ROLL
Internet-Draft
Intended status: Standards Track
Expires: 25 March 2022

P. Thubert, Ed.
Cisco Systems
R.A. Jadhav
Huawei Tech
M. Gillmore
Itron

21 September 2021

Root initiated routing state in RPL
draft-ietf-roll-dao-projection-20

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 self, for a chosen duration. This potentially enables routes that are more optimized or resilient than those obtained with the classical distributed operation of RPL, either in terms of the size of a Routing Header or in terms of path length, which impacts both the latency and the packet delivery ratio.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

2. Terminology

2.1. Requirements Language

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

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" [6TiSCH-ARCHI], the "Deterministic Networking Architecture" [RFC8655], the "Reliable and Available Wireless (RAW) Architecture/Framework" [RAW-ARCHI], and "Terminology in Low power And Lossy Networks" [RFC7102].

2.3. Glossary

This document often uses the following acronyms:

CMO:	Control Message Option
DAO:	destination Advertisement Object
DAG:	Directed Acyclic Graph
DODAG:	destination-Oriented Directed Acyclic Graph; A DAG with only one vertex (i.e., node) that has no outgoing edge (i.e., link)
GUA:	IPv6 Global Unicast Address
LLN:	Low-Power and Lossy Network

MOP:	RPL Mode of Operation
P-DAO:	Projected DAO
P-Route:	Projected Route
PDR:	P-DAO Request
RAN:	RPL-Aware Node (either a RPL router or a RPL-Aware Leaf)
RAL:	RPL-Aware Leaf
RH:	Routing Header
RPI:	RPL Packet Information
RTO:	RPL Target Option
RUL:	RPL-Unaware Leaf
SIO:	RPL Sibling Information Option
ULA:	IPv6 Unique Local Address
NSM-VIO:	A Source-Routed Via Information Option, used in Non-Storing Mode P-DAO messages.
TIO:	RPL Transit Information Option
SM-VIO:	A strict Via Information Option, used in Storing Mode P-DAO messages.
VIO:	A Via Information Option; it can be a SM-VIO or an NSM-VIO.

2.4. Domain Terms

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.

Projected DAO: A DAO message used to install a P-Route.

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.

Track: A networking graph that can be followed to transport packets

with equivalent treatment; as opposed to the definition of a path above, a Track 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.
This specification builds Tracks that are DODAGs oriented towards a Track Ingress, and the forward direction for packets is East-West 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.

TrackID: A RPL Local InstanceID that identifies a Track using the namespace owned ny the Track Ingress. The TrackID is associated with the IPv6 Address of the Track Ingress that is used as

DODAGID, and together they form a unique identification of the Track (see the definition of DODAGID in section 2 of [RPL].

Serial Track: A Track that has only one path.

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

subTrack: A Track within a Track. As the Non-Storing Mode Via Information option (NSM-VIO) can only signal a loose sequence of nodes, it takes a number of them to signal a complex Track. Each NSM-VIO for the same TrackId but a different Segment ID signals a different subTracks that the Track Ingress adds to the topology.

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

Track Segment: A serial path formed by a strict sequence of nodes, along which a P-Route is installed. With this specification, a Segment is typically installed by the Root of the main DODAG using Storing Mode P-DAO messages. A Segment used as the topological edge of a Track. Since this specification builds only DODAGs, all Segments are oriented from Ingress (East) to Egress (West), as opposed to the general RAW model, which allows North/South Segments that can be bidirectional.

3. Context and Goal

3.1. RPL Applicability

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

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

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

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

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

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

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

3.2. RPL Routing Modes

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

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

In Storing Mode, the routing information percolates upwards, and each node maintains the routes to the subDAG of its descendants down the DODAG. The maintenance is lazy, either reactive upon traffic or as a slow background process. Packets flow via the common parent and the routing stretch is reduced vs. Non-Storing, for a better P2P connectivity, while the internet connectivity is restored more slowly, time for the DV operation to operate hop-by-hop.

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

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

3.3. On Tracks

A Track is typically a collection of parallel loose source routed sequences of nodes from Ingress to Egress, forming so-called Track Legs, that are joined with strict Segments of other nodes. The Legs are expressed in RPL Non-Storing Modes and require an encapsulation to add a Source Route Header, whereas the Segments are expressed in Storing Mode.

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

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

The concept of a Track was introduced in the "6TiSCH Architecture" [6TiSCH-ARCHI], as a collection of potential paths that leverage redundant forwarding solutions along the way. With this specification, a Track forms DODAG that is directed towards a Track Ingress. 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.

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.

A Track is normally associated with a Local RPL Instance which RPLInstanceID is used as the TrackID, more in Section 6.2. A Track Leg may also be used as a subTrack that extends the RPL main DODAG. In that case, the TrackID is set to the global RPLInstanceID of the main DODAG, which suffices to identify the routing topology. As opposed to local RPL instances, the Track Ingress that encapsulates the packets over a subtrack is not Root, and that the source address of the encapsulated packet is not used to determine the Track.

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 to be used as source to encapsulated packets along the Track is signaled in the DODAGID field of the Projected DAO Base Object; that field is elided in the case of the RPL Main DODAG, see Figure 3.

3.4. Serial Track Signaling

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

Before diving deeper into Track Legs and Segments signaling and operation, this section provides examples of what how route projection works through variations of a simple example. In this simple example we show host routes though RPL Targets can be prefixes. Say we want to build a Serial Track from node A to E in Figure 1, so A can route packets to E's neighbors F and G along A, B, C, D and E as opposed to via the Root:

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

Figure 1: Reference Track

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

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 ommitted if P-DAO 1 and 2 signal F and G as Targets.

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

3.4.1. Using Storing Mode Segments

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

3.4.1.1. 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 succesfully acknowledged, Storing Mode P-DAO 2 is sent to C, as follows:

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

Table 1: P-DAO Messages

As a result the RIBs are set as follows:

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

D	E	P-DAO 1	Neighbor	(A, 129)
"	F, G	P-DAO 1	E	(A, 129)
C	D	P-DAO 1	Neighbor	(A, 129)
"	F, G	P-DAO 1	D	(A, 129)
B	C	P-DAO 2	Neighbor	(A, 129)
"	F, G	P-DAO 2	C	(A, 129)
A	B	P-DAO 2	Neighbor	(A, 129)
"	F, G	P-DAO 2	B	(A, 129)

Table 2: RIB setting

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

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

Table 3: Packet Header Settings

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

* From P-DAO 2: A forwards to B and B forwards to C.

* From P-DAO 1: C forwards to D and D forwards to E.

* From Neighbor Cache Entry: C delivers the packet to F.

3.4.1.2. 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 6.6.

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

* P-DAO 1 signals C==>D==>E-to-E

* P-DAO 2 signals A==>B==>C-to-E

* P-DAO 3 signals F and G via the A-->E-to-F,G Track

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

	P-DAO 1 to E	P-DAO 2 to C	P-DAO 3 to A
Mode	Storing	Storing	Non-Storing
Track Ingress	A	A	A
(DODAGID, TrackID)	(A, 129)	(A, 129)	(A, 129)
SegmentID	1	2	3

VIO	C, D, E	A, B, C	E
Targets	E	E	F, G

Table 4: P-DAO Messages

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

As a result the RIBs are set as follows:

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

Table 5: RIB setting

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

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

Table 6: Packet Header Settings

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

* From P-DAO 3: A encapsulates the packet the Track signaled by P-DAO 3, with the outer header above. Now the packet destination is E.

* From P-DAO 2: A forwards to B and B forwards to C.

* From P-DAO 1: C forwards to D and D forwards to E; E decapsulates the packet.

* From Neighbor Cache Entry: C delivers packets to F or G.

3.4.1.3. Segment Routing

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

* P-DAO 1 signals C==>D==>E-to-E

* P-DAO 2 signals A==>B-to-B,C

* P-DAO 3 signals F and G via the A-->C-->E-to-F,G Track

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

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

Table 7: P-DAO Messages

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

As a result the RIBs are set as follows:

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

Table 8: RIB setting

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

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

Table 9: Packet Header Settings

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

* From P-DAO 3: A encapsulates the packet the Track signaled by P-DAO 3, with the outer header above. Now the destination in the IPv6 Header is C, and a SRH signals the final destination is E.

* From P-DAO 2: A forwards to B and B forwards to C.

* From P-DAO 3: C processes the SRH and sets the destination in the IPv6 Header to E.

* From P-DAO 1: C forwards to D and D forwards to E; E decapsulates the packet.

* From the Neighbor Cache Entry: C delivers packets to F or G.

3.4.2. Using Non-Storing Mode joining Tracks

In this formulation:

* P-DAO 1 signals C==>D==>E-to-F,G

* P-DAO 2 signals A==>B==>C-to-C,F,G

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

3.4.2.1. Stitched Tracks

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

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

Table 10: P-DAO Messages

As a result the RIBs are set as follows:

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

A	B	ND	Neighbor	Any
"	C, E, F, G	P-DAO 2	B, C	(A, 131)

Table 11: RIB setting

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

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

Table 12: Packet Header Settings between C and E

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

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

* From the SRH: Packets forwarded by B have source A, destination C, a consumed SRH, and a RPI indicating a TrackId of 131 from A's namespace. C decapsulates.

* From P-DAO 1: C encapsulates the packet with destination of F in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E

decapsulates.

3.4.2.2. External routes

In this formulation:

* P-DAO 1 signals C==>D==>E-to-E

* P-DAO 2 signals A==>B==>C-to-C,E

* P-DAO 3 signals F and G via the A-->E-to-F,G Track

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

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

Table 13: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 14: RIB setting

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

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

Table 15: Packet Header Settings

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

* From P-DAO 3: A encapsulates the packet with destination of F in the Track signaled by P-DAO 3. The outer header has source A, destination E, and a RPI indicating a TrackId of 141 from A's namespace. This recurses with:

* From P-DAO 2: A encapsulates the packet with destination of E in the Track signaled by P-DAO 2. The outer header has source A, destination B, an SRH that indicates C as the next loose hop, and a RPI indicating a TrackId of 129 from A's namespace.

* From the SRH: Packets forwarded by B have source A, destination C , a consumed SRH, and a RPI indicating a TrackId of 129 from A's namespace. C decapsulates.

* From P-DAO 1: C encapsulates the packet with destination of E in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E decapsulates.

3.4.2.3. Segment Routing

In this formulation:

* P-DAO 1 signals C==>D==>E-to-E

* P-DAO 2 signals A==>B-to-C

* P-DAO 3 signals F and G via the A-->C-->E-to-F,G Track

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

	P-DAO 1 to C	P-DAO 2 to A	P-DAO 3 to A
Mode	Non-Storing	Non-Storing	Non-Storing

Track Ingress	C	A	A
(DODAGID, TrackID)	(C, 131)	(A, 129)	(A, 141)
SegmentID	1	1	1
VIO	D, E	B	C, E
Targets		C	F, G

Table 16: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 17: RIB setting

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

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

Table 18: Packet Header Settings

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

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

Table 19: Packet Header Settings

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

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

Table 20: Packet Header Settings

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

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

* From P-DAO 2: A encapsulates the packet with destination of C in the Track signaled by P-DAO 2. The outer header has source A, destination B, and a RPI indicating a TrackId of 129 from A's namespace. B decapsulates forwards to C based on a sibling connected route.

* From the SRH: C consumes the SRH and makes the destination E.

* From P-DAO 1: C encapsulates the packet with destination of E in the Track signaled by P-DAO 1. The outer header has source C, destination D, an SRH that indicates E as the next loose hop, and a RPI indicating a TrackId of 131 from C's namespace. E decapsulates.

3.5. Complex Tracks

To increase the reliability of the P2P transmission, this specification enables to build a collection of Legs between the same Ingress and Egress Nodes and combine them with the same TrackID, as shown in Figure 2. Legs may cross at loose hops edges or remain parallel.

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

```
           CPF               CPF          CPF                 CPF

                          Southbound API

      _-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-
    _-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-._-

                         +----------+
                         | RPL Root |
                         +----------+
                           (      )
                 (                                  )
           (              DODAG                              )
             (                                           )
     (                                                         )
                                                                     )
     <-    Leg 1            B,                            E ->
     <--- Segment 1 A,B ---> <------- Segment 2 C,D,E ------->

                FWD  --z  Relay --z   FWD  --z   FWD          Target 1
            z-- Node  z--  Node  z--  Node  z--  Node --z     /
         --z    (A)        (B) \      (C)        (D)  z--    /
   Track                        \                       Track
   Ingress                    Segment 5                 Egress - Tgt 2
     (I)                           \                     (E)
         --z                        \                 z--    \
          z-- FWD   --z  FWD  --z  Relay --z  FWD  --z        \
              Node   z-- Node  z-- Node   z-- Node            Target 3
              (F)        (G)       (H)        (J)

     <------ Segment 3 F,G,H ------> <---- Segment 4 J,E ---->
     <-      Leg 2                  H,                    E ->

     <--- Segment 1 A,B ---> <- S5-> <---- Segment 4 J,E ---->
     <-      Leg 3          B,      H,                    E ->
```

```
                                                                     )
      (
                (                                     )
```

Figure 2: Segments and Tracks

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

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

DetNet Forwarding Nodes only understand the simple 1-to-1 forwarding sublayer transport operation along a segment whereas the more sophisticated Relay nodes can also provide service sublayer functions such as Replication and Elimination. One possible mapping between DetNet and this specification is to signal the Relay Nodes as the hops of a Leg and the forwarding Nodes as the hops in a Segment that join the Relay nodes as illustrated in Figure 2.

3.6. Scope and Expectations

This specification expects that the RPL Main DODAG is operated in RPL Non-Storing Mode to sustain the exchanges with the Root. Based on its comprehensive knowledge of the parent-child relationship, the Root can form an abstracted view of the whole DODAG topology. This 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.

The "6TiSCH Architecture" [6TiSCH-ARCHI] 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. 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 [RFC5551]. 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 expects a single PCE with a full view of the network. Distributing the PCE function for a large network is out of scope. This specification uses the RPL Root as a proxy to the PCE. The PCE may be collocated with the Root, or may reside in an external Controller. In that case, the protocol between the Root and the PCE is out of scope and abstracted by / mapped to RPL inside the DODAG; one possibility is for the Root to transmit the RPL DAOs with the SIOs that detail the parent/child and sibling information.

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

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

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

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

4. Extending existing RFCs

4.1. Extending RFC 6550

This specification extends RPL [RPL] to enable the Root to install East-West routes inside a Main DODAG that is operated as non-Storing Mode. A Projected DAO (P-DAO) message (see Section 4.1.1) contains a new Via Information Option that installs a strict or a loose sequence of hops to form respectively a Track Segment or a Track Leg. A new P-DAO Request (PDR) message (see Section 5.1) enables a Track Ingress to request the Track from the Root for which it is the Root and it owns the address that serves as TrackID, as well as the associated namespace from which it selects the TrackID. In the context of this specification, the installed route appears as a more specific route to the Track Targets, and the Track Ingress routes the packets towards the Targets via the Track using the longest match as usual.

To ensure that the PDR and P-DAO messages can flow at most times, it is RECOMMENDED that the nodes involved in a Track mantain 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 ot 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. 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 [6TiSCH-ARCHI] as a collection of multiple P-Routes.

The P-DAO is signaled with a new "Projected DAO" (P) flag, see Figure 3. The 'P' flag is encoded in bit position 2 (to be confirmed by IANA) of the Flags field in the DAO Base Object. The Root MUST set it to 1 in a Projected DAO message. Otherwise it MUST be set to 0. It is set to 0 in Legacy implementations as specified respectively in Sections 20.11 and 6.4 of [RPL]

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

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    TrackID    |K|D|P|  Flags  |    Reserved   | DAOSequence   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     +                                                               +
     |                 DODAGID field set to the                      |
     +               IPv6 Address of the Track Ingress               +
     |              used to source encapsulated packets              |
     +                                                               +
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Option(s)...
     +-+-+-+-+-+-+-+-+
```

Figure 3: Projected DAO Base Object

New fields:

TrackID: The local or global RPLInstanceID of the DODAG that serves as Track, more in Section 6.2

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.

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. Options may be factorized; multiple RTOs may be present to signal a collection of children that can be reached via the parent(s) indicated in the TIO(s) that follows the RTOs. This specification generalizes the case of a parent that can be used to reach a child with that of a whole Track through which children and siblings of the Track Egress are reachable.

4.1.2. Via Information Option

New CMOs called the Via Information Options (VIO) are introduced for use in P-DAO messages as a multihop alternative to the TIO, more in Section 5.3. One VIO is the stateful Storing Mode VIO (SM-VIO); an SM-VIO installs a strict hop-by-hop P-Route called a Track Segment. The other is the Non-Storing Mode VIO (NSM-VIO); the NSM-VIO installs a loose source-routed P-Route called a Track Leg at the Track Ingress, which uses that state to encapsulate a packet IPv6_in_IPv6 with a new Routing Header (RH) to the Track Egress, more in Section 6.6.

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 6.3.2 and Section 6.3.3 respectively for more.

4.1.3. Sibling Information Option

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

4.1.4. P-DAO Request

Two new RPL Control Messages are also introduced, to enable a RPL-Aware Node to request the establishment of a Track between self 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.

4.1.5. Extending 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 extends [RPL] to create a new flag that signals that a packet is forwarded along a P-Route.

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

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

4.2. 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 modifies the RPL Option to encode the 'P' flag as follows:

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

Figure 4: Extended 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 4.1.5.

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 introduces a new 6LoRH, the P-RPI-6LoRH that can be used in either Elective or Critical 6LoRH form, see Table 21 and Table 22 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:

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|E| Length  |  6LoRH Type   | RPLInstanceID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: 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 3.3 and Section 6.2 .

Section 6.7 details how a a Track Ingress leverages the P-RPI-6LoRH Header as part of the encapsulation of a packet to place it into a Track.

5. New RPL Control Messages and Options

5.1. New P-DAO Request Control Message

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

The Root may use an asynchronous PDR-ACK with an negative status to indicate that the Track was terminated before its time. A status of

"Transient Failure" (see Section 10.9) is an indication that the PDR may be retried after a reasonable time that depends on the deployment. Other negative status values indicate a permanent error; the tentative must be abandoned until a corrective action is taken at the application layer or through network management.

The source IPv6 address of the PDR signals the Track Ingress to-be of the requested Track, and the TrackID is indicated in the message itself. One and only one RPL Target Option MUST be present in the message. The RTO signals the Track Egress, more in Section 6.1.

The RPL Control Code for the PDR is 0x09, to be confirmed by IANA. The format of PDR Base Object 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    TrackID    |K|R|   Flags   |  ReqLifetime  | PDRSequence   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Option(s)...
   +-+-+-+-+-+-+-+-+
```

Figure 6: 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 3.3 and Section 6.2. 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 initialized to zero by the sender and MUST be ignored by the receiver

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

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

   PDRSequence: 8-bit wrapping sequence number, obeying the operation
      in section 7.2 of [RPL]. The PDRSequence is used to correlate a
      PDR-ACK message with the PDR message that triggered it. It is
      incremented at each PDR message and echoed in the PDR-ACK by the
      Root.

5.2. New PDR-ACK Control Message

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

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    TrackID    |      Flags    | Track Lifetime|  PDRSequence  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | PDR-ACK Status|                Reserved                       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Option(s)...
     +-+-+-+-+-+-+-+-+
```

Figure 7: New PDR-ACK Control Message Format

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

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

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

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

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

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|E|R|  Value    |
+-+-+-+-+-+-+-+-+
```

Figure 8: 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 27 and Table 28.

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

5.3. Via Information Options

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

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

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

In the case of a SM-VIO, or if [RFC8138] is not used in the data packets, then the Root MUST use only one SRH-6LoRH per Via Information Option, and the compression is the same forall the addresses, as shown in Figure 9, for simplicity.

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

The format of the Via Information Options is as follows:

```
        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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Type        | Option Length |     Flags     |   P-RouteID   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Segm. Sequence | Seg. Lifetime |        SRH-6LoRH head         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .           Via Address 1 (compressed by RFC 8138)              .
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .                              ....                             .
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .           Via Address n (compressed by RFC 8138)              .
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .              Additional SRH-6LoRH Header(s)                   .
    |                                                               |
    .                              ....                             .
```

Figure 9: VIO format (uncompressed form)

Option Type: 0x0E for SM-VIO, 0x0F for NSM-VIO (to be confirmed by IANA), see =Table 25

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.4. Sibling Information Option

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

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

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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Type        | Option Length |D| Flags |Comp.|     Opaque    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             Step of Rank      |            Reserved           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    .                                                               .
    .       Sibling DODAGID (if the D flag not set)                .
    .                                                               .
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    .                                                               .
    .                        Sibling Address                        .
    .                                                               .
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Sibling Information Option Format

Option Type: 0x10 for SIO (to be confirmed by IANA), see =Table 25

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.

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

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

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

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

Step of Rank: 16-bit unsigned integer. This is the Step of Rank [RPL] as computed by the Objective Function between this node and the sibling.

Reserved: The Reserved field MUST 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. Requesting a Track

This specification introduces the PDR message, used by an LLN node to request the formation of a new Track for which this node is Ingress. Note that the namespace for the TrackID is owned by the Ingress node, and in the absence of a PDR, there must be some procedure for the Root to assign TrackIDs in that namespace while avoiding collisions, more in Section 6.2.

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

The Root designs the Track as it sees best, and updates / changes the Segments overtime to serve the Track as needed. There is no notification to the requesting node when those changes happen. The Segment Lifetime in the P-DAO messages does not need to be aligned to the Requested Lifetime in the PDR, or between P-DAO messages for different Segments. The Root may use shorter lifetimes for the Segments and renew them faster than the Track is, or longer lifetimes in which case it will need to tear down the Segments if the Track is not renewed.

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

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

6.2. Identifying a Track

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

This draft leverages the RPL Instance model as follows:

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

  To achieve this, the Root MAY install a Segment along a path down the main Non-Storing Mode DODAG. This enables a loose source routing and reduces the size of the Routing Header, see Appendix A.1. The Root MAY also install a Track Leg across the Main DODAG to complement the routing topology.

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

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

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

  The P-DAO Base Object MUST indicate the tuple (DODAGID, TrackID) that identifies the Track as shown in Figure 3, and the P-RouteID that identifies the P-Route MUST be signaled in the VIO as shown in Figure 9.

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

  particular deployment where PDR are not used, the namespace can be delegated to the main Root, which can assign the TrackIDs for the Tracks it creates without collision.

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

6.3. Installing a Track

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

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

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

* Longest match MUST be the prime comparison for routing.

* In case of equal length match, the route along the Track MUST be preferred vs. the one along the Main DODAG.

* There SHOULD NOT be 2 different Tracks leading to the same Target from same Ingress node, unless there's a policy for selecting which packets use which Track; such policy is out of scope.

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

6.3.1. Signaling a Projected Route

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

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

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

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

In Non-Storing Mode, as discussed in Section 6.3.3, 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 6.3.2. In both cases the Track Ingress is the owner of the Track, and it generates the P-DAO-ACK when the installation is successful.

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

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

A node that processes a VIO MAY verify whether one of these conditions happen, and when so, it MUST ignore the P-DAO and reject it with a RPL Rejection Status of "Error in VIO" in the DAO-ACK, see Section 10.14.

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

6.3.2. Installing a Track Segment with a Storing Mode P-Route

As illustrated in Figure 11, a Storing Mode P-DAO installs a route along the Segment signaled by the SM-VIO towards the Targets indicated in the Target Options. The Segment is to be included in a DODAG indicated by the P-DAO Base Object, that may be the one formed by the RPL Main DODAG, or a Track associated with a local RPL Instance.

```
           ------+---------
                 |          Internet
                 |
              +-----+
              |     | Border router
              |     |  (RPL Root)
              +-----+                      |     ^                   |
                 |                         | DAO | ACK               |
           o    o   o    o                 |     |                   |
       o o   o  o   o  o  o o   o          |  ^       | Projected    .
      o  o o  o o    o   o   o  o  o       |  | DAO   | Route        .
      o   o    o  o     o  o    o  o  o    | ^        |              .
     o  o   o  o   o         o   o o       v | DAO    v              .
     o          o   LLN   o   o     o                                |
         o o   o        o     o              Loose Source Route Path |
      o       o      o    o                                          v
```

Figure 11: Projecting a route

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

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

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

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

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

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

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

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

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

As illustrated in Figure 12, a Non-Storing Mode P-DAO installs a source-routed path within the Track indicated by the P-DAO Base Object, towards the Targets indicated in the Target Options. The source-routed path requires a Source-Routing header which implies an

IP-in-IP encapsulation to add the SRH to an existing packet. It is sent to the Track Ingress which creates a tunnel associated with the Track, and connected routes over the tunnel to the Targets in the RTO. The tunnel encapsulation MUST incorporate a routing header via the list addresses listed in the VIO in the same order. The content of the NSM-VIO starting at the first SRH-6LoRH header MUST be used verbatim by the Track Ingress when it encapsulates a packet to forward it over the Track.

```
              ------+---------
                    |          Internet
                    |
                 +-----+
                 |     | Border router
                 |     |  (RPL Root)
                 +-----+                    |  P  ^ ACK
                    |        Track          | DAO |
              o    o   o  o  Ingress X      V     |   X
          o o   o  o   o  o     o   X   o             X Source
         o  o o  o o    o   o  o    X  o  o           X Routed
         o   o    °  o     o   o   o X     o          X Segment
        o  o   o  o   o  o    o  o     X Egress       X
           o  o  o  o             o    |
                                     Target
          o       o     LLN          o    o
        o          o             o     o
```

Figure 12: Projecting a Non-Storing Route

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

If the next entry in the loose sequence is reachable over a Storing Mode P-Route, it MUST be the Target of a Segment and the Ingress of a next segment, both already setup; the segments are associated with the same Track, which avoids the need of an additional encapsulation. For instance, in Section 3.4.1.3, 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 examplified in Section 3.4.2.3 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.

6.4. Tearing Down a P-Route

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

Since the state about a Leg of a Track is located only the Ingress Node, the Root cleans up the Leg by sending an NSM-VIO to the Ingress indicating the TrackID and the P-RouteID of the Leg being removed, a Segment Lifetime of 0 and a newer Segment Sequence. The SRH-6LoRH with the Via Addresses in the NSM-VIO are not needed and MUST be omitted. 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.

6.5. Maintaining a Track

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

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

6.5.1. Maintaining a Track Segment

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

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

After a reasonable time to enable the deprecated sections to empty, the root tears down the remaining section(s) of the old segments are teared down as described in Section 6.4.

6.5.2. Maintaining a Track Leg

This specification allows to add Legs to a Track by sending a Non-Storing Mode P-DAO to the Ingress associated to the same TrackID, and a new Segment ID. If the Leg is loose, then the Segments that join the hops must be created first. It makes sense to add a new Leg before removing one that is misbehaving, and switch to the new Leg before removing the old.

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

Packets that are in flight over the old version of the Track Leg still follow the old source route path over the old Segments. After a reasonable time to enable the deprecated Segments to empty, the root tears down those Segments as described in Section 6.4.

6.6. Encapsulating and Forwarding Along a Track

When forwarding a packet to a destination for which a router determines that routing happens via a Track for which it is Ingress, the router must encapsulated the packet using IP-in-IP to add the

Source Routing Header with the final destination set to the Track Egress. Though fragmentation is possible in a 6LoWPAN LLN, e.g., using [6LoWPAN], [RFC8930], and/or [RFC8931], it is RECOMMENDED to allow an MTU that is larger than 1280 in the main DODAG and allows for the additional headers while exposing only 1280 to the 6LoWPAN Nodes as indicated by section 4 of [6LoWPAN].

All properties of a Track operations are inherited form the main RPL Instance that is used to install the Track. For instance, the use of compression per [RFC8138] is determined by whether it is used in the RPL Main DODAG, e.g., by setting the "T" flag [TURN-ON_RFC8138] in the RPL configuration option.

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

* In the uncompressed form the source of the packet is the address that this router uses as DODAGID for the Track, the destination is the first Via Address in the NSM-VIO, and the RH is a Source Routing Header (SRH) [RFC6554] that contains the list of the remaining Via Addresses terminating by the Track Egress.

* The preferred alternate in a network where 6LoWPAN Header Compression [RFC6282] is used is to leverage "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch" [RFC8025] to compress the RPL artifacts as indicated in [RFC8138].

   In that case, the source routed header is the exact copy of the (chain of) SRH-6LoRH found in the NSM-VIO, also terminating by the Track Egress. The RPI-6LoRH is appended next, followed by an IP-in-IP 6LoRH Header that indicates the Ingress router in the Encapsulator Address field, see as a similar case Figure 20 of [TURN-ON_RFC8138].

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

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

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

The D flag in the RPLInstanceID MUST be set to 0 to indicate that the source address in the IPv6 header is set ot the DODAGID, more in Section 6.2.

* This draft conforms to the principles of [RFC9008] with regards to packet forwarding and encapsulation along a Track, as follows:

  - With this draft, the Track is a RPL DODAG. From the perspective of that DODAG, the Track Ingress is the Root, the Track Egress is a RPL-Aware 6LR, and neighbors of the Track Egress that can be reached via the Track, but are external to it, are external destinations and treated as RPL-Unaware Leaves (RULs). The encapsulation rules in [RFC9008] apply.

  - If the Track Ingress is the originator of the packet and the Track Egress is the destination of the packet, there is no need for an encapsulation.

  - So the Track Ingress must encapsulate the traffic that it did not originate, and add an RPI.

  A packet that is being routed over the RPL Instance associated to a first Non-Storing Mode Track MAY be placed (encapsulated) in a second Track to cover one loose hop of the first Track as discussed in more details Section 3.4.2.3. On the other hand, a Storing Mode Track must be strict and a packet that it placed in a Storing Mode Track MUST follow that Track till the Track Egress.

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

* In the case of a strict path along a Segment, if the next strict hop is not reachable, the packet is dropped.

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

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

In case of a failure forwarding a packet along a Segment, e.g., the next hop is unreachable, the node that discovers the fault MUST send an ICMPv6 Error message [RFC4443] to the Root, with a new Code "Error in P-Route" (See Section 10.13). 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 ot one or more nodes being removed (See Section 6.5).

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

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

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

6.7. 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 13, representing the case where :

```
+-+ ... -+- ... -+- ... -+-+- ... +-+-+-+ ... +-+-+ ... -+ ... +-...
|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 13: A Packet as Forwarded along the Main DODAG

Since there is no page switch between the encapsulated packet and the encapsulation, the first octet of the compressed packet that acts as page selector is actually removed at encapsulation, so the inner packet used in the descriptions below start with the SRH-6LoRH, and is verbatim the packet represented in Figure 13 from the second octet on.

When encapsulating that inner packet to place it in the Track, the first header that the Ingress appends at the head of the inner packet is an IP-in-IP 6LoRH Header; in that header, the encapsulator address, which maps to the IPv6 source address in the uncompressed form, contains a GUA or ULA IPv6 address of the Ingress node that serves as DODAG ID for the Track, expressed in the compressed form and using the DODAGID of the Main DODAG as compression reference. If the address is compressed to 2 bytes, the resulting value for the Length field shown in Figure 14 is 3, meaning that the SRH-6LoRH as a whole is 5-octets long.

```
  0                   1                   2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-     ...     -+
 |1|0|1| Length  | 6LoRH Type 6  |  Hop Limit    | Track DODAGID |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-     ...     -+
```

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

At the head of the resulting sequence of bytes, the track Ingress then adds the RPI that carries the TrackID as RPLinstanceID as a P-RPI-6LoRH Header, as illustrated in Figure 5, using the TrackID as RPLInstanceID. Combined with the IP-in-IP 6LoRH Header, this allows to identify the Track without ambiguity.

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

```
     0                   1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5  ..          ..          ..
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-    -+-    -+ ... +-    -+
    |1|0|0|  Size   |6LoRH Type 0..4| Hop1 | Hop2 |      | HopN |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-    -+-    -+ ... +-    -+
                                            Where N = Size + 1
```

Figure 15: The SRH 6LoRH Header

The format of the resulting encapsulated packet in [RFC8138] compressed form is illustrated in Figure 16:

```
+-+ ... -+-+-+- ... -+-+-+- ... -+-+-+-+-+- ... +-+-+-+-+-+-+- ...
| Page 1 |  SRH-6LoRH  | P-RPI-6LoRH | IP-in-IP 6LoRH | Inner Packet
+-+ ... -+-+-+- ... -+-+-+- ... -+-+-+-+-+- ... +-+-+-+-+-+-+- ...

 Signals :  Loose Hops :    TrackID  :  Track DODAGID :
```

Figure 16: A Packet as Forwarded along a Track

7. Lesser Constrained Variations

7.1. Storing Mode Main DODAG

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

* In Non-Storing Mode The Root already possesses the DODAG topology, so the additional topological information is reduced to the siblings.

* The downwards routes are updated with unicast messages to the Root, which ensures that the Root can reach back to the LLN nodes after a repair faster than in the case of Storing Mode. Also the Root can control the use of the path diversity in the DODAG to reach to the LLN nodes. For both reasons, Non-Storing Mode provides better capabilities for the Root to maintain the P-Routes.

* When the Main DODAG is operated in Non-Storing Mode, P-Routes enable loose Source Routing, which is only an advantage in that mode. Storing Mode does not use Source Routing Headers, and does not derive the same benefits from this capability.

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

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

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

This section specifies the additions that are needed to support Projected Routes when the Main DODAG is operated in Storing Mode. As long as the RPI can be processed adequately by the dataplane, the changes to this specification are limited to the DAO message. The Track structure, routes and forwarding operations remain the same.

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

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

The DAO message MUST be formed as follows:

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

* The TIO is formed as in Storing Mode, and the Parent Address is not present. The Path Sequence and Path Lifetime fields are aligned with the values used in the Address Registration of the node(s) advertised in the SIO, as explained in Section 9.1. of [RFC9010]. Having similar values in all nodes allows to factorise the TIO for multiple SIOs as done with [RPL].

* The TIO is followed by one or more SIOs that provide an address (ULA or GUA) of the advertised neighbor node.

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

7.2. A Track as a Full DODAG

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

* With this specification, the node in the LLN has no topological awareness, and does not need to maintain dynamic information about the link quality and availability.

* The Root has a complete topological information and statistical metrics that allow it or its PCE to perform a global optimization of all Tracks in its DODAG. Based on that information, the Root computes the Track Leg and predigest the source route paths.

* The node merely selects one of the proposed paths and applies the associated pre-computed routing header in the encapsulation. This alleviates both the complexity of computing a path and the compressed form of the routing header.

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

This section specifies the additions that are needed to turn the Track into a full DODAG and enable the main Root to provide the necessary topological information to the Track Ingress. The expectation is that the metrics that the PSE uses are of an order

other than that of the PCE, because of the difference of time scale between routing and forwarding, mor e in [RAW-ARCHI]. It follows that the PSE will learn the metrics it needs from an alternate source, e.g., OAM frames.

To pass the topological information to the Ingress, the Root uses a P-DAO messages that contains sequences of Target and Transit

Information options that collectively represent the Track, expressed in the same fashion as in classical Non-Storing Mode. The difference as 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 tupples 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 to implement only a portion of this specification to meet a particular use case.

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

Profiles 0 and 1 are REQUIRED by all implementations that may be used in LLNs; this enables to use 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 DOAG). 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) Profi
   le 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) P
   rofile 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 require an encapsulation, in order to insert
   an additional SRH between the loose source routing hops. In that
   case, the Tracks MUST be installed as subTracks of the Main DODAG,
   the main RPL Instance MUST be used as TrackID, and the Ingress
   node that encapsulates is not the Root as it does not own the
   DODAGID.

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

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

Profile 5 Profile 5 Combines Profile 4 with Profile 1 and enables to
   loose source routing between the Ingress and the Egress of the
   Track. As in Profile 1, Storing Mode P-Routes connect the dots in
   the loose source route.

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

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

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

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

9. 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 must be put in place to ensure that the right devices are acting in these roles, so as to avoid threats such as black-holing, (see [RFC7416] section 7). This trust model could be at a minimum based on a Layer-2 Secure joining and the Link-Layer security. This is a generic 6LoWPAN requirement, see Req5.1 in Appendix B.5 of [RFC8505].

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

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 tha can inject P-DAOs can reroute any traffic and deplete critical resources such as spectrum and battery in the LLN rapidly.

10. IANA Considerations

10.1. New Elective 6LoWPAN Routing Header Type

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

Value	Description	Reference
8 (Suggested)	P-RPI-6LoRH	This document

Table 21: New Elective 6LoWPAN Routing Header Type

10.2. New Critical 6LoWPAN Routing Header Type

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

Value	Description	Reference
8 (Suggested)	P-RPI-6LoRH	This document

Table 22: New Critical 6LoWPAN Routing Header Type

10.3. New Subregistry For The RPL Option Flags

IANA is required to create a subregistry for the 8-bit RPL Option Flags field, as detailed in Figure 4, under the "Routing Protocol for Low Power and Lossy Networks (RPL)" registry. The bits are indexed from 0 (leftmost) to 7. Each bit is Tracked with the following qualities:

* Bit number (counting from bit 0 as the most significant bit)

* Indication When Set

* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 26:

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

Table 23: Initial PDR Flags

10.4. New RPL Control Codes

This document extends the IANA Subregistry created by RFC 6550 for RPL Control Codes as indicated in Table 24:

Code	Description	Reference
0x09 (Suggested)	Projected DAO Request (PDR)	This document
0x0A (Suggested)	PDR-ACK	This document

Table 24: New RPL Control Codes

10.5. New RPL Control Message Options

This document extends the IANA Subregistry created by RFC 6550 for RPL Control Message Options as indicated in Table 25:

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

Table 25: RPL Control Message Options

10.6. SubRegistry for the Projected DAO Request Flags

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

* Bit number (counting from bit 0 as the most significant bit)

* Capability description

* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 26:

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

+------------+-----------------------+---------------+

Table 26: Initial PDR Flags

10.7. SubRegistry for the PDR-ACK Flags

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

* Bit number (counting from bit 0 as the most significant bit)

* Capability description

* Reference

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

10.8. Subregistry for the PDR-ACK Acceptance Status Values

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

* Possible values are 6-bit unsigned integers (0..63).

* Registration procedure is "Standards Action" [RFC8126].

* Initial allocation is as indicated in Table 27:

Value	Meaning	Reference
0	Unqualified Acceptance	This document

Table 27: Acceptance values of the PDR-ACK Status

10.9. Subregistry for the PDR-ACK Rejection Status Values

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

* Possible values are 6-bit unsigned integers (0..63).

* Registration procedure is "Standards Action" [RFC8126].

* Initial allocation is as indicated in Table 28:

Value	Meaning	Reference
0	Unqualified Rejection	This document
1	Transient Failure	This document

Table 28: Rejection values of the PDR-ACK Status

10.10. SubRegistry for the Via Information Options Flags

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

* Bit number (counting from bit 0 as the most significant bit)

* Capability description

* Reference

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

10.11. SubRegistry for the Sibling Information Option Flags

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

* Bit number (counting from bit 0 as the most significant bit)

* Capability description

* Reference

Registration procedure is "Standards Action" [RFC8126]. The initial allocation is as indicated in Table 29:

Bit number	Capability description	Reference
0 (Suggested)	"D" flag: Sibling in same DODAG as Self	This document

Table 29: Initial SIO Flags

10.12. New destination Advertisement Object Flag

This document modifies the "destination Advertisement Object (DAO) Flags" registry initially created in Section 20.11 of [RPL] .

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

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

Table 30: New destination Advertisement Object (DAO) Flag

10.13. New ICMPv6 Error Code

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

IANA has defined an ICMPv6 "Code" Fields Registry for ICMPv6 Message Types. ICMPv6 Message Type 1 describes "destination Unreachable" codes. This specification requires that a new code is allocated from

the ICMPv6 Code Fields Registry for ICMPv6 Message Type 1, for "Error in P-Route", with a suggested code value of 8, to be confirmed by IANA.

10.14. New RPL Rejection Status values

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

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

Table 31: Rejection values of the RPL Status

11. 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 document, as well as text suggestions that were incorporated, and to Michael Richardson for his useful recommendations based on his global view of the system. Also special thanks Toerless Eckert for his deep review, with many excellent suggestions that improved the readability and well as the content of the specification.

12. Normative References

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

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

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC5551] Gellens, R., Ed., "Lemonade Notifications Architecture", RFC 5551, DOI 10.17487/RFC5551, August 2009, <https://www.rfc-editor.org/info/rfc5551>.

[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>.

[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>.

[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>.

13. Informative References

[6LoWPAN] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[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>.

[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>.

[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>.

[6TiSCH-ARCHI]
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., Papadopoulos, G. Z., and L. Berger, "Reliable and Available Wireless Architecture/Framework", Work in Progress, Internet-Draft, draft-ietf-raw-architecture-01, 28 July 2021, <https://datatracker.ietf.org/doc/html/

draft-ietf-raw-architecture-01>.

[USE-CASES]
Papadopoulos, G. Z., Thubert, P., Theoleyre, F., and C. J. Bernardos, "RAW use cases", Work in Progress, Internet-Draft, draft-ietf-raw-use-cases-02, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-raw-use-cases-02>.

[TURN-ON_RFC8138]
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>.

[RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <https://www.rfc-editor.org/info/rfc8025>.

[RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <https://www.rfc-editor.org/info/rfc8138>.

[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>.

[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>.

[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>.

   [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>.

   [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>.

   [I-D.irtf-panrg-path-properties]
              Enghardt, T. and C. Krähenbühl, "A Vocabulary of Path
              Properties", Work in Progress, Internet-Draft, draft-irtf-
              panrg-path-properties-03, 9 July 2021,
              <https://datatracker.ietf.org/doc/html/draft-irtf-panrg-
              path-properties-03>.

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

Appendix A. Applications

A.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 17. 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.

```
              ------+---------
                    |          Internet
                    |
                 +-----+
                 |     | Border router
                 |     |  (RPL Root)
                 +-----+                      ^     |        |
                    |                         | DAO | ACK    |
              o    o   o    o                 |     |        | Strict
          o o   o  o   o  o  o o   o          |     |        | Source
         o  o o  o o    o   o   o  o  o       |     |        | Route
         o   o    o  o     o  o    o  o  o    |     |        |
        o  o   o  o   o         o   o o       |     v        v
        o          o             o     o
                          LLN
```

Figure 17: RPL Non-Storing Mode of operation

Based on the parent-children relationships expressed in the Non-Storing DAO messages,the Root possesses topological information about the whole network, though this information is limited to the structure of the DODAG for which it is the destination. A packet that is generated within the domain will always reach the Root, which can then apply a source routing information to reach the destination if the destination is also in the DODAG. Similarly, a packet coming from the outside of the domain for a destination that is expected to be in a RPL domain reaches the Root.

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

As a network gets deep, the size of the source routing header that the Root must add to all the downward packets becomes an issue for nodes that are many hops away. In some use cases, a RPL network forms long lines and a limited amount of well-targeted routing state would allow to make the source routing operation loose as opposed to strict, and save packet size. Limiting the packet size is directly beneficial to the energy budget, but, mostly, it reduces the chances of frame loss and/or packet fragmentation, which is highly detrimental to the LLN operation. Because the capability to store a routing state in every node is limited, the decision of which route is installed where can only be optimized with a global knowledge of the system, a knowledge that the Root or an associated PCE may possess by means that are outside of the scope of this specification.

This specification enables to store a Storing Mode state in intermediate routers, which enables to limit the excursion of the source route headers in deep networks. Once a P-DAO exchange has taken place for a given Target, if the Root operates in non Storing Mode, then it may elide the sequence of routers that is installed in the network from its source route headers to destination that are reachable via that Target, and the source route headers effectively become loose.

A.2. East-West Routes

RPL is optimized for Point-to-Multipoint (P2MP) and Multipoint-to-Point (MP2P), whereby routes are always installed North-South (aka up/down) along the RPL DODAG respectively from and towards the DODAG Root. Peer to Peer (P2P) East-West routes in a RPL network will generally suffer from some elongated (stretched) path versus a direct (optimized) path, since routing between two nodes always happens via a common parent, as illustrated in Figure 18:

```
                  ------+---------
```

```
                       |          Internet
                       |
                    +-----+
                    |     | Border router
                    |     |  (RPL Root)
                    +-----+
                       X
                 ^    v   o    o
             ^ o   o  v   o  o  o o   o
            ^  o o  o v    o   o   o  o  o
            ^   o    o  v     o  o    o  o  o
           S  o   o  o   D         o   o o
           o          o             o     o
                             LLN
```

Figure 18: Routing Stretch between S and D via common parent X along North-South Paths

* 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.

* 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.

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

and that it is even more critical in Non-Storing Mode than it is in Storing Mode, because the routing stretch is wider. For that reason, earlier work at the IETF introduced the "Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks" [RFC6997], which specifies a distributed method for establishing optimized P2P routes. This draft proposes an alternate based on a centralized route computation.

```
                 ------+---------
                       |          Internet
                       |
                    +------+
                    |      | Border router
                    |      |  (RPL Root)
                    +------+
                       |
                 o    o   o    o
             o o   o  o   o  o  o o   o
            o  o o  o o    o   o   o  o  o
            o   o    o  o     o  o    o  o  o
           S>>A>>>B>>C>>>D         o   o o
           o          o             o     o
                             LLN
```

Figure 19: More direct East-West Route between S and D

This specification enables to store source-routed or Storing Mode state in intermediate routers, which enables to limit the stretch of a P2P route and maintain the characteristics within a given SLA. An example of service using this mechanism oculd be a control loop that would be installed in a network that uses classical RPL for asynchronous data collection. In that case, the P2P path may be installed in a different RPL Instance, with a different objective function.

Authors' Addresses

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
06254 Mougins - Sophia Antipolis
France

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

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

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

Matthew Gillmore
Itron, Inc
Building D
2111 N Molter Road
Liberty Lake, 99019
United States

Phone: +1.800.635.5461
Email: matthew.gillmore@itron.com