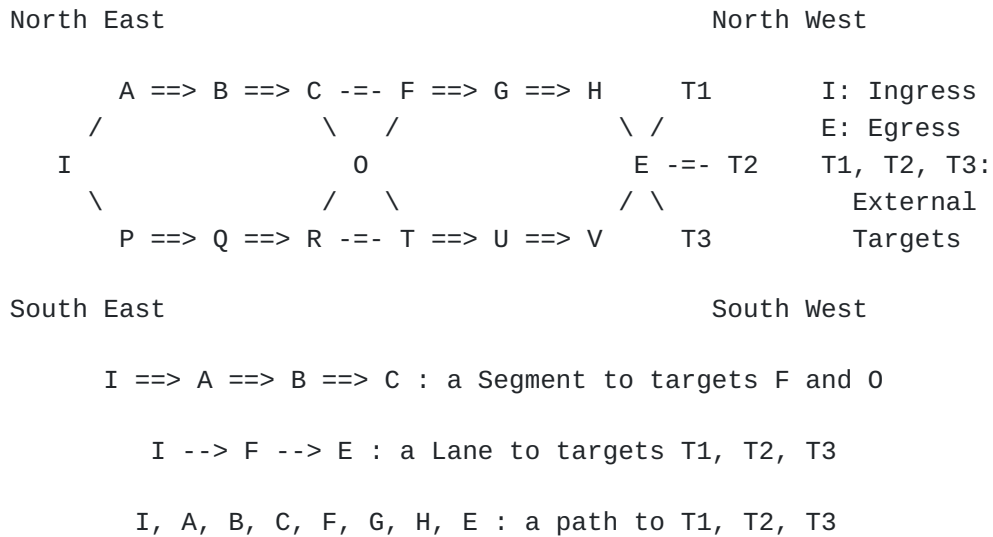


Figure 1: A Track and its Components

This specification builds Tracks that are DODAGs oriented towards a Track Ingress, and the forward direction for packets (aka forward) 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 InstanceID (typically of a Local Instance) that identifies a Track using the namespace owned by the Track Ingress. For Local Instances, 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. For Local Instances, the namespace is identified by the DODAGID for the Track and the tuple (DODAGID, TrackID) is globally unique. For Global Instances, the namespace is the whole RPL domain.

#### 2.4.5.3. Complex Track

A Track that can be traversed via more than one path (e.g., a DODAG).

#### 2.4.5.4. Stand-Alone

Refers to a Segment or a Lane that is installed with a single P-DAO that fully defines the path, e.g., a stand-alone segment is installed with a single Storing Mode Via Information option (SM-VIO) all the way between Ingress and Egress.

#### 2.4.5.5. Stitching

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

#### 2.4.5.6. Lane

The concept of Lane is defined in the [RAW Architecture](#) [RAW-ARCHI] as an end-to-end forward serial path. With this specification, a Lane is installed by the Root of the main DODAG using a Non-Storing Mode P-DAO message, e.g., I --> F --> E in [Figure 1](#).

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 Lane to signal the structure of a complex Track.

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

#### 2.4.5.7. Segment

A serial path formed by a strict sequence of nodes, along which a P-Route is installed, e.g., I ==> A ==> B ==> C in [Figure 1](#). 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 Lanes that form the structure of a complex Track. The same Segment may be leveraged by more than one Lane where the Lanes 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.

#### **2.4.5.7.1. Section of a Segment**

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

#### **2.4.5.7.2. 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 is constrained and the transmissions are 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 platoon 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 to 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 for selecting the RPL Instance that is used to forward a packet coming from the Backbone into the RPL domain and for setting the related RPL information in the packets. Each Instance creates its own routing table (RIB) in participating nodes, and the RIB associated to the instance must be used end to end in the RPL domain. To that effect, RPL tags the packets with the Instance ID in a Hop-by-Hop extension Header. 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 [\[RFC9450\]](#), which demand high availability and reliability, and as a consequence require both short and diverse paths.

### **3.2. Multi-Topology Routing and Loop Avoidance**

RPL first forms a default route in each node towards the Root, and those routes together coalesce as a Directed Acyclic Graph oriented upwards. RPL then constructs routes to destinations signaled as Targets in the reverse direction, down the same DODAG. To 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 itself. 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 a single unicast packet to the Root along the default route to update it, and the connectivity to the node is restored immediately; this mode is preferable for use cases where internet connectivity is dominant, or when the Root controls the network activity in the nodes, which is the case of this draft.

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 compared to Non-Storing MOP, for better P2P connectivity. However, a new route takes a longer time to propagate to the Root, since it takes time for the Distance-Vector protocol to operate hop-by-hop, and the connectivity from the internet to the node is restored more slowly upon node 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 a 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.

In the simplest mode of this specification, Storing-Mode P-Routes can be deployed to join the dots of a loose source routing header (SRH) in the main DODAG. In that case, all the routes (source routed and P-Routes) belong to the Routing Information base (RIB) associated with the main Instance. Storing-Mode P-Routes are referred to as Segments in this specification.

A set of P-Routes can also be projected to form a dotted-line underlay of the main Instance and provide Traffic Engineered paths for an application. In that case, the P-Routes are installed in Non-Storing Mode and the set of P-Routes is called a Track. A Track is associated with its own RPL Instance, and, as any RPL Instance, with its own Routing Information base (RIB). As a result, each Track defines a routing topology in the RPL domain. As for the main DODAG, Segments associated to the Track Instance may be deployed to join the dots using Storing-Mode P-Routes.

Routing in a multi-topology domain may cause loops unless strict rules are applied. This specification defines two strict orders to ensure loop avoidance when projected routes are used in a RPL domain, one between forwarding methods and one between RPL Instances, seen as routing topologies.

The first and strict order relates to the forwarding method and the more specifically the origin of the information used in the next-hop

computation. The possible forwarding methods are: 1) to a direct next hop, 2) to an indirect neighbor via a common neighbor, 3) along a Segment, and 4) along a nested Track. The methods are strictly ordered as listed above, more in [Section 6.7](#). A forwarding method may leverage any of the lower order ones, but never one with a higher order; for instance, when forwarding a packet along a Segment, the router may use direct or indirect neighbors but cannot use a Track. The lower order methods have a strict precedence, so the router will always prefer a direct neighbor over an indirect one, or a Segment within the current RPL Instance vs. another Track.

The second strict and partial order is between RPL Instances. It allows the RPL node to detect an error in the state installed by the PCE, e.g., after a desynchronization. That order must be defined by the administrator for his RPL domain and defines a DODAG of underlays with the main Instance as Root. The relation of RPL instances may be represented as a DODAG of instances where the main instance is Root. The rule is that a RPL Instance may leverage another RPL instance as underlay if and only if that other Instance is one of its descendants in the graph. Supporting this method is OPTIONAL for nested Tracks and REQUIRED between a Track instance and the main instance. It may be done using network management, or future extensions to this specifications. When it is not communicated, then the RPL nodes consider by default that all Track instances are children of the main instance, and do not attempt to validate the order for nested Tracks, trusting the PCE implicitly. As a result, a packet that is being forwarded along the main Instance may be encapsulated in any Track, but a packet that was forwarded along a Track MUST NOT be forwarded along the default route of main Instance.

### **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.

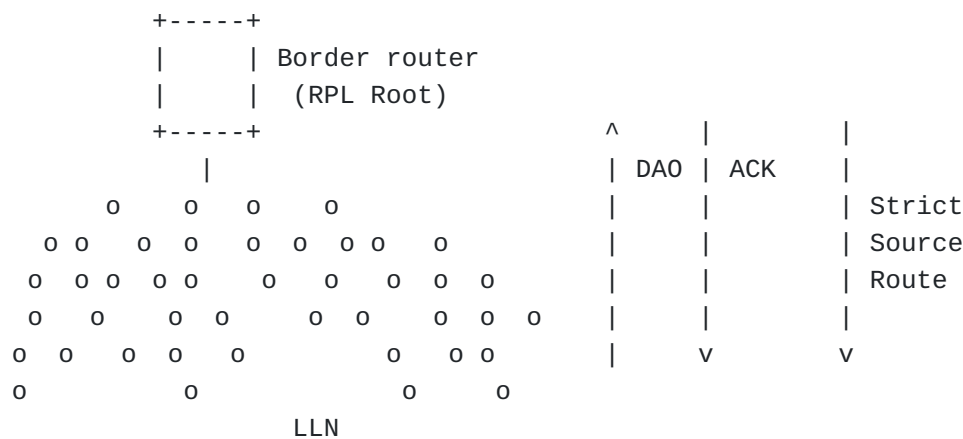


Figure 2: RPL Non-Storing Mode of operation

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. This results in the wireless bandwidth near the Root being the limiting 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 increases with the depth of the network. In some use cases, a RPL network forms long lines along physical structures such as streets for lighting. Limiting the packet size is beneficial to the energy budget, directly for the current transmission, but also indirectly since it reduces the chances of frame loss and energy spent in retries, e.g., by ARQ over one hop at Layer-2, or end-to-end at upper layers. Using smaller packets also reduces the chances of packet fragmentation, which is highly detrimental to the LLN operation, in particular when fragments are forwarded but not recovered, see [[RFC8930](#)] vs. [[RFC8931](#)] for more.

A limited amount of well-targeted routing state would allow the source routing operation to be loose as opposed to strict, and reduce the overhead of routing information in packets. Because the capability to store routing state in every node is limited, the decision of which route is installed where can only be optimized with global knowledge of the system, knowledge that the Root or an

associated PCE may possess by means that are outside 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 itself and then back down, including 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 by reducing the load at the Root.

To limit the need for source route headers in deep networks, one possibility is to store a routing state associated with the main DODAG in select RPL routers down the path. The Root may elide the sequence of routers that is installed in the network from its source route header, which therefore becomes loose, in contrast to being strict in [RPL].

### 3.3.2. forward 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) forward routes in a RPL network will generally experience elongated (stretched) paths versus direct (optimized) paths, since routing between two nodes always happens via a common parent, as illustrated in [Figure 3](#):

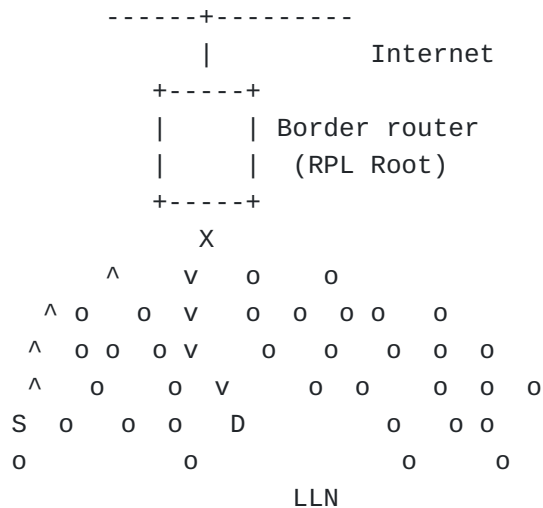


Figure 3: Routing Stretch between S and D via common parent X along North-South Paths

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 turns out that it is often beneficial to enable forward P2P routes, either if the RPL route presents a stretch from the shortest path, or if the new route is engineered with a different objective, and this 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 alternative based on centralized route computation.

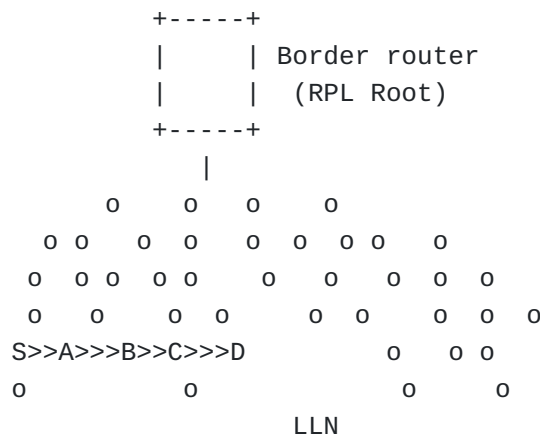


Figure 4: More direct forward Route between S and D



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 lane. 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 Lanes 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 path provides only one path between Ingress and Egress. It comprises at most one Lane. A Stand-Alone Segment implicitly defines a path 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 Lanes 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 in the rightmost bits and Bit 0 set.

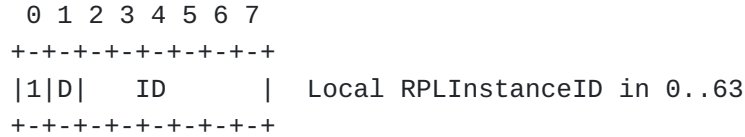


Figure 5: Local RPLInstanceID Encoding

A Track typically forms an underlay to the main Instance, and is associated with a Local RPL Instance from which the RPLInstanceID is used as the TrackID. When a packet is placed on a Track, it is encapsulated IP-in-IP with a RPL Option containing a RPI which signals the RPLInstanceID. The encapsulating source IP address and RPI Instance are set to the Track Ingress IP address and local RPLInstanceID, respectively, more in [Section 6.3](#).

A Track typically offers service protection across several lanes. As a degraded form of a Track, a path made of a single lane (i.e., offering no protection) can be used as an alternative to a Segment for forwarding along a RPL Instance. In that case, instead of following native routes along the instance, the packets are encapsulated to signal a more specific source-routed path between the loose hops in the encapsulated source routing header.

If the encapsulated packet follows a global instance, then the lane may be part that global instance as well, for instance the global instance of the main DODAG. This can only be done for global instances because the Ingress node that encapsulates the packets over the lane is not the Root of the instance, so the source address of the encapsulated packet cannot be used to determine the Track along the way.

### 3.5. path Signaling

This specification enables setting up a P-Route along either a lane 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. In order to clarify the techniques that may be used to install a P-Route, this section takes the simple case of the path illustrated in [Figure 6](#). So the goal is to build a path from node A to E for packets towards E's neighbors F and G along A, B, C, D and E as opposed to via the Root:

```

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

```

Figure 6: Reference Track

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 Lane or a Segment in the P-DAO messages, either in Storing Mode (SM-VIO) or Non-Storing Mode (NSM-VIO). One P-DAO message 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 Lanes and Segments signaling and operation, this section provides examples of how route projection works through variations of a simple example. This simple example illustrates the case of host routes, though RPL Targets can also be prefixes.

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 comma-separated Targets, e.g., F is a Target for Segment C==>D==>E. In this example, A is the Track Ingress and E is the 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).

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==>B==>C
- \*P-DAO 3 signals F and G via the A-->E Track

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

Loose sequences of hops are expressed in Non-Storing Mode; this is why P-DAO 3 contains a NSM-VIO. With this specification:

- \*the DODAGID to be used by the Ingress as source address is signaled in the DAO base object (see [Figure 8](#)) .
- \*the via list in the VIO is encoded as an SRH-6LoRH (see [Figure 16](#)), and it starts with the address of the first hop node after the Ingress node in the loose hop sequence.
- \*the via list ends with the address of the Egress node.

Note well:

The Egress of a Non-Storing Mode P-Route is implicitly a target; it is not listed in the RPL Target Options but still accounted for as if it was. The only exception is when the Egress is the only address listed in the VIO, in which case it would indicate via itself which would be non-sensical.

Also:

By design, the list of nodes in a VIO in Non-Storing Mode is exactly the list that shows in the encapsulation SRH. So in the cases detailed below, if the Mode of the P-DAO is Non-Storing, then the VIO row can be read as indicating the SRH as well.

### 3.5.1. Using Storing Mode Segments

A==>B==>C and C==>D==>E are Segments of the 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==>D==>E-to-F,G

\*P-DAO 2 signals A==>B==>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

Field	P-DAO 1 to E	P-DAO 2 to C
(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

Note:

the " sign is used throughout those tables to indicate the same value as in the row above.

Packets originating at A going to F or G do not require 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	Any but 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 lane 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 for P-DAO 1 and 2. E is not an implicit Target for P-DAO 3 either, since E is the only entry in the VIO.

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. This is why in the tables below, E may show as IPv6 Destination Address only if the IPv6 Source Address X is different from A. Conversely, the encapsulation is always done when the IPv6 Destination Address is F or G. Other destination addresses do not match this P-Route and are not subject to 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	either E if(X!=A), or 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 and sends it down 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 Track Lanes are leveraged 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-(E),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

	P-DAO 1 to E	P-DAO 2 to B	P-DAO 3 to A
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	B, 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 lane. 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 until C then E	(A, 129)
Inner	X	either E if(X!=A), or 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  $C \Rightarrow D \Rightarrow E$ -to-(E), F, G

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

$A \Rightarrow B \Rightarrow C$  and  $C \Rightarrow D \Rightarrow E$  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 (using ND to indicate that the address is discovered by IPv6 Neighbor Discovery [[RFC4861](#)] [[RFC8505](#)] or an equivalent method:

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 could be generated with the RPI and the SRH, and no encapsulation. Alternatively, A may generate a native packet to the target, and then encapsulate it with an RPI and an SRH indicating the source-routed path leading to E, as it would for a packet that it routes coming from another node. This is effectively the same case as for packets generated by the root in a RPL network in Non-Storing mode, see section 8.1.3 of [RFC9008]. The latter is often preferred since it leads to a single code path, and the destination when it is F or G, does not understand and process the RPI or the SRH. 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 until 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 to 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	F, G

Table 13: P-DAO Messages

Note in the above that E is an implicit Target in P-DAO 1 and so is C in P-DAO 2. As Non-Storing Mode Egress nodes addresses, they not listed in the respective RTOs.

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 until 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-(E),F,G Track

Non-Storing Mode P-DAO 1 is sent to C and Non-Storing Mode P-DAO 2 and 3 are sent to 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
"	B, C	P-DAO 2	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 until 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 until 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 building a collection of Lanes between the same Ingress and Egress Nodes and combining them within the same TrackID, as shown in [Figure 7](#). Lanes may cross at the edges of loose hops or remain parallel.

The Segments that join the loose hops of a Lane are installed with the same TrackID as the Lane. But each individual Lane and Segment has its own P-RouteID which allows it to be managed separately. 2 Lanes of the same Track may cross at a common node that participates to a Segment of Each Lane. In that case the common node has more than one next hop in its RIB associated to the Track, but no specific signal in the packet to indicate which Segment is being followed. A next hop that can reach the loose hop is selected.



congruent with Lane 1 until node B and congruent with Lane 2 from node H on, abstracting Segment 5 as an forward Segment.

### **3.7. Scope and Expectations**

#### **3.7.1. External Dependencies**

This specification expects that the 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 document 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 document. 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 Lane 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, 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 a 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 forward 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 Lanes, joined by strict routed Segments, all oriented forward.

The RAW Architecture defines a dataplane extension of the PCE called the Point of Local Repair (PLR), that adapts the use of the path redundancy within a Track to defeat the diverse causes of packet loss. The PLR controls the forwarding operation of the packets within a Track. This specification can use but does not impose a PLR and does not provide the policies that would select which packets are routed through which path within a Track, in other words, how the PLR 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 forward 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 specifications. 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 forward 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 a Track Segment or a lane, respectively.

The 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, it then discards the message. When the Root initiates communication to a node that it has not communicated with before and which it has not ascertained to implement this specification (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 forwards the packets towards the Targets via the Track using normal longest match IP forwarding.

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

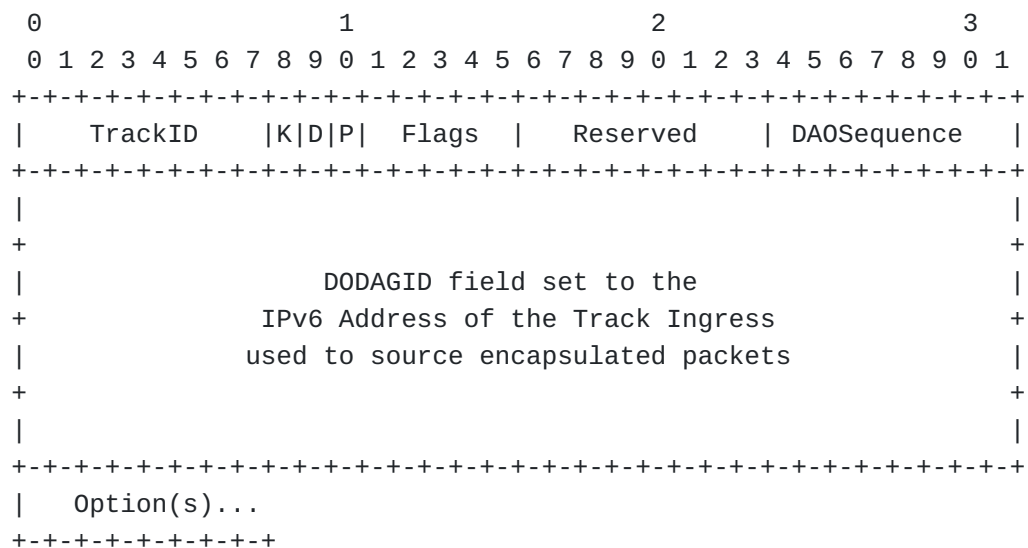


Figure 8: Projected DAO Base Object

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 document 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](#):

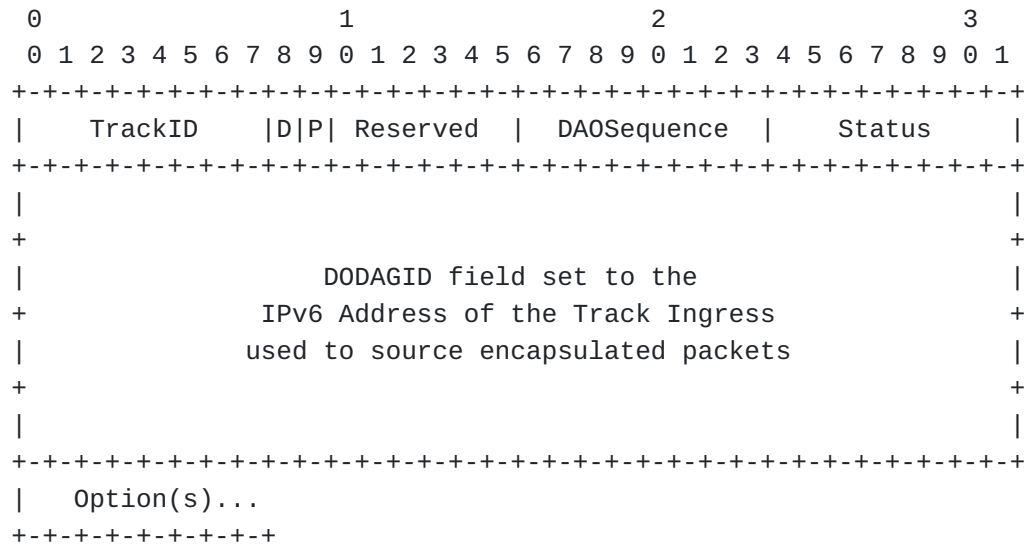


Figure 9: Projected DAO-ACK Base Object

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 document Extends the CM0 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 lane 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 main Root, including multicast DAO (see section 9.10 of [\[RPL\]](#)).

This draft AMENDS the multicast DAO operation as follows:

1. A multicast DAO message MUST be used only to advertise information about the node (using the Target Option), and direct Link Neighbors such as learned by Neighbor Discovery (using the Sibling Information Option).
2. The multicast DAO may be used to enable direct and indirect (via a common neighbor) P2P communication without needing the DODAG to relay the packets. The multicast DAO exposes the sender's addresses as Targets in RTOs and the sender's neighbors addresses as siblings in SIOs; this tells the sender's neighbors that the sender is willing to act as a relay between those of its neighbors that are too far apart.

#### **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 DODAG (as a 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.

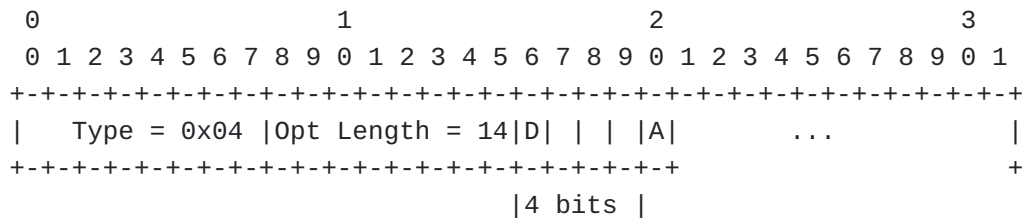


Figure 10: DODAG Configuration Option (Partial View)

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 [\[RFC9008\]](#) Amends [\[RPL\]](#) 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:

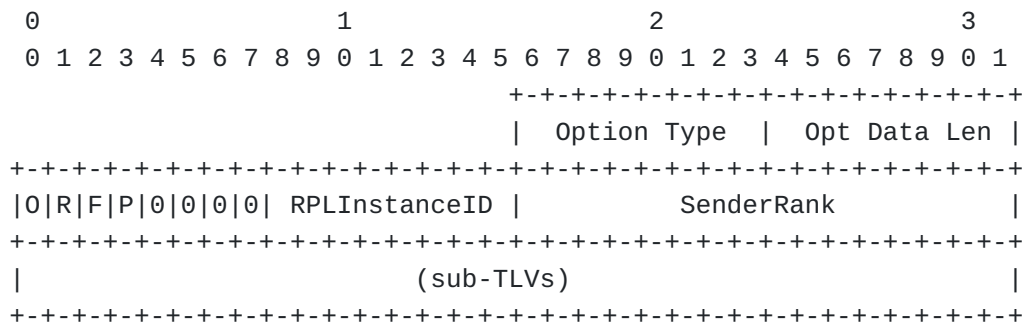


Figure 11: Amended RPL Option Format

**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:

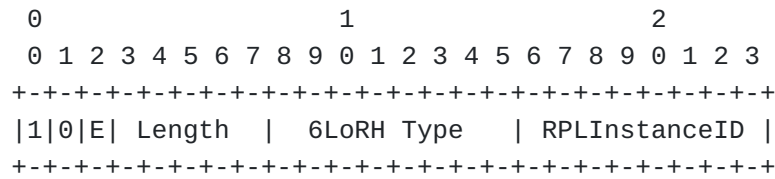


Figure 12: P-RPI-6LoRH Format

**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 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 Root commits to maintaining 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 use an asynchronous PDR-ACK with a 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 attempt 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 RT0 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:

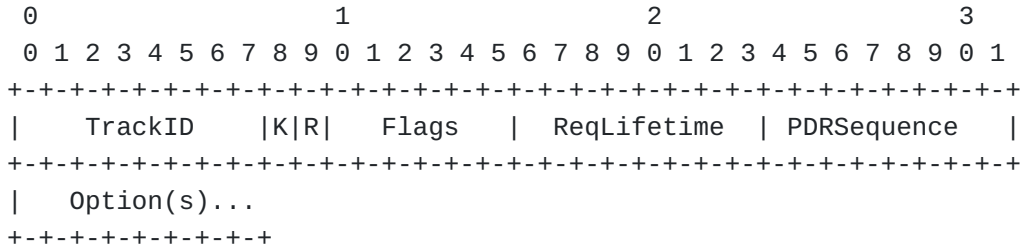


Figure 13: New P-DAO Request Format

**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). The value of 255 (0xFF) represents infinity (never time out).

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:

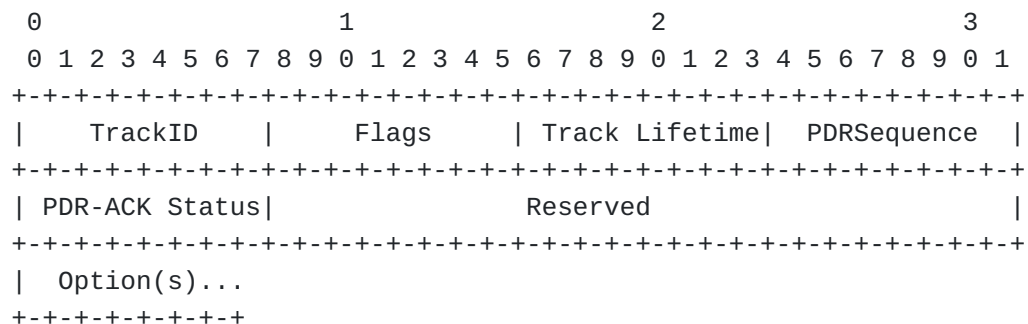


Figure 14: New PDR-ACK Control Message Format

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

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

**Track Lifetime:** Indicates the remaining Lifetime for the Track, expressed in Lifetime Units; The value of 255 (0xFF) represents infinity. 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](#):

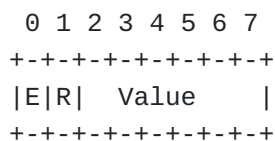


Figure 15: PDR-ACK status Format

- 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 Lane (using Non-Storing Mode) or 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 would make the VIO globally shorter; this means that more than one SRH-6LoRH may be present.

The format of the Via Information Option is as follows:

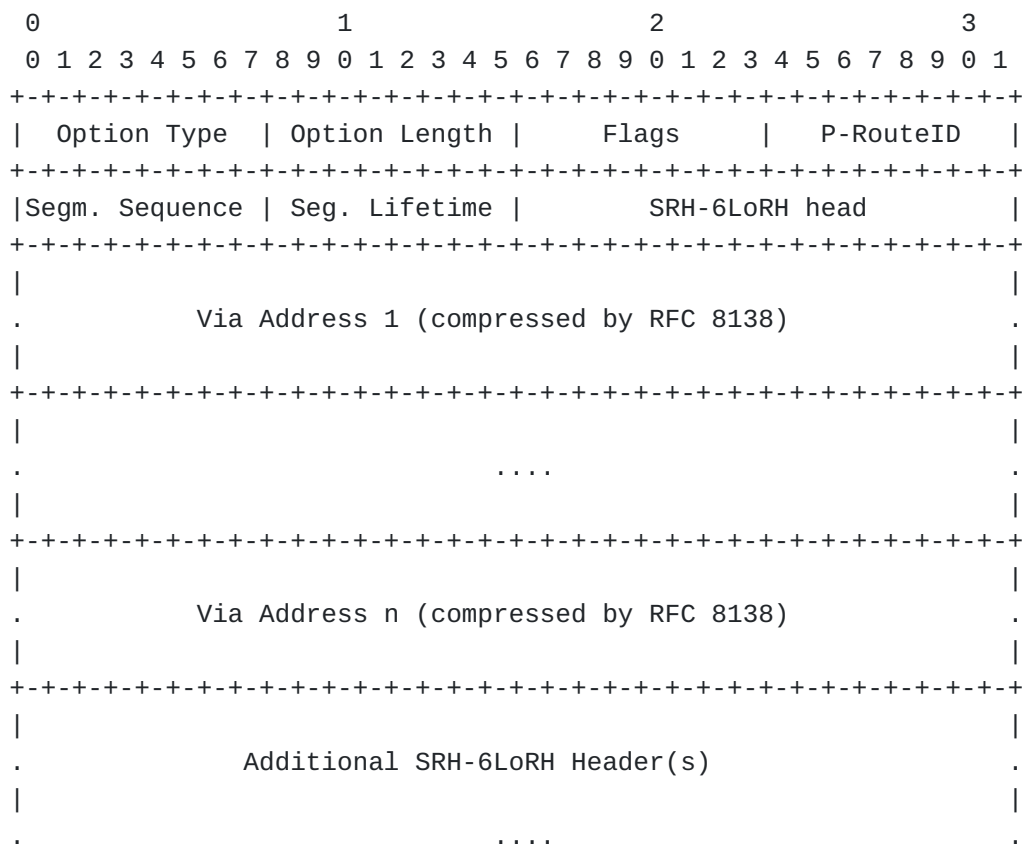


Figure 16: VIO format

**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.

**Flags:** 8-bit field. No flag is defined in this specification. The field MUST be set to 0 by the sender and ignored by the receiver.

**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 path, i.e., made of a single Segment/Lane. In an SM-VIO, the P-RouteID indicates an actual Segment. In an NSM-VIO, it indicates a Lane, that is a path 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 information about 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 to 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 performance can be expected in 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:

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	2	3	4	5	6	7	8	9
Type										Option Length										S B Flags Comp.										Opaque									
Step in Rank										Reserved																													
Sibling DODAGID (if the D flag not set)																																							
Sibling Address																																							



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 be 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. In some scenarios such as the case of an Industrial Alliances that uses RPL for a particular use / environment, this field MAY be redefined to fit the needs of that case.

**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 present. 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 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 which 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 the 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 over time 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 elapLR - 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 main Root MAY use P-DAO messages to add better routes in the main Instance in conformance with the routing objectives in that Instance.

To achieve this, the main Root MAY install a Segment along a path down the main DODAG, which is operated in Non-Storing Mode. This enables a loose source routing and reduces the size of the Routing Header, see [Section 3.3.1](#). The main Root MAY also install a lane across the main DODAG to complement the routing topology.

When adding a P-Route to the RPL main DODAG, the main 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 main Root, so it is preferred.

- \*The main 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 Egress Endpoints. To achieve this, the main Root MUST set up a Local RPL Instance (see section 5 of [\[RPL\]](#)), and the Local

RPLInstanceID serves as the 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 PDRs 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 main 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 main 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 path 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-Lane Track. A single-Lane 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 path.

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 Lane of the same Track. In the latter case, the Targets of the P-Route must include the next hop of the Lane 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 a 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 DODAG installs a Track as a collection of P-Routes, using a Projected-DAO (P-DAO) message for each individual lane 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 lane MUST be sent to a GUA or a ULA of the Ingress of the Lane; it must contain the full list of hops in the Lane unless the Lane 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 Lane, 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 any of these conditions happen, and when one does, 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 installation of 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 main DODAG, or a Track associated with a local RPL Instance.



position preceeding the one that contains 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 appear in the SM-VIO after its own address, 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 until the P-DAO is propagated to the 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 lane 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 RT0. 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.



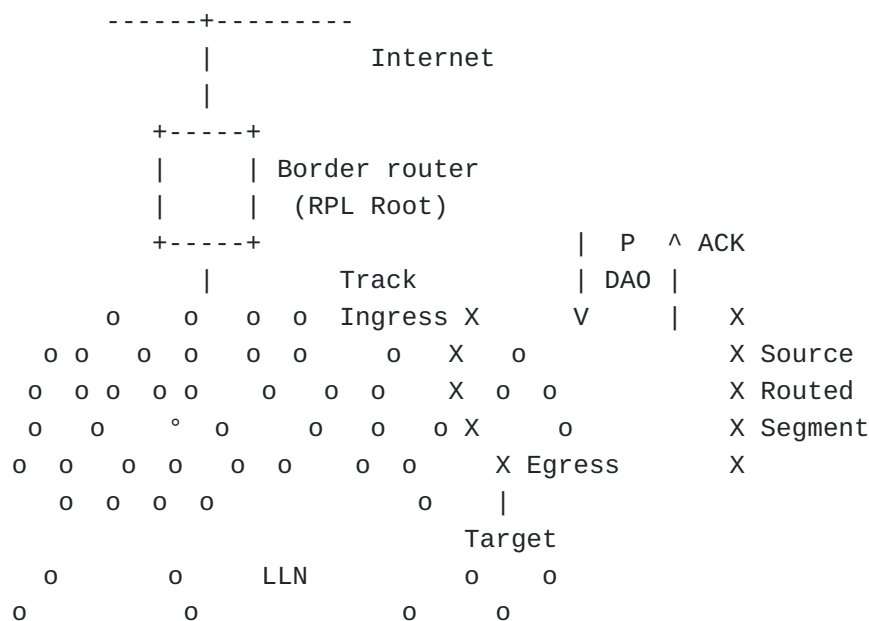


Figure 19: Projecting a Non-Storing Route

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 Lane(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 Lanes that compose it one by one.

Since the state about a Lane of a Track is located only on the Ingress Node, the Root cleans up the Lane by sending an NSM-VIO to the Ingress indicating the TrackID and the P-RouteID of the Lane 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 Lane 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 Lane 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, respectively), while retaining 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 drain their traffic, the Root tears down the remaining section(s) of the old Segments as described in [Section 6.5](#).

### 6.6.2. Maintaining a lane

This specification allows the Root to add Lanes 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 Lane is loose, then the Segments that join the hops must be created first. It makes sense to add a new Lane before removing one that is becoming excessively lossy, and switch to the new Lane before removing the old. Dropping a Track before the new one is installed would reroute the traffic via the root; this may increase the latency beyond acceptable thresholds, and overload 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 might be routed back and reinjected at the Ingress of the first.

It is also possible to update a lane 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 Lane 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 Lane is issued.

Packets that are in flight over the old version of the lane still follow the old source route path over the old Segments. After a reasonable time to enable the deprecated Segments to drain their

traffic, 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 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, ending with the Track Egress.

- \*The preferred alternative 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 ending with 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 as 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 it must include an RPI in the encapsulation to signal the TrackID.

A packet that is being routed over the RPL Instance associated to a first Non-Storing Mode Track MAY be placed recursively 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 Segment must be strict and a packet that it placed in a Storing Mode Segment MUST follow that Segment till the Segment Egress.

It is known that a packet is forwarded along a Track by the source address and the RPI in the encapsulation. The Track ID is used to identify the RIB entries associated to that Track, which, in intermediate nodes, correspond to the P-routes for the segments of the Track that the forwarding router is aware of. The packet processing uses a precedence that favors self delivery or routing header handling when one is present, then delivery to direct neighbors, then to indirect neighbors, then routing along a segment along the Track, and finally as a last resort injecting the packet in another Track.

To achieve this, the packet handling logic MUST happen in the following order:

\*If the destination of the packet is self:

1. if the header chain contains a RPL Source Route Header that is not fully consumed, then the packet is forwarded along the Track as prescribed by [\[RFC6554\]](#), meaning that the next entry in the routing header becomes the destination;

2. otherwise: if the packet was encapsulated, then the packet is decapsulated and the forwarding process recurses; else the packet is delivered to the stack.

\*Otherwise, the packet is forwarded as follows:

1. If the destination of the packet is a direct neighbor, e.g., installed by IPv6 Neighbor Discovery, then the packet the packet MUST be forwarded to that neighbor;
2. Else If the destination of the packet is an indirect neighbor, e.g., installed by a multicast DAO message from a common neighbor, see [Section 4.1.4](#), then the packet MUST be forwarded to the common neighbor;
3. Else, if there is a RIB entry for the same Track (e.g., installed by an SM-VIO in a DAO message with the destination as target), and the next hop in the RIB entry is a direct neighbor, then the packet is passed to that neighbor;
4. Else, if there is a RIB entry for the different Track (e.g., installed by an NSM-VIO in a DAO message with the destination as target), then the packet is encapsulated to be forwarded along that Track and the forwarding process recurses; otherwise the packet is dropped.
5. To avoid loops, and as opposed to packets that were not encapsulated, a packet that was decapsulated from a Track MUST NOT be routed along the default route of the main DODAG; this would mean that the end-to-end path is uncontrolled. The node that discovers the fault MUST discard the packet.

The node that drops a packet for either of the reasons above 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 to 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 receiving this message, the encapsulating node SHOULD stop using the source route path for a reasonable period of time which 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 an IPv6-in-IPv6 encapsulation is needed (see Table 19 of [RFC9008]):

```

+-+ ... +- ... +- ... +-+- ... +-+-+- ... +-+-+ ... +- ... +-...
|11110001| SRH- | RPI- | IP-in-IP | NH=1      |11110CPP| UDP | UDP
| Page 1 | 6LoRH | 6LoRH | 6LoRH   |LOWPAN_IPHC| UDP   | hdr |Payld
+-+ ... +- ... +- ... +-+- ... +-+-+- ... +-+-+ ... +- ... +-...
                                     <=      RFC 6282      =>
<===== Inner packet ===== = =

```

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 starts with the SRH-6LoRH, and is exactly the packet represented in Figure 20, from the second octet onward.

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.





## 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 knows the DODAG topology, so the additional topological information is reduced to the siblings.

- \*The downward 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 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 less 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 just like 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 factorising 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 Lanes 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 the 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 lane and produces 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 PLR 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 PLR needs the full richness of a DODAG to form any path that could 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 PLR 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 PLR 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 document 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 implementing only a portion of this specification to meet a particular use case.

Profiles 0 to 2 operate in the main 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; Profile 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 Instance MUST be used as

TrackID. Note that the Ingress node encapsulates but is not the Root, as it does not own the DODAGID.

**Profile 3** In order to form the best path possible, this Profile requires 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 forward Tracks that are inside the main DODAG but do not necessarily follow it. A Track is formed as one or more strict 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 loose source routing between the Ingress and the Egress of the Track. As in Profile 1, Storing Mode P-Routes form the connections in the loose source route.

**Profile 6** Profile 6 Combines Profile 4 with Profile 2 and also enables 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 the node capabilities and whether they 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 the DIO when the conditions of success are in place, in particular when all the nodes that could be on the 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 injecting 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 not 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 under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" 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

IANA is requested to update the "Elective 6LoWPAN Routing Header Type" registry that was created for [[RFC8138](#)] under the heading "Elective 6LoWPAN Routing Header Type" in [[IANA-6LO](#)] and assign 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

IANA is requested to update the "Critical 6LoWPAN Routing Header Type" registry that was created for [[RFC8138](#)] under the heading "Critical 6LoWPAN Routing Header Type" in [[IANA-6LO](#)] and assign the following value:

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

Table 23: New Critical 6LoWPAN Routing Header Type

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

IANA is requested to create a registry for the 8-bit "RPL Option Flags" field, as detailed in [Figure 11](#), under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". 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 24](#):

Bit number	Indication When Set	Reference
0	Down '0'	[ <a href="#">RFC6553</a> ]
1	Rank-Error (R)	[ <a href="#">RFC6553</a> ]
2	Forwarding-Error (F)	[ <a href="#">RFC6553</a> ]
3 (Suggested)	Projected-Route (P)	THIS RFC
4..255	Unassigned	

Table 24: Initial PDR Flags

#### 11.5. RPL Control Codes

IANA is requested to update the "RPL Control Codes" registry under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" 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

IANA is requested to update the "RPL Control Message Options" registry under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" 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 requested to create a registry for the 8-bit "Projected DAO Request (PDR)" field under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". 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)

\*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
2..255	Unassigned	

Table 27: Initial PDR Flags

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

IANA is requested to create a registry for the 8-bit "PDR-ACK Flags" field under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". 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)

\*Capability description

\*Reference

Registration procedure is "Standards Action" [[RFC8126](#)]. No bit is currently assigned for the PDR-ACK Flags.

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

IANA is requested to create a registry for the 8-bit "PDR-ACK Acceptance Status Values" under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". Each value is tracked with the following qualities:

\*Value

\*Meaning

\*Reference

the possible values are expressed as a 6-bit unsigned integer (0..63). the registration procedure is "Standards Action" [[RFC8126](#)].

The (suggested) initial allocation is as indicated in [Table 28](#):

Value	Meaning	Reference
0	Unqualified Acceptance	THIS RFC
1..63	Unassigned	

Table 28: Acceptance values of the PDR-ACK Status

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

IANA is requested to create a registry for the 6-bit "PDR-ACK Rejection Status Values" under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". Each value is tracked with the following qualities:

\*Value

\*Meaning

\*Reference

the possible values are expressed as a 6-bit unsigned integer (0..63). the registration procedure is "Standards Action" [[RFC8126](#)].

The (suggested) initial allocation is as indicated in [Table 29](#):

Value	Meaning	Reference
0	Unqualified Rejection	THIS RFC
1	Transient Failure	THIS RFC
2..63	Unassigned	

Table 29: Rejection values of the PDR-ACK Status

#### 11.11. SubRegistry for the Via Information Options Flags

IANA is requested to create a registry for the 8-bit "Via Information Options (VIO) Flags" field under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". 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)
- \*Capability description
- \*Reference

Registration procedure is "Standards Action" [[RFC8126](#)]. No bit is currently assigned for the VIO Flags, more in [Section 5.3](#).

#### 11.12. SubRegistry for the Sibling Information Option Flags

IANA is requested to create a registry for the 5-bit "Sibling Information Option (SIO) Flags" field under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". The bits are indexed from 0 (leftmost) to 4. 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](#), more in [Figure 17](#):

Bit number	Capability description	Reference
0 (Suggested)	"S" flag: Sibling in same DODAG as Self	THIS RFC
1..4	Unassigned	

Table 30: Initial SIO Flags

#### 11.13. Destination Advertisement Object Flag

IANA is requested to update the "Destination Advertisement Object (DAO) Flags" registry created in Section 20.11 of [[RPL](#)] under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" as indicated in [Table 31](#), more in [Section 4.1.1](#):

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

IANA is requested to update the "Destination Advertisement Object (DAO) Acknowledgment Flags" registry created in Section 20.12 of [RPL] under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" as indicated in [Table 32](#), more in [Section 4.1.2](#):

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.

This specification requires that a new code is allocated from the 'ICMPv6 "Code" Fields' heading of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" Registry for "Type 1 - Destination Unreachable", with a suggested code value of 9, to be confirmed by IANA to indicate an "Error in P-Route".

#### 11.16. RPL Rejection Status values

IANA is requested to update the "RPL Rejection Status" registry under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)" as indicated in [Table 33](#):

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 to Remous-Aris Koutsiamanis, Li Zhao, Dominique Barthel, 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 and to Ines Robles for her shepherding and thorough review. Many thanks to Sue Hares for their comments and suggestions during the IETF last call and IESG review cycle.

### 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>>.
- [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>>.
- [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>>.
- [RAW-ARCHI] Thubert, P., "Reliable and Available Wireless Architecture", Work in Progress, Internet-Draft, draft-ietf-raw-architecture-16, 20 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-raw-architecture-16>>.

#### 14. Informative References

- [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>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [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>>.

**[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>>.

**[RFC9450]**

Bernardos, C.J., Ed., Papadopoulos, G., Thubert, P., and F. Theoleyre, "Reliable and Available Wireless (RAW) Use Cases", RFC 9450, DOI 10.17487/RFC9450, August 2023, <<https://www.rfc-editor.org/info/rfc9450>>.

**[I-D.kuehlewind-update-tag]** Kühlewind, 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, R. and C. Krähenbühl, "A

Vocabulary of Path Properties", Work in Progress, Internet-Draft, draft-irtf-panrg-path-properties-08, 6 March 2023, <<https://datatracker.ietf.org/doc/html/draft-irtf-panrg-path-properties-08>>.

**[PCE]**

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

**[IANA-6LO]**

IETF, "IPv6 Low Power Personal Area Network Parameters", <<https://www.iana.org/assignments/6lowpan-parameters/6lowpan-parameters.xhtml>>.

**Authors' Addresses**

Pascal Thubert (editor)



06330 Roquefort-les-Pins  
France

Email: [pascal.thubert@gmail.com](mailto:pascal.thubert@gmail.com)

Rahul Arvind Jadhav  
Huawei Tech  
Kundalahalli Village, Whitefield,  
Bangalore 560037  
Karnataka  
India

Phone: [+91-080-49160700](tel:+91-080-49160700)

Email: [rahul.ietf@gmail.com](mailto:rahul.ietf@gmail.com)

Michael C. Richardson  
Sandelman Software Works

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)

URI: <http://www.sandelman.ca/>