

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 September 2022.

Copyright Notice

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

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

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

Table of Contents

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

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

1. Introduction

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

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

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

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

2. Terminology

2.1. Requirements Language

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

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

2.2. References

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

2.3. Glossary

THIS RFC often uses the following acronyms:

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

2.4. Domain Terms

This specification uses the following terminology:

2.4.1. Projected Route

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

2.4.2. Projected DAO

A DAO message used to install a P-Route.

2.4.3. Path

Quoting section 1.1.3 of [INT-ARCHI]:

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

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

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

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

2.4.4. Routing Stretch

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

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

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

2.4.5. Track

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

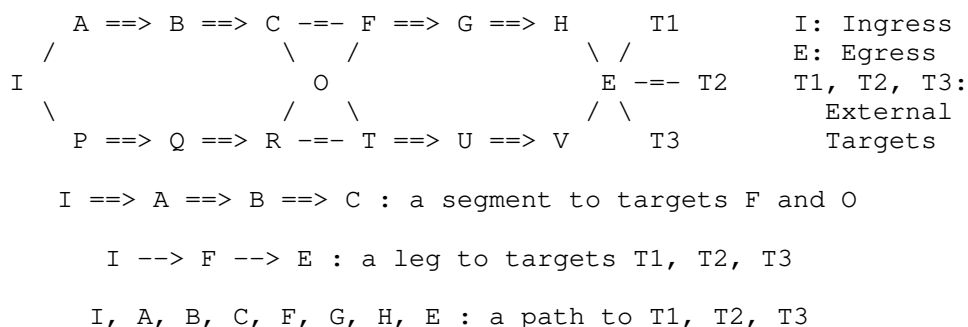


Figure 1: A Track and its Components

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

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

2.4.5.1. TrackID

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

2.4.5.2. Namespace

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

2.4.5.3. Serial Track

A Track that has only one path.

2.4.5.4. Stand-Alone

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

2.4.5.5. Stitching

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

2.4.5.6. Leg

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

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

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

2.4.5.7. subTrack

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

2.4.5.8. Segment

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

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

2.4.5.8.1. Section of a Segment

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

2.4.5.8.2. Segment Routing and SRH

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

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

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

3. Context and Goal

3.1. RPL Applicability

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

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

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

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

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

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

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

3.2. RPL Routing Modes

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

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

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

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

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

3.3. Requirements

3.3.1. Loose Source Routing

A RPL implementation operating in a very constrained LLN typically uses the Non-Storing Mode of Operation as represented in Figure 2. In that mode, a RPL node indicates a parent-child relationship to the Root, using a destination Advertisement Object (DAO) that is unicast from the node directly to the Root, and the Root typically builds a source routed path to a destination down the DODAG by recursively concatenating this information.

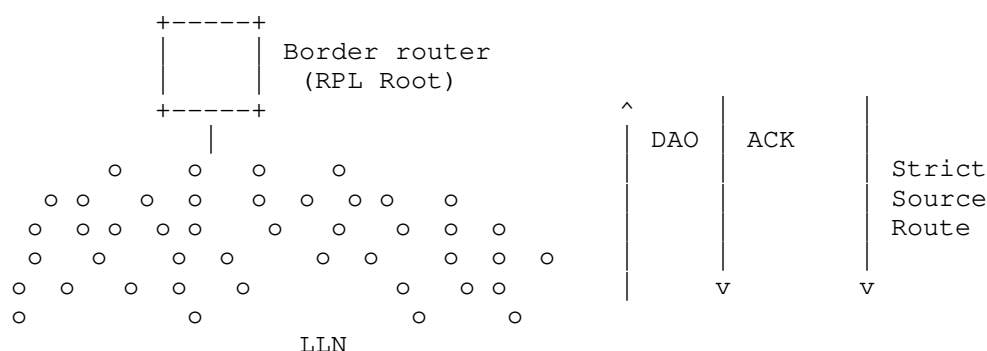


Figure 2: RPL Non-Storing Mode of operation

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

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

allow the source routing operation to be loose as opposed to strict, and save packet size. Because the capability to store a routing state in every node is limited, the decision of which route is installed where can only be optimized with a global knowledge of the system, a knowledge that the Root or an associated PCE may possess by means that are outside of the scope of this specification.

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

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

3.3.2. East-West Routes

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

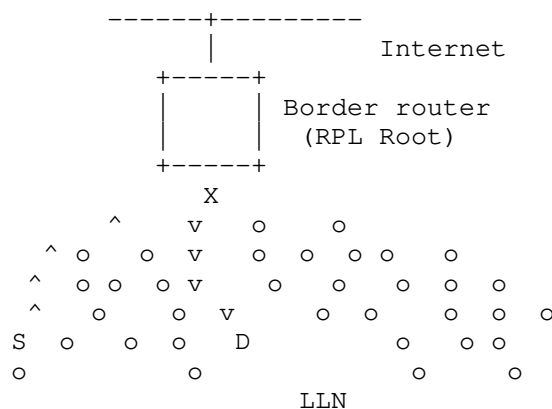


Figure 3: Routing Stretch between S and D via common parent X
along North-South Paths

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

- * In Non-Storing Mode, all packets routed within the DODAG flow all the way up to the Root of the DODAG. If the destination is in the same DODAG, the Root must encapsulate the packet to place an RH that has the strict source route information down the DODAG to the destination. This will be the case even if the destination is relatively close to the source and the Root is relatively far off.
- * In Storing Mode, unless the destination is a child of the source, the packets will follow the default route up the DODAG as well. If the destination is in the same DODAG, they will eventually reach a common parent that has a route to the destination; at worse, the common parent may also be the Root. From that common parent, the packet will follow a path down the DODAG that is optimized for the Objective Function that was used to build the DODAG.

It results that it is often beneficial to enable East-West P2P routes, either if the RPL route presents a stretch from shortest path, or if the new route is engineered with a different objective, and that it is even more critical in Non-Storing Mode than it is in Storing Mode, because the routing stretch is wider. For that reason, earlier work at the IETF introduced the "Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks" [RFC6997], which specifies a distributed method for establishing optimized P2P routes. This draft proposes an alternate based on a centralized route computation.

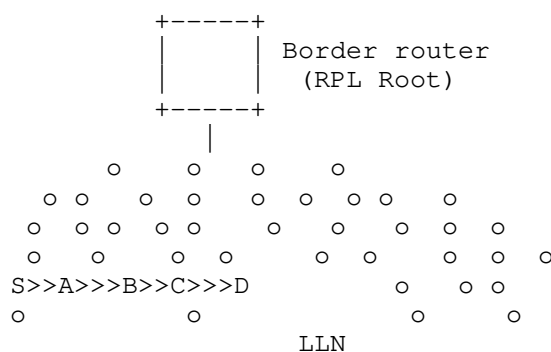


Figure 4: More direct East-West Route between S and D

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

3.4. On Tracks

3.4.1. Building Tracks With RPL

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

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

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

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

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

3.4.2. Tracks and RPL Instances

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

```

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

```

Figure 5: Local RPLInstanceID Encoding

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

3.5. Serial Track Signaling

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

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

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

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

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

```

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

```

Figure 6: Reference Track

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

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

In a general manner the desired outcome is as follows:

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

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

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

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

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

3.5.1. Using Storing Mode Segments

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

3.5.1.1. Stitched Segments

In this formulation:

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

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

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

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

Table 1: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 2: RIB setting

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

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

Table 3: Packet Header Settings

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

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

3.5.1.2. External routes

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

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

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

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

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

Table 4: P-DAO Messages

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

As a result the RIBs are set as follows:

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

Table 5: RIB setting

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

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

Table 6: Packet Header Settings

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

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

3.5.1.3. Segment Routing

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

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

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

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

Table 7: P-DAO Messages

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

As a result the RIBs are set as follows:

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

Table 8: RIB setting

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

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

Table 9: Packet Header Settings

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

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

3.5.2. Using Non-Storing Mode joining Tracks

In this formulation:

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

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

3.5.2.1. Stitched Tracks

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

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

Table 10: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 11: RIB setting

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

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

Table 12: Packet Header Settings between C and E

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

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

3.5.2.2. External routes

In this formulation:

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

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

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

Table 13: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 14: RIB setting

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

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

Table 15: Packet Header Settings

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

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

3.5.2.3. Segment Routing

In this formulation:

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

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

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

Table 16: P-DAO Messages

As a result the RIBs are set as follows:

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

Table 17: RIB setting

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

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

Table 18: Packet Header Settings

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

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

Table 19: Packet Header Settings

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

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

Table 20: Packet Header Settings

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

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

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

3.6. Complex Tracks

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

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

CPF

CPF

CPF

CPF

Southbound API

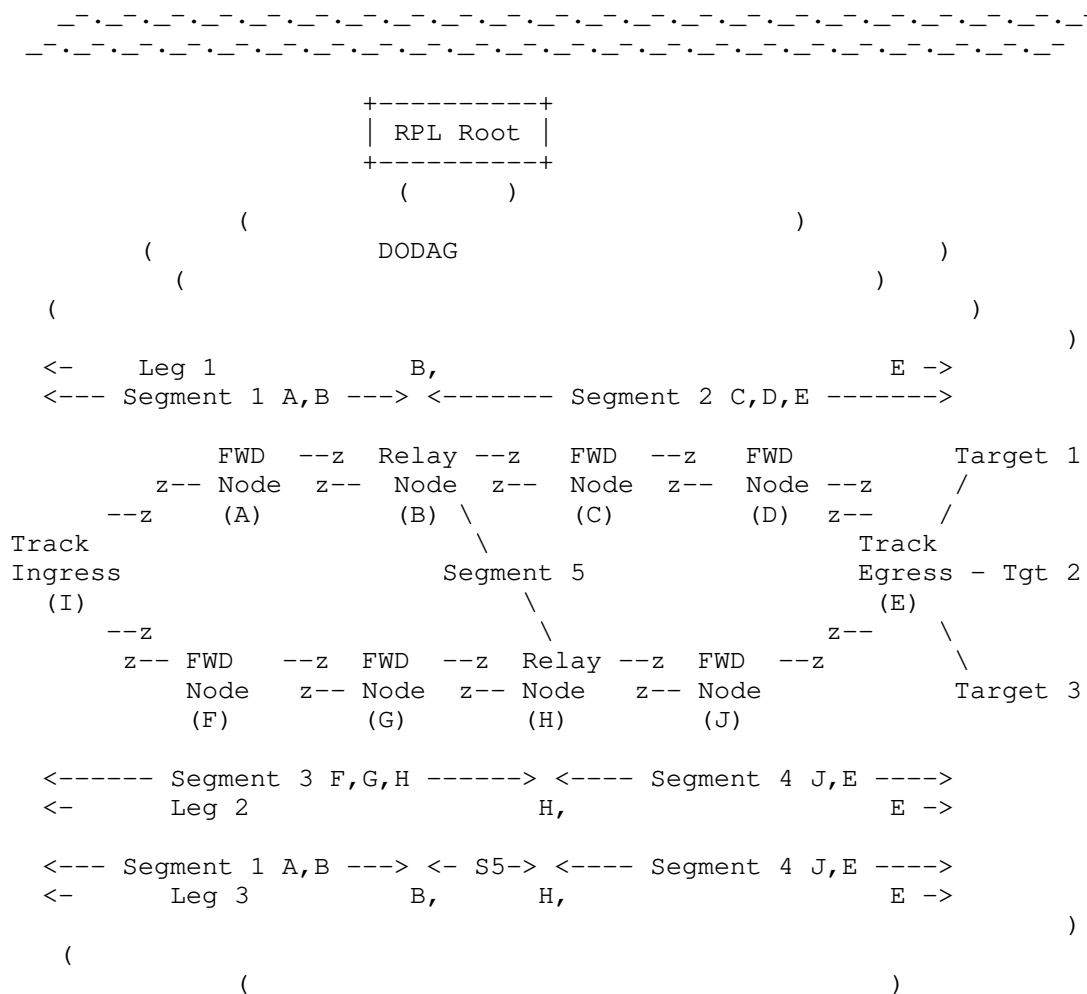


Figure 7: Segments and Tracks

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

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

3.7. Scope and Expectations

3.7.1. External Dependencies

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

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

3.7.2. Positioning vs. Related IETF Standards

3.7.2.1. Extending 6TiSCH

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

3.7.2.2. Mapping to DetNet

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

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

3.7.2.3. Leveraging PCE

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

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

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

3.7.2.4. Providing for RAW

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

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

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

4. Extending existing RFCs

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

4.1. Extending RFC 6550

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

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

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

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

4.1.1. Projected DAO

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

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

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

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

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

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

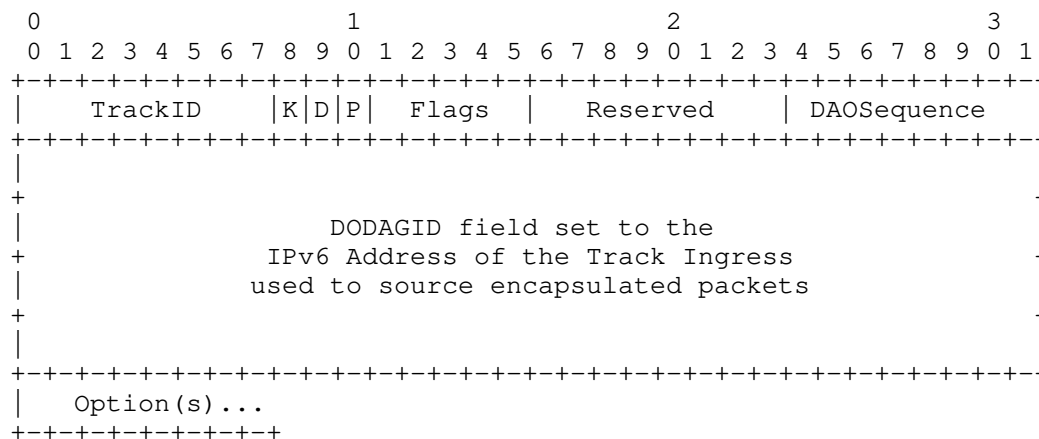


Figure 8: Projected DAO Base Object

New fields:

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

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

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

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

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

4.1.2. Projected DAO-ACK

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

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

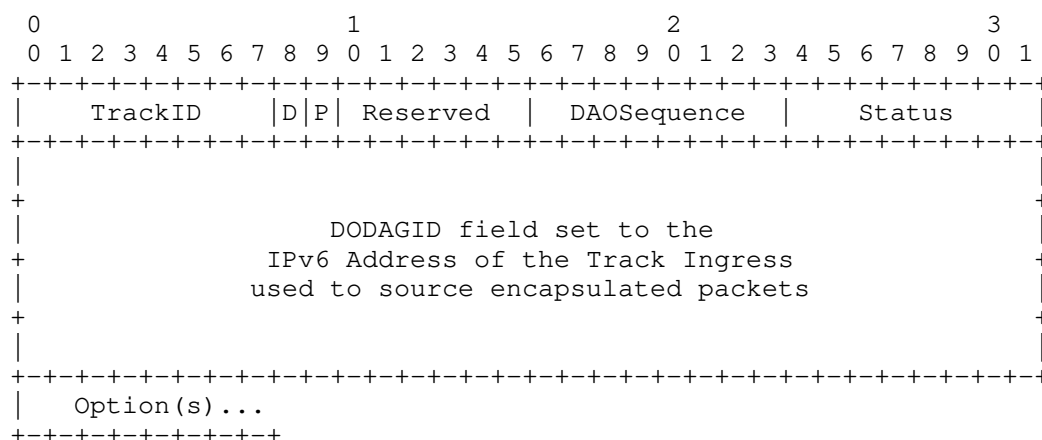


Figure 9: Projected DAO-ACK Base Object

New fields:

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

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

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

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

4.1.3. Via Information Option

THIS RFC Extends the CMO to create new objects called the Via Information Options (VIO). The VIOs are the multihop alternative to the TIO, more in Section 5.3. One VIO is the stateful Storing Mode VIO (SM-VIO); an SM-VIO installs a strict hop-by-hop P-Route called a Track Segment. The other is the Non-Storing Mode VIO (NSM-VIO); the NSM-VIO installs a loose source-routed P-Route called a Track Leg at the Track Ingress, which uses that state to encapsulate a packet IPv6_in_IPv6 with a new Routing Header (RH) to the Track Egress, more in Section 6.7.

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

4.1.4. Sibling Information Option

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

4.1.5. P-DAO Request

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

4.1.6. Amending the RPI

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

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

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

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

4.1.7. Additional Flag in the RPL DODAG Configuration Option

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

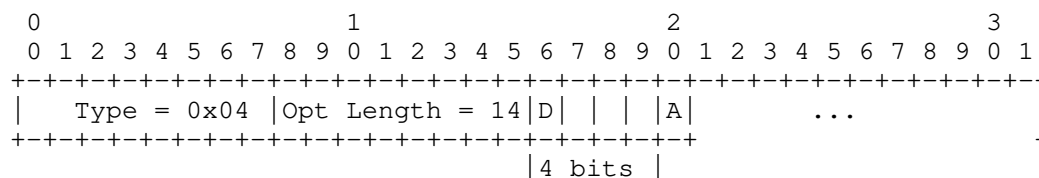


Figure 10: DODAG Configuration Option (Partial View)

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

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

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

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

[RPL] states that:

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

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

4.2. Extending RFC 6553

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

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

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

```

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

```

Figure 11: Amended RPL Option Format

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

Opt Data Len: See [RFC6553]

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

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

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

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

4.3. Extending RFC 8138

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

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

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

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

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

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

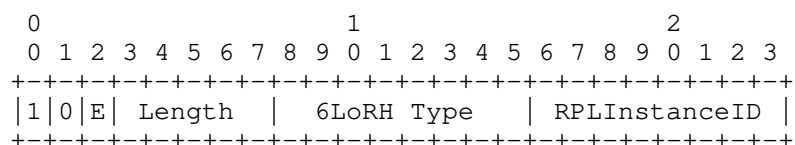


Figure 12: P-RPI-6LoRH Format

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

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

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

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

5. New RPL Control Messages and Options

5.1. New P-DAO Request Control Message

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

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

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

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

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

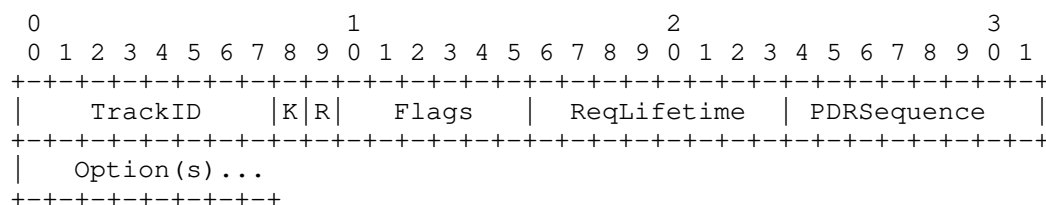


Figure 13: New P-DAO Request Format

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

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

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

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

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

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

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

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

5.2. New PDR-ACK Control Message

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

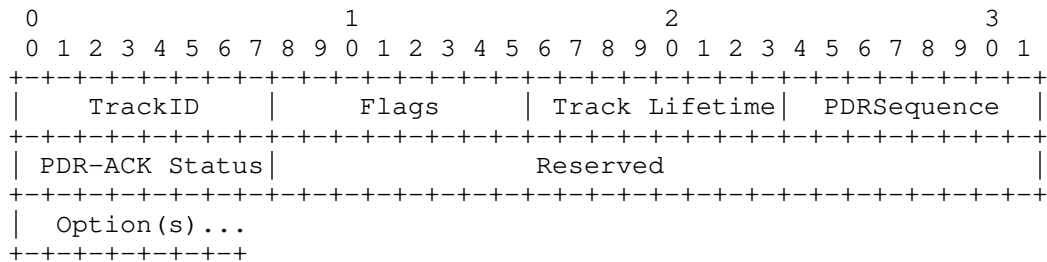


Figure 14: New PDR-ACK Control Message Format

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

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

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

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

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

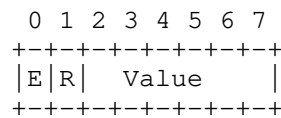


Figure 15: PDR-ACK status Format

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

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

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

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

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

5.3. Via Information Options

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

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

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

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

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

The format of the Via Information Options is as follows:

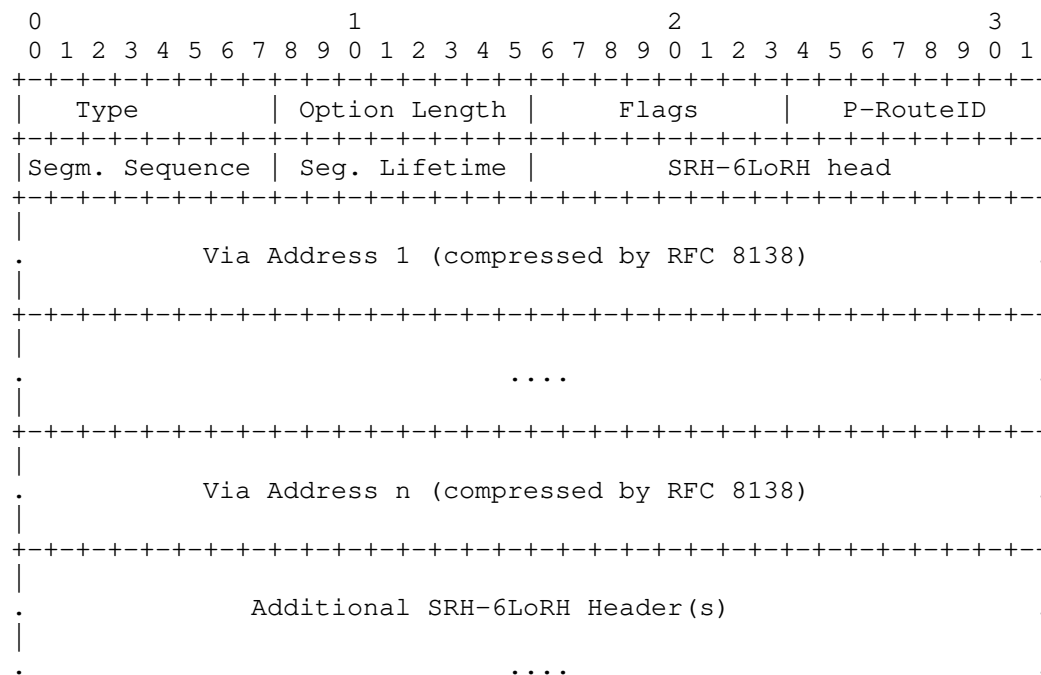


Figure 16: VIO format (uncompressed form)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.4. Sibling Information Option

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

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

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

The format of the SIO is as follows:

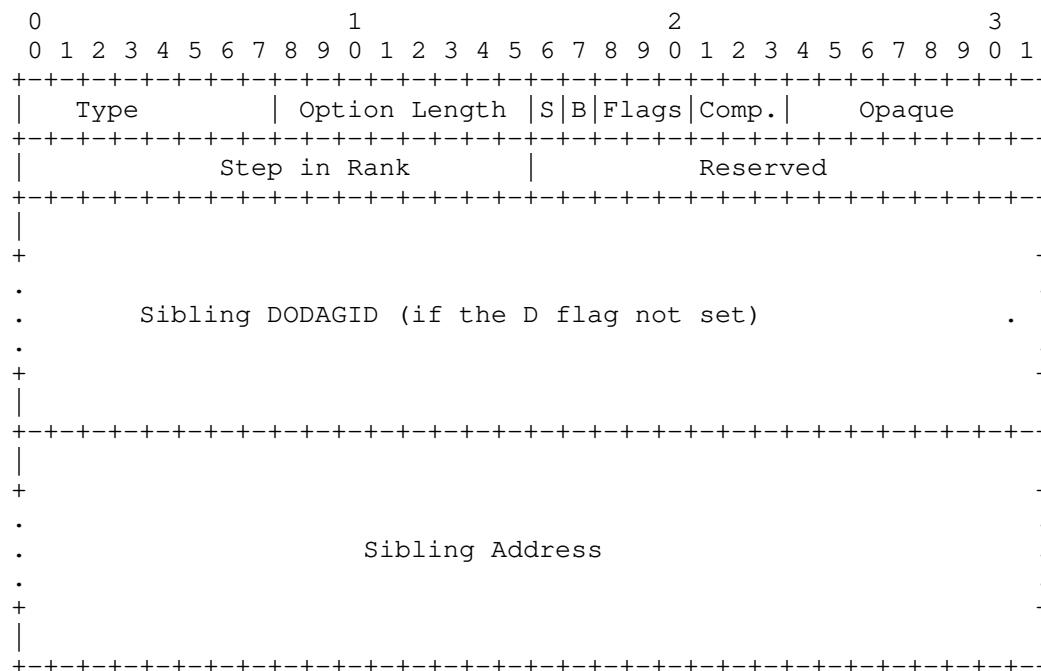


Figure 17: Sibling Information Option Format

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

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

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

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

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

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

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

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

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

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

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

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

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

6. Root Initiated Routing State

6.1. RPL Network Setup

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

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

6.2. Requesting a Track

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

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

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

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

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

6.3. Identifying a Track

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

This draft leverages the RPL Instance model as follows:

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

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

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

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

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

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

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

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

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

6.4. Installing a Track

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

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

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

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

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

6.4.1. Signaling a Projected Route

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

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

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

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

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

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

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

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

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

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

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

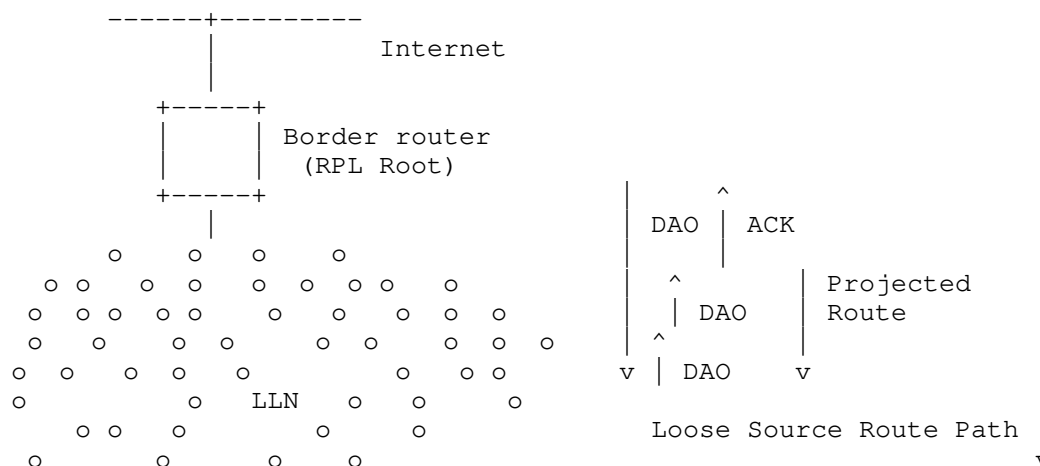


Figure 18: Projecting a route

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

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

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

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

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

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

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

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

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

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

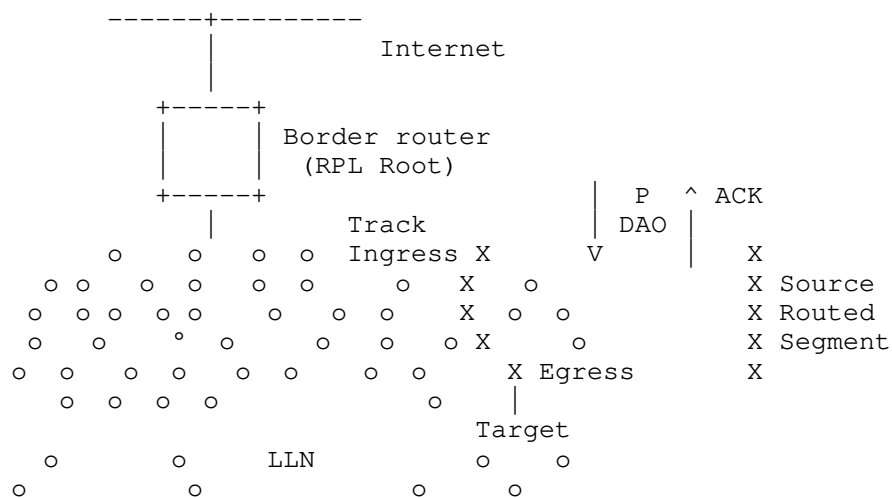


Figure 19: Projecting a Non-Storing Route

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

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

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

6.5. Tearing Down a P-Route

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

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

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

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

6.6. Maintaining a Track

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

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

6.6.1. Maintaining a Track Segment

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

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

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

6.6.2. Maintaining a Track Leg

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

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

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

6.7. Encapsulating and Forwarding Along a Track

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.8. Compression of the RPL Artifacts

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

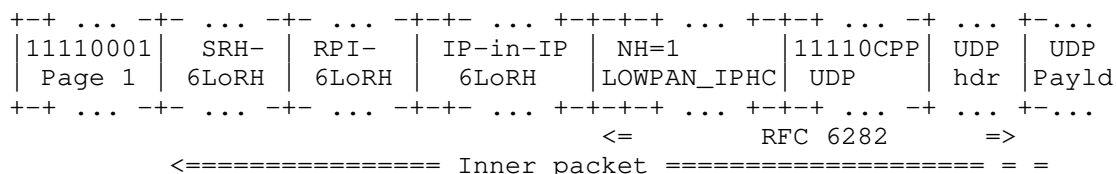


Figure 20: A Packet as Forwarded along the Main DODAG

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

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

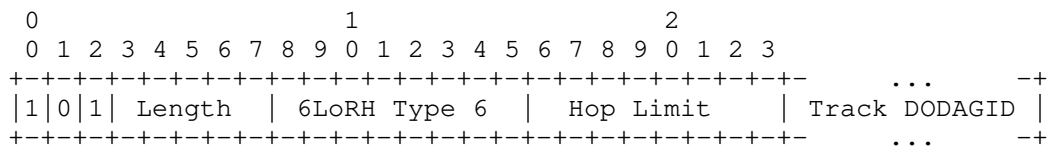


Figure 21: The IP-in-IP 6LoRH Header

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

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

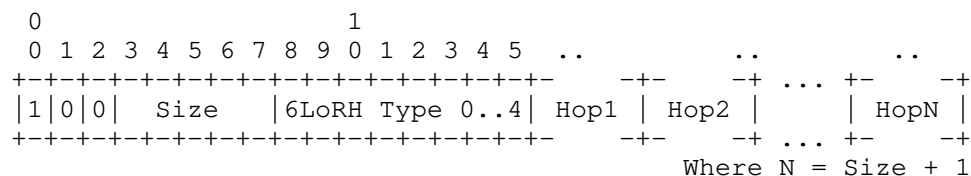
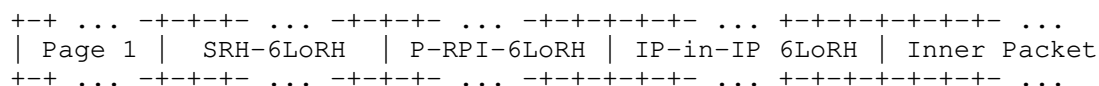


Figure 22: The SRH 6LoRH Header

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



Signals : Loose Hops : TrackID : Track DODAGID :

Figure 23: A Packet as Forwarded along a Track

7. Lesser Constrained Variations

7.1. Storing Mode Main DODAG

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

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

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

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

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

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

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

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

The DAO message MUST be formed as follows:

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

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

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

7.2. A Track as a Full DODAG

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

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

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

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

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

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

8. Profiles

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9. Backwards Compatibility

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

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

10. Security Considerations

It is worth noting that with [RPL], every node in the LLN is RPL-aware and can inject any RPL-based attack in the network. This draft uses messages that are already present in RPL [RPL] with optional secured versions. The same secured versions may be used with this draft, and whatever security is deployed for a given network also applies to the flows in this draft.

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

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

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

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

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

11. IANA Considerations

11.1. RPL DODAG Configuration Option Flag

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

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

Table 21: New DODAG Configuration Option Flag

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

11.2. Elective 6LoWPAN Routing Header Type

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

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

Table 22: New Elective 6LoWPAN Routing Header Type

11.3. Critical 6LoWPAN Routing Header Type

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

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

Table 23: New Critical 6LoWPAN Routing Header Type

11.4. Subregistry For The RPL Option Flags

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

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

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

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

Table 24: Initial PDR Flags

11.5. RPL Control Codes

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

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

Table 25: New RPL Control Codes

11.6. RPL Control Message Options

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

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

Table 26: RPL Control Message Options

11.7. SubRegistry for the Projected DAO Request Flags

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

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

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

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

Table 27: Initial PDR Flags

11.8. SubRegistry for the PDR-ACK Flags

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

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

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

11.9. Subregistry for the PDR-ACK Acceptance Status Values

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

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

Value	Meaning	Reference
0	Unqualified Acceptance	THIS RFC

Table 28: Acceptance values of the PDR-ACK Status

11.10. Subregistry for the PDR-ACK Rejection Status Values

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

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

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

Table 29: Rejection values of the PDR-ACK Status

11.11. SubRegistry for the Via Information Options Flags

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

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

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

11.12. SubRegistry for the Sibling Information Option Flags

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

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

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

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

Table 30: Initial SIO Flags

11.13. Destination Advertisement Object Flag

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

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

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

Table 31: New Destination Advertisement Object (DAO) Flag

11.14. Destination Advertisement Object Acknowledgment Flag

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

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

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

Table 32: New Destination Advertisement Object Acknowledgment Flag

11.15. New ICMPv6 Error Code

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

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

11.16. RPL Rejection Status values

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

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

Table 33: Rejection values of the RPL Status

12. Acknowledgments

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

13. Normative References

[INT-ARCHI]

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

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

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

- [RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RPL] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<https://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <<https://www.rfc-editor.org/info/rfc8138>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

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

14. Informative References

- [6LoWPAN] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks", RFC 6997, DOI 10.17487/RFC6997, August 2013, <<https://www.rfc-editor.org/info/rfc6997>>.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <<https://www.rfc-editor.org/info/rfc7102>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<https://www.rfc-editor.org/info/rfc8025>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

- [RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <<https://www.rfc-editor.org/info/rfc8505>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <<https://www.rfc-editor.org/info/rfc8655>>.
- [RFC8930] Watteyne, T., Ed., Thubert, P., Ed., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multi-Hop IPv6 Network", RFC 8930, DOI 10.17487/RFC8930, November 2020, <<https://www.rfc-editor.org/info/rfc8930>>.
- [RFC8931] Thubert, P., Ed., "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery", RFC 8931, DOI 10.17487/RFC8931, November 2020, <<https://www.rfc-editor.org/info/rfc8931>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.
- [RFC9010] Thubert, P., Ed. and M. Richardson, "Routing for RPL (Routing Protocol for Low-Power and Lossy Networks) Leaves", RFC 9010, DOI 10.17487/RFC9010, April 2021, <<https://www.rfc-editor.org/info/rfc9010>>.
- [RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <<https://www.rfc-editor.org/info/rfc9030>>.
- [RFC9035] Thubert, P., Ed. and L. Zhao, "A Routing Protocol for Low-Power and Lossy Networks (RPL) Destination-Oriented Directed Acyclic Graph (DODAG) Configuration Option for the 6LoWPAN Routing Header", RFC 9035, DOI 10.17487/RFC9035, April 2021, <<https://www.rfc-editor.org/info/rfc9035>>.

[RAW-ARCHI]

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

[USE-CASES]

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

[I-D.kuehlewind-update-tag]

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

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

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

[PCE]

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

Authors' Addresses

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

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

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

ROLL
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2020

R. Jadhav, Ed.
Huawei
P. Thubert
Cisco
R. Sahoo
Z. Cao
Huawei
October 29, 2019

Efficient Route Invalidation
draft-ietf-roll-efficient-npdao-17

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 1, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

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

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

1. Introduction

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

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

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

1.1. Requirements Language and Terminology

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

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

Low Power and Lossy Networks (LLN):

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

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

6LoWPAN Router (6LR):

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

Directed Acyclic Graph (DAG):

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

Destination-Oriented DAG (DODAG):

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

6LoWPAN Border Router (6LBR):

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

Destination Advertisement Object (DAO):

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

DODAG Information Object (DIO):

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

Common Ancestor node

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

No-Path DAO (NPDAO):

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

Destination Cleanup Object (DCO):

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

Regular DAO:

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

Target node:

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

1.2. Current NPDAO messaging

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

For the rest of this document consider the following topology:

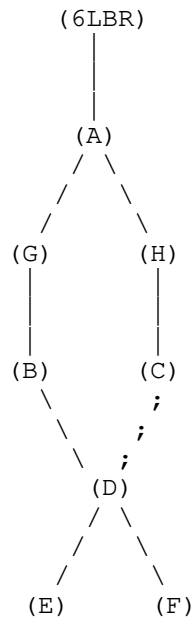


Figure 1: Sample topology

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

1.3. Why Is NPDAO Important?

Nodes in LLNs may be resource constrained. There is limited memory available and routing entry records are one of the primary elements occupying dynamic memory in the nodes. Route invalidation helps 6LR nodes to decide which entries could be discarded to better optimize resource utilization. Thus it becomes necessary to have an efficient route invalidation mechanism. Also note that a single parent switch may result in a "sub-tree" switching from one parent to another. Thus the route invalidation needs to be done on behalf of the sub-tree and not the switching node alone. In the above example, when Node (D) switches parent, the route updates needs to be done for the routing tables entries of (C), (H), (A), (G), and (B) with destination (D), (E) and (F). Without efficient route invalidation, a 6LR may have to hold a lot of stale route entries.

2. Problems with current NPDAO messaging

2.1. Lost NPDAO due to link break to the previous parent

When a node switches its parent, the NPDAO is to be sent to its previous parent and a regular DAO to its new parent. In cases where the node switches its parent because of transient or permanent parent link/node failure then the NPDAO message is bound to fail.

2.2. Invalidate Routes of Dependent Nodes

RPL does not specify how route invalidation will work for dependent nodes rooted at the switching node, resulting in stale routing entries of the dependent nodes. The only way for 6LR to invalidate the route entries for dependent nodes would be to use route lifetime expiry which could be substantially high for LLNs.

In the example topology, when Node (D) switches its parent, Node (D) generates an NPDAO on its behalf. There is no NPDAO generated by the dependent child nodes (E) and (F), through the previous path via (D) to (B) and (G), resulting in stale entries on nodes (B) and (G) for nodes (E) and (F).

2.3. Possible route downtime caused by asynchronous operation of NPDAO and DAO

A switching node may generate both an NPDAO and DAO via two different paths at almost the same time. There is a possibility that an NPDAO generated may invalidate the previous route and the regular DAO sent via the new path gets lost on the way. This may result in route downtime impacting downward traffic for the switching node.

In the example topology, consider Node (D) switches from parent (B) to (C). An NPDAO sent via the previous route may invalidate the previous route whereas there is no way to determine whether the new DAO has successfully updated the route entries on the new path.

3. Requirements for the NPDAO Optimization

3.1. Req#1: Remove messaging dependency on link to the previous parent

When the switching node sends the NPDAO message to the previous parent, it is normal that the link to the previous parent is prone to failure (that's why the node decided to switch). Therefore, it is required that the route invalidation does not depend on the previous link which is prone to failure. The previous link referred here represents the link between the node and its previous parent (from whom the node is now disassociating).

3.2. Req#2: Dependent nodes route invalidation on parent switching

It should be possible to do route invalidation for dependent nodes rooted at the switching node.

3.3. Req#3: Route invalidation should not impact data traffic

While sending the NPDAO and DAO messages, it is possible that the NPDAO successfully invalidates the previous path, while the newly sent DAO gets lost (new path not set up successfully). This will result in downstream unreachability to the node switching paths. Therefore, it is desirable that the route invalidation is synchronized with the DAO to avoid the risk of route downtime.

4. Changes to RPL signaling

4.1. Change in RPL route invalidation semantics

As described in Section 1.2, the NPDAO originates at the node changing to a new parent and traverses upstream towards the root. In order to solve the problems as mentioned in Section 2, the document adds a new proactive route invalidation message called "Destination Cleanup Object" (DCO) that originates at a common ancestor node and flows downstream between the new and old path. The common ancestor node generates a DCO in response to the change in the next-hop on receiving a regular DAO with updated Path Sequence for the target.

The 6LRs in the path for DCO take action such as route invalidation based on the DCO information and subsequently send another DCO with the same information downstream to the next hop. This operation is similar to how the DAOs are handled on intermediate 6LRs in storing MOP in [RFC6550]. Just like DAO in storing MOP, the DCO is sent using link-local unicast source and destination IPv6 address. Unlike DAO, which always travels upstream, the DCO always travels downstream.

In Figure 1, when node D decides to switch the path from B to C, it sends a regular DAO to node C with reachability information containing the address of D as the target and an incremented Path Sequence. Node C will update the routing table based on the reachability information in the DAO and in turn generate another DAO with the same reachability information and forward it to H. Node H also follows the same procedure as Node C and forwards it to node A. When node A receives the regular DAO, it finds that it already has a routing table entry on behalf of the target address of node D. It finds however that the next hop information for reaching node D has changed i.e., node D has decided to change the paths. In this case, Node A which is the common ancestor node for node D along the two

paths (previous and new), should generate a DCO which traverses downwards in the network. Node A handles normal DAO forwarding to 6LBR as required by [RFC6550].

4.2. Transit Information Option changes

Every RPL message is divided into base message fields and additional Options as described in Section 6 of [RFC6550]. The base fields apply to the message as a whole and options are appended to add message/use-case specific attributes. As an example, a DAO message may be attributed by one or more "RPL Target" options which specify the reachability information for the given targets. Similarly, a Transit Information option may be associated with a set of RPL Target options.

This document specifies a change in the Transit Information Option to contain the "Invalidate previous route" (I) flag. This 'I' flag signals the common ancestor node to generate a DCO on behalf of the target node with a RPL Status of 130 indicating that the address has moved. The 'I' flag is carried in the Transit Information Option which augments the reachability information for a given set of RPL Target(s). Transit Information Option with 'I' flag set should be carried in the DAO message when route invalidation is sought for the corresponding target(s).

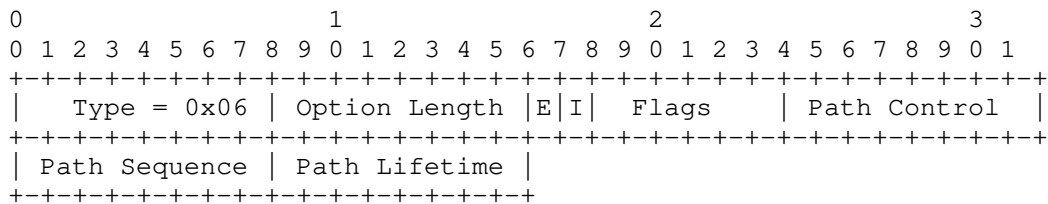


Figure 2: Updated Transit Information Option (New I flag added)

I (Invalidate previous route) flag: The 'I' flag is set by the target node to indicate to the common ancestor node that it wishes to invalidate any previous route between the two paths.

[RFC6550] allows the parent address to be sent in the Transit Information Option depending on the mode of operation. In case of storing mode of operation the field is usually not needed. In case of DCO, the parent address field MUST NOT be included.

The common ancestor node SHOULD generate a DCO message in response to this 'I' flag when it sees that the routing adjacencies have changed for the target. The 'I' flag is intended to give the target node

control over its own route invalidation, serving as a signal to request DCO generation.

4.3. Destination Cleanup Object (DCO)

A new ICMPv6 RPL control message code is defined by this specification and is referred to as "Destination Cleanup Object" (DCO), which is used for proactive cleanup of state and routing information held on behalf of the target node by 6LRs. The DCO message always traverses downstream and cleans up route information and other state information associated with the given target.

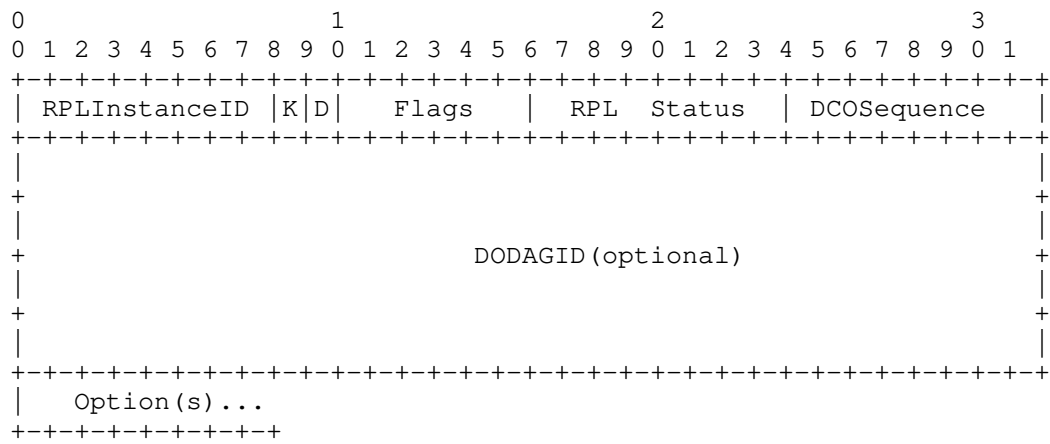


Figure 3: DCO base object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

K: The 'K' flag indicates that the recipient of DCO message is expected to send a DCO-ACK back. If the DCO-ACK is not received even after setting the 'K' flag, an implementation may retry the DCO at a later time. The number of retries are implementation and deployment dependent and are expected to be kept similar with those used in DAO retries in [RFC6550]. Section 4.6.3 specifies the considerations for DCO retry. A node receiving a DCO message without the 'K' flag set MAY respond with a DCO-ACK, especially to report an error condition. An example error condition could be that the node sending the DCO-ACK does not find the routing entry for the indicated target. When the sender does not set the 'K' flag it is an indication that the sender does not expect a response, and the sender SHOULD NOT retry the DCO.

D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: The 6 bits remaining unused in the Flags field are reserved for future use. These bits MUST be initialized to zero by the sender and MUST be ignored by the receiver.

RPL Status: As defined in [RFC6550] and updated in [I-D.ietf-roll-unaware-leaves]. The root or common parent that generates a DCO is authoritative for setting the status information and the information is unchanged as propagated down the DODAG. This document does not specify a differentiated action based on the RPL status.

DCOSequence: 8-bit field incremented at each unique DCO message from a node and echoed in the DCO-ACK message. The initial DCOSequence can be chosen randomly by the node. Section 4.4 explains the handling of the DCOSequence.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present if 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

4.3.1. Secure DCO

A Secure DCO message follows the format in [RFC6550] Figure 7, where the base message format is the DCO message shown in Figure 3.

4.3.2. DCO Options

The DCO message MUST carry at least one RPL Target and the Transit Information Option and MAY carry other valid options. This specification allows for the DCO message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x06 Transit Information
- 0x09 RPL Target Descriptor

Section 6.7 of [RFC6550] defines all the above mentioned options. The DCO carries an RPL Target Option and an associated Transit Information Option with a lifetime of 0x00000000 to indicate a loss of reachability to that Target.

4.3.3. Path Sequence number in the DCO

A DCO message may contain a Path Sequence in the Transit Information Option to identify the freshness of the DCO message. The Path Sequence in the DCO MUST use the same Path Sequence number present in the regular DAO message when the DCO is generated in response to a DAO message. Thus if a DCO is received by a 6LR and subsequently a DAO is received with an old sequence number, then the DAO MUST be ignored. When the DCO is generated in response to a DCO from upstream parent, the Path Sequence MUST be copied from the received DCO.

4.3.4. Destination Cleanup Option Acknowledgment (DCO-ACK)

The DCO-ACK message SHOULD be sent as a unicast packet by a DCO recipient in response to a unicast DCO message with 'K' flag set. If 'K' flag is not set then the receiver of the DCO message MAY send a DCO-ACK, especially to report an error condition.

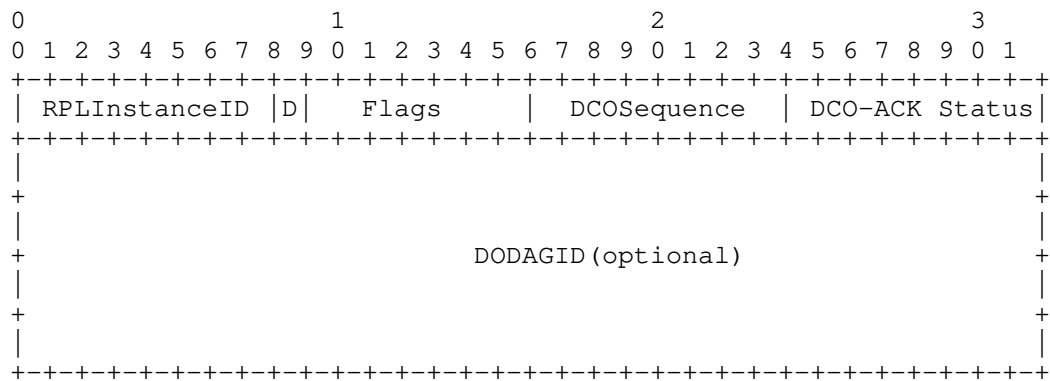


Figure 4: DCO-ACK base object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: 7-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

DCOSequence: 8-bit field. The DCOSequence in DCO-ACK is copied from the DCOSequence received in the DCO message.

DCO-ACK Status: Indicates the completion. A value of 0 is defined as unqualified acceptance in this specification. A value of 1 is defined as "No routing-entry for the Target found". The remaining status values are reserved as rejection codes.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present when 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

4.3.5. Secure DCO-ACK

A Secure DCO-ACK message follows the format in [RFC6550] Figure 7, where the base message format is the DCO-ACK message shown in Figure 4.

4.4. DCO Base Rules

1. If a node sends a DCO message with newer or different information than the prior DCO message transmission, it MUST increment the DCOSequence field by at least one. A DCO message transmission that is identical to the prior DCO message transmission MAY increment the DCOSequence field. The DCOSequence counter follows the sequence counter operation as defined in Section 7.2 of [RFC6550].
2. The RPLInstanceID and DODAGID fields of a DCO message MUST be the same value as that of the DAO message in response to which the DCO is generated on the common ancestor node.
3. A node MAY set the 'K' flag in a unicast DCO message to solicit a unicast DCO-ACK in response in order to confirm the attempt.
4. A node receiving a unicast DCO message with the 'K' flag set SHOULD respond with a DCO-ACK. A node receiving a DCO message without the 'K' flag set MAY respond with a DCO-ACK, especially to report an error condition.
5. A node receiving a unicast DCO message MUST verify the stored Path Sequence in context to the given target. If the stored Path Sequence is more fresh, newer than the Path Sequence received in the DCO, then the DCO MUST be dropped.
6. A node that sets the 'K' flag in a unicast DCO message but does not receive DCO-ACK in response MAY reschedule the DCO message transmission for another attempt, up until an implementation specific number of retries.
7. A node receiving a unicast DCO message with its own address in the RPL Target Option MUST strip-off that Target Option. If this Target Option is the only one in the DCO message then the DCO message MUST be dropped.

The scope of DCOSequence values is unique to the node which generates it.

4.5. Unsolicited DCO

A 6LR may generate an unsolicited DCO to unilaterally cleanup the path on behalf of the target entry. The 6LR has all the state information, namely, the Target address and the Path Sequence, required for generating DCO in its routing table. The conditions why 6LR may generate an unsolicited DCO are beyond the scope of this document but some possible reasons could be:

1. On route expiry of an entry, a 6LR may decide to graciously cleanup the entry by initiating DCO.
2. 6LR needs to entertain higher priority entries in case the routing table is full, thus resulting in eviction of an existing routing entry. In this case the eviction can be handled graciously using DCO.

Note that if the 6LR initiates a unilateral path cleanup using DCO and if it has the latest state for the target then the DCO would finally reach the target node. Thus the target node would be informed of its invalidation.

4.6. Other considerations

4.6.1. Dependent Nodes invalidation

Current RPL [RFC6550] does not provide a mechanism for route invalidation for dependent nodes. This document allows the dependent nodes invalidation. Dependent nodes will generate their respective DAOs to update their paths, and the previous route invalidation for those nodes should work in the similar manner described for switching node. The dependent node may set the 'I' flag in the Transit Information Option as part of regular DAO so as to request invalidation of previous route from the common ancestor node.

Dependent nodes do not have any indication regarding if any of their parents in turn have decided to switch their parent. Thus for route invalidation the dependent nodes may choose to always set the 'I' flag in all its DAO message's Transit Information Option. Note that setting the 'I' flag is not counterproductive even if there is no previous route to be invalidated.

4.6.2. NPDAO and DCO in the same network

The current NPDAO mechanism in [RFC6550] can still be used in the same network where DCO is used. The NPDAO messaging can be used, for example, on route lifetime expiry of the target or when the node simply decides to gracefully terminate the RPL session on graceful node shutdown. Moreover, a deployment can have a mix of nodes supporting the DCO and the existing NPDAO mechanism. It is also possible that the same node supports both the NPDAO and DCO signaling for route invalidation.

Section 9.8 of [RFC6550] states, "When a node removes a node from its DAO parent set, it SHOULD send a No-Path DAO message to that removed DAO parent to invalidate the existing router". This document introduces an alternative and more optimized way of route invalidation but it also allows existing NPDAO messaging to work. Thus an implementation has two choices to make when a route invalidation is to be initiated:

1. Use NPDAO to invalidate the previous route and send regular DAO on the new path.
2. Send regular DAO on the new path with the 'I' flag set in the Transit Information Option such that the common ancestor node initiates the DCO message downstream to invalidate the previous route.

This document recommends using option 2 for reasons specified in Section 3 in this document.

This document assumes that all the 6LRs in the network support this specification. If there are 6LRs en-route DCO message path which do not support this document, then the route invalidation for corresponding targets may not work or may work partially i.e., only part of the path supporting DCO may be invalidated. Alternatively, a node could generate an NPDAO if it does not receive a DCO with itself as target within specified time limit. The specified time limit is deployment specific and depends upon the maximum depth of the network and per hop average latency. Note that sending NPDAO and DCO for the same operation would not result in unwanted side-effects because the acceptability of NPDAO or DCO depends upon the Path Sequence freshness.

4.6.3. Considerations for DCO retry

A DCO message could be retried by a sender if it sets the 'K' flag and does not receive a DCO-ACK. The DCO retry time could be dependent on the maximum depth of the network and average per hop latency. This could range from 2 seconds to 120 seconds depending on

the deployment. In case the latency limits are not known, an implementation MUST NOT retry more than once in 3 seconds and MUST NOT retry more than 3 times.

The number of retries could also be set depending on how critical the route invalidation could be for the deployment and the link layer retry configuration. For networks supporting only MP2P and P2MP flows, such as in AMI and telemetry applications, the 6LRs may not be very keen to invalidate routes, unless they are highly memory-constrained. For home and building automation networks which may have substantial P2P traffic, the 6LRs might be keen to invalidate efficiently because it may additionally impact the forwarding efficiency.

4.6.4. DCO with multiple preferred parents

[RFC6550] allows a node to select multiple preferred parents for route establishment. Section 9.2.1 of [RFC6550] specifies, "All DAOs generated at the same time for the same Target MUST be sent with the same Path Sequence in the Transit Information". Subsequently when route invalidation has to be initiated, RPL mentions use of NPDAO which can be initiated with an updated Path Sequence to all the parent nodes through which the route is to be invalidated.

With DCO, the Target node itself does not initiate the route invalidation and it is left to the common ancestor node. A common ancestor node when it discovers an updated DAO from a new next-hop, it initiates a DCO. With multiple preferred parents, this handling does not change. But in this case it is recommended that an implementation initiates a DCO after a time period (DelayDCO) such that the common ancestor node may receive updated DAOs from all possible next-hops. This will help to reduce DCO control overhead i.e., the common ancestor can wait for updated DAOs from all possible directions before initiating a DCO for route invalidation. After timeout, the DCO needs to be generated for all the next-hops for whom the route invalidation needs to be done.

This document recommends using a DelayDCO timer value of 1sec. This value is inspired by the default DelayDAO value of 1sec in [RFC6550]. Here the hypothesis is that the DAOs from all possible parent sets would be received on the common ancestor within this time period.

It is still possible that a DCO is generated before all the updated DAOs from all the paths are received. In this case, the ancestor node would start the invalidation procedure for paths from which the updated DAO is not received. The DCO generated in this case would start invalidating the segments along these paths on which the updated DAOs are not received. But once the DAO reaches these

segments, the routing state would be updated along these segments and should not lead to any inconsistent routing state.

Note that there is no requirement for synchronization between DCO and DAOs. The DelayDCO timer simply ensures that the DCO control overhead can be reduced and is only needed when the network contains nodes using multiple preferred parent.

5. Acknowledgments

Many thanks to Alvaro Retana, Cenk Gundogan, Simon Duquennoy, Georgios Papadopoulos, Peter Van Der Stok for their review and comments. Alvaro Retana helped shape this document's final version with critical review comments.

6. IANA Considerations

IANA is requested to allocate new codes for the DCO and DCO-ACK messages from the RPL Control Codes registry.

Code	Description	Reference
TBD1	Destination Cleanup Object	This document
TBD2	Destination Cleanup Object Acknowledgment	This document
TBD3	Secure Destination Cleanup Object	This document
TBD4	Secure Destination Cleanup Object Acknowledgment	This document

IANA is requested to allocate bit 1 from the Transit Information Option Flags registry for the 'I' flag (Section 4.2)

6.1. New Registry for the Destination Cleanup Object (DCO) Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Flags field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description

- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DCO-ACK request (K)	This document
1	DODAGID field is present (D)	This document

DCO Base Flags

6.2. New Registry for the Destination Cleanup Object Acknowledgment (DCO-ACK) Status field

IANA is requested to create a registry for the 8-bit Destination Cleanup Object Acknowledgment (DCO-ACK) Status field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New Status values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- o Status Code
- o Description
- o Defining RFC

The following values are currently defined:

Status Code	Description	Reference
0	Unqualified acceptance	This document
1	No routing-entry for the indicated Target found	This document

DCO-ACK Status Codes

6.3. New Registry for the Destination Cleanup Object (DCO) Acknowledgment Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Acknowledgment Flags field. This registry

should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description
- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DODAGID field is present (D)	This document

DCO-ACK Base Flags

7. Security Considerations

This document introduces the ability for a common ancestor node to invalidate a route on behalf of the target node. The common ancestor node could be directed to do so by the target node using the 'I' flag in DCO's Transit Information Option. However, the common ancestor node is in a position to unilaterally initiate the route invalidation since it possesses all the required state information, namely, the Target address and the corresponding Path Sequence. Thus a rogue common ancestor node could initiate such an invalidation and impact the traffic to the target node.

The DCO carries a RPL Status value, which is informative. New Status values may be created over time and a node will ignore an unknown Status value. This enables RPL Status field to be used as a cover channel. But the channel only works once since the message destroys its own medium, that is the existing route that it is removing.

This document also introduces an 'I' flag which is set by the target node and used by the ancestor node to initiate a DCO if the ancestor sees an update in the route adjacency. However, this flag could be spoofed by a malicious 6LR in the path and can cause invalidation of an existing active path. Note that invalidation will happen only if the other conditions such as Path Sequence condition is also met. Having said that, such a malicious 6LR may spoof a DAO on behalf of the (sub) child with the 'I' flag set and can cause route invalidation on behalf of the (sub) child node. Note that, using existing mechanisms offered by [RFC6550], a malicious 6LR might also

spoof a DAO with lifetime of zero or otherwise cause denial of service by dropping traffic entirely, so the new mechanism described in this document does not present a substantially increased risk of disruption.

This document assumes that the security mechanisms as defined in [RFC6550] are followed, which means that the common ancestor node and all the 6LRs are part of the RPL network because they have the required credentials. A non-secure RPL network needs to take into consideration the risks highlighted in this section as well as those highlighted in [RFC6550].

All RPL messages support a secure version of messages which allows integrity protection using either a MAC or a signature. Optionally, secured RPL messages also have encryption protection for confidentiality.

The document adds new messages (DCO, DCO-ACK) which are syntactically similar to existing RPL messages such as DAO, DAO-ACK. Secure versions of DCO and DCO-ACK are added similar to other RPL messages (such as DAO, DAO-ACK).

RPL supports three security modes as mentioned in Section 10.1 of [RFC6550]:

1. Unsecured: In this mode, it is expected that the RPL control messages are secured by other security mechanisms, such as link-layer security. In this mode, the RPL control messages, including DCO, DCO-ACK, do not have Security sections. Also note that unsecured mode does not imply that all messages are sent without any protection.
2. Preinstalled: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.
3. Authenticated: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.

8. Normative References

[I-D.ietf-roll-unaware-leaves]

Thubert, P. and M. Richardson, "Routing for RPL Leaves", draft-ietf-roll-unaware-leaves-04 (work in progress), September 2019.

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

- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Appendix A. Example Messaging

A.1. Example DCO Messaging

In Figure 1, node (D) switches its parent from (B) to (C). This example assumes that Node D has already established its own route via Node B-G-A-6LBR using pathseq=x. The example uses DAO and DCO messaging convention and specifies only the required parameters to explain the example namely, the parameter 'tgt', which stands for Target Option and value of this parameter specifies the address of the target node. The parameter 'pathseq', which specifies the Path Sequence value carried in the Transit Information Option. The parameter 'I_flag' specifies the 'I' flag in the Transit Information Option. sequence of actions is as follows:

1. Node D switches its parent from node B to node C
2. D sends a regular DAO(tgt=D,pathseq=x+1,I_flag=1) in the updated path to C
3. C checks for a routing entry on behalf of D, since it cannot find an entry on behalf of D it creates a new routing entry and forwards the reachability information of the target D to H in a DAO(tgt=D,pathseq=x+1,I_flag=1).
4. Similar to C, node H checks for a routing entry on behalf of D, cannot find an entry and hence creates a new routing entry and forwards the reachability information of the target D to A in a DAO(tgt=D,pathseq=x+1,I_flag=1).
5. Node A receives the DAO(tgt=D,pathseq=x+1,I_flag=1), and checks for a routing entry on behalf of D. It finds a routing entry but checks that the next hop for target D is different (i.e., Node G). Node A checks the I_flag and generates DCO(tgt=D,pathseq=x+1) to previous next hop for target D which is G. Subsequently, Node A updates the routing entry and forwards the reachability information of target D upstream DAO(tgt=D,pathseq=x+1,I_flag=1).
6. Node G receives the DCO(tgt=D,pathseq=x+1). It checks if the received path sequence is later than the stored path sequence. If it is later, Node G invalidates the routing entry of target D

- and forwards the (un)reachability information downstream to B in `DCO(tgt=D,pathseq=x+1)`.
7. Similarly, B processes the `DCO(tgt=D,pathseq=x+1)` by invalidating the routing entry of target D and forwards the (un)reachability information downstream to D.
 8. D ignores the `DCO(tgt=D,pathseq=x+1)` since the target is itself.
 9. The propagation of the DCO will stop at any node where the node does not have an routing information associated with the target. If cached routing information is present and the cached Path Sequence is higher than the value in the DCO, then the DCO is dropped.

A.2. Example DCO Messaging with multiple preferred parents

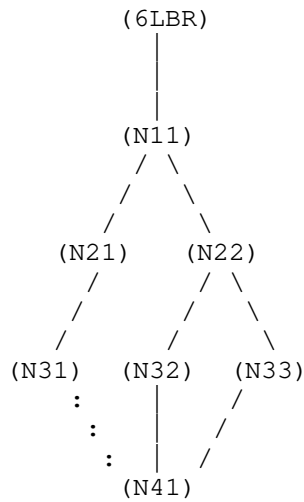


Figure 5: Sample topology 2

In Figure 5, node (N41) selects multiple preferred parents (N32) and (N33). The sequence of actions is as follows:

1. (N41) sends `DAO(tgt=N41,PS=x,I_flag=1)` to (N32) and (N33). Here `I_flag` refers to the Invalidation flag and `PS` refers to Path Sequence in Transit Information option.
2. (N32) sends `DAO(tgt=N41,PS=x,I_flag=1)` to (N22). (N33) also sends `DAO(tgt=N41,PS=x,I_flag=1)` to (N22). (N22) learns multiple routes for the same destination (N41) through multiple next-hops. (N22) may receive the DAOs from (N32) and (N33) in any order with the `I_flag` set. The implementation should use the DelayDCO timer to wait to initiate the DCO. If (N22) receives an updated DAO from all the paths then the DCO need not

- be initiated in this case. Thus the route table at N22 should contain (Dst,NextHop,PS): { (N41,N32,x), (N41,N33,x) }.
3. (N22) sends DAO(tgt=N41,PS=x,I_flag=1) to (N11).
 4. (N11) sends DAO(tgt=N41,PS=x,I_flag=1) to (6LBR). Thus the complete path is established.
 5. (N41) decides to change preferred parent set from { N32, N33 } to { N31, N32 }.
 6. (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N32). (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N31).
 7. (N32) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N22). (N22) has multiple routes to destination (N41). It sees that a new Path Sequence for Target=N41 is received and thus it waits for pre-determined time period (DelayDCO time period) to invalidate another route { (N41), (N33), x}. After time period, (N22) sends DCO(tgt=N41,PS=x+1) to (N33). Also (N22) sends the regular DAO(tgt=N41,PS=x+1,I_flag=1) to (N11).
 8. (N33) receives DCO(tgt=N41,PS=x+1). The received Path Sequence is latest and thus it invalidates the entry associated with target (N41). (N33) then sends the DCO(tgt=N41,PS=x+1) to (N41). (N41) sees itself as the target and drops the DCO.
 9. From Step 6 above, (N31) receives the DAO(tgt=N41,PS=x+1,I_flag=1). It creates a routing entry and sends the DAO(tgt=N41,PS=x+1,I_flag=1) to (N21). Similarly (N21) receives the DAO and subsequently sends the DAO(tgt=N41,PS=x+1,I_flag=1) to (N11).
 10. (N11) receives DAO(tgt=N41,PS=x+1,I_flag=1) from (N21). It waits for DelayDCO timer since it has multiple routes to (N41). (N41) will receive DAO(tgt=N41,PS=x+1,I_flag=1) from (N22) from Step 7 above. Thus (N11) has received regular DAO(tgt=N41,PS=x+1,I_flag=1) from all paths and thus does not initiate DCO.
 11. (N11) forwards the DAO(tgt=N41,PS=x+1,I_flag=1) to 6LBR and the full path is established.

Authors' Addresses

Rahul Arvind Jadhav (editor)
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

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

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
France

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

Rabi Narayan Sahoo
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

Phone: +91-080-49160700
Email: rabinarayans@huawei.com

Zhen Cao
Huawei
W Chang'an Ave
Beijing
P.R. China

Email: zhencao.ietf@gmail.com

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

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

Table of Contents

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

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

1. Introduction

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

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

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

1.1. Requirements Language and Terminology

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

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

Low Power and Lossy Networks (LLN):

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

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

6LoWPAN Router (6LR):

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

Directed Acyclic Graph (DAG):

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

Destination-Oriented DAG (DODAG):

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

6LoWPAN Border Router (6LBR):

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

Destination Advertisement Object (DAO):

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

DODAG Information Object (DIO):

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

Common Ancestor node

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

No-Path DAO (NPDAO):

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

Destination Cleanup Object (DCO):

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

Regular DAO:

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

Target node:

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

1.2. Current NPDAO messaging

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

For the rest of this document consider the following topology:

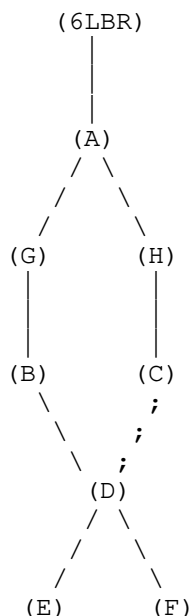


Figure 1: Sample topology

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

1.3. Why Is NPDAO Important?

Nodes in LLNs may be resource constrained. There is limited memory available and routing entry records are one of the primary elements occupying dynamic memory in the nodes. Route invalidation helps 6LR nodes to decide which entries could be discarded to better optimize resource utilization. Thus it becomes necessary to have an efficient route invalidation mechanism. Also note that a single parent switch may result in a "sub-tree" switching from one parent to another. Thus the route invalidation needs to be done on behalf of the sub-tree and not the switching node alone. In the above example, when Node (D) switches parent, the route updates needs to be done for the routing tables entries of (C), (H), (A), (G), and (B) with destination (D), (E) and (F). Without efficient route invalidation, a 6LR may have to hold a lot of stale route entries.

2. Problems with current NPDAO messaging

2.1. Lost NPDAO due to link break to the previous parent

When a node switches its parent, the NPDAO is to be sent to its previous parent and a regular DAO to its new parent. In cases where the node switches its parent because of transient or permanent parent link/node failure then the NPDAO message is bound to fail.

2.2. Invalidate Routes of Dependent Nodes

RPL does not specify how route invalidation will work for dependent nodes rooted at the switching node, resulting in stale routing entries of the dependent nodes. The only way for 6LR to invalidate the route entries for dependent nodes would be to use route lifetime expiry which could be substantially high for LLNs.

In the example topology, when Node (D) switches its parent, Node (D) generates an NPDAO on its behalf. There is no NPDAO generated by the dependent child nodes (E) and (F), through the previous path via (D) to (B) and (G), resulting in stale entries on nodes (B) and (G) for nodes (E) and (F).

2.3. Possible route downtime caused by asynchronous operation of NPDAO and DAO

A switching node may generate both an NPDAO and DAO via two different paths at almost the same time. There is a possibility that an NPDAO generated may invalidate the previous route and the regular DAO sent via the new path gets lost on the way. This may result in route downtime impacting downward traffic for the switching node.

In the example topology, consider Node (D) switches from parent (B) to (C). An NPDAO sent via the previous route may invalidate the previous route whereas there is no way to determine whether the new DAO has successfully updated the route entries on the new path.

3. Requirements for the NPDAO Optimization

3.1. Req#1: Remove messaging dependency on link to the previous parent

When the switching node sends the NPDAO message to the previous parent, it is normal that the link to the previous parent is prone to failure (that's why the node decided to switch). Therefore, it is required that the route invalidation does not depend on the previous link which is prone to failure. The previous link referred here represents the link between the node and its previous parent (from whom the node is now disassociating).

3.2. Req#2: Dependent nodes route invalidation on parent switching

It should be possible to do route invalidation for dependent nodes rooted at the switching node.

3.3. Req#3: Route invalidation should not impact data traffic

While sending the NPDAO and DAO messages, it is possible that the NPDAO successfully invalidates the previous path, while the newly sent DAO gets lost (new path not set up successfully). This will result in downstream unreachability to the node switching paths. Therefore, it is desirable that the route invalidation is synchronized with the DAO to avoid the risk of route downtime.

4. Changes to RPL signaling

4.1. Change in RPL route invalidation semantics

As described in Section 1.2, the NPDAO originates at the node changing to a new parent and traverses upstream towards the root. In order to solve the problems as mentioned in Section 2, the document adds a new proactive route invalidation message called "Destination Cleanup Object" (DCO) that originates at a common ancestor node and flows downstream between the new and old path. The common ancestor node generates a DCO in response to the change in the next-hop on receiving a regular DAO with updated Path Sequence for the target.

The 6LRs in the path for DCO take action such as route invalidation based on the DCO information and subsequently send another DCO with the same information downstream to the next hop. This operation is similar to how the DAOs are handled on intermediate 6LRs in storing MOP in [RFC6550]. Just like DAO in storing MOP, the DCO is sent using link-local unicast source and destination IPv6 address. Unlike DAO, which always travels upstream, the DCO always travels downstream.

In Figure 1, when node D decides to switch the path from B to C, it sends a regular DAO to node C with reachability information containing the address of D as the target and an incremented Path Sequence. Node C will update the routing table based on the reachability information in the DAO and in turn generate another DAO with the same reachability information and forward it to H. Node H also follows the same procedure as Node C and forwards it to node A. When node A receives the regular DAO, it finds that it already has a routing table entry on behalf of the target address of node D. It finds however that the next hop information for reaching node D has changed i.e., node D has decided to change the paths. In this case, Node A which is the common ancestor node for node D along the two

paths (previous and new), should generate a DCO which traverses downwards in the network. Node A handles normal DAO forwarding to 6LBR as required by [RFC6550].

4.2. Transit Information Option changes

Every RPL message is divided into base message fields and additional Options as described in Section 6 of [RFC6550]. The base fields apply to the message as a whole and options are appended to add message/use-case specific attributes. As an example, a DAO message may be attributed by one or more "RPL Target" options which specify the reachability information for the given targets. Similarly, a Transit Information option may be associated with a set of RPL Target options.

This document specifies a change in the Transit Information Option to contain the "Invalidate previous route" (I) flag. This 'I' flag signals the common ancestor node to generate a DCO on behalf of the target node with a RPL Status of 195 indicating that the address has moved. The 'I' flag is carried in the Transit Information Option which augments the reachability information for a given set of RPL Target(s). Transit Information Option with 'I' flag set should be carried in the DAO message when route invalidation is sought for the corresponding target(s).

Value 195 represents 'E' and 'A' bit in RPL Status to be set as per Figure 3 of [I-D.ietf-roll-unaware-leaves] with the lower 6 bits with value 3 indicating 'Moved' as per Table 1 of [RFC8505].

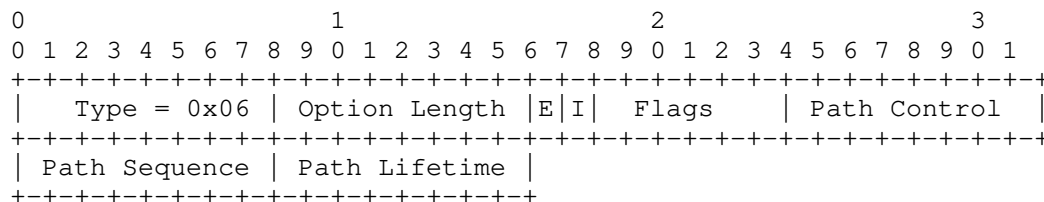


Figure 2: Updated Transit Information Option (New I flag added)

I (Invalidate previous route) flag: The 'I' flag is set by the target node to indicate to the common ancestor node that it wishes to invalidate any previous route between the two paths.

[RFC6550] allows the parent address to be sent in the Transit Information Option depending on the mode of operation. In case of storing mode of operation the field is usually not needed. In case of DCO, the parent address field MUST NOT be included.

D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: The 6 bits remaining unused in the Flags field are reserved for future use. These bits MUST be initialized to zero by the sender and MUST be ignored by the receiver.

RPL Status: As defined in [RFC6550] and updated in [I-D.ietf-roll-unaware-leaves]. The root or common parent that generates a DCO is authoritative for setting the status information and the information is unchanged as propagated down the DODAG. This document does not specify a differentiated action based on the RPL status.

DCOSequence: 8-bit field incremented at each unique DCO message from a node and echoed in the DCO-ACK message. The initial DCOSequence can be chosen randomly by the node. Section 4.4 explains the handling of the DCOSequence.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present if 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

4.3.1. Secure DCO

A Secure DCO message follows the format in [RFC6550] Figure 7, where the base message format is the DCO message shown in Figure 3.

4.3.2. DCO Options

The DCO message MUST carry at least one RPL Target and the Transit Information Option and MAY carry other valid options. This specification allows for the DCO message to carry the following options:

- 0x00 Pad1
- 0x01 PadN
- 0x05 RPL Target
- 0x06 Transit Information
- 0x09 RPL Target Descriptor

Section 6.7 of [RFC6550] defines all the above mentioned options. The DCO carries an RPL Target Option and an associated Transit Information Option with a lifetime of 0x00000000 to indicate a loss of reachability to that Target.

4.3.3. Path Sequence number in the DCO

A DCO message may contain a Path Sequence in the Transit Information Option to identify the freshness of the DCO message. The Path Sequence in the DCO MUST use the same Path Sequence number present in the regular DAO message when the DCO is generated in response to a DAO message. Thus if a DCO is received by a 6LR and subsequently a DAO is received with an old sequence number, then the DAO MUST be ignored. When the DCO is generated in response to a DCO from upstream parent, the Path Sequence MUST be copied from the received DCO.

4.3.4. Destination Cleanup Option Acknowledgment (DCO-ACK)

The DCO-ACK message SHOULD be sent as a unicast packet by a DCO recipient in response to a unicast DCO message with 'K' flag set. If 'K' flag is not set then the receiver of the DCO message MAY send a DCO-ACK, especially to report an error condition.

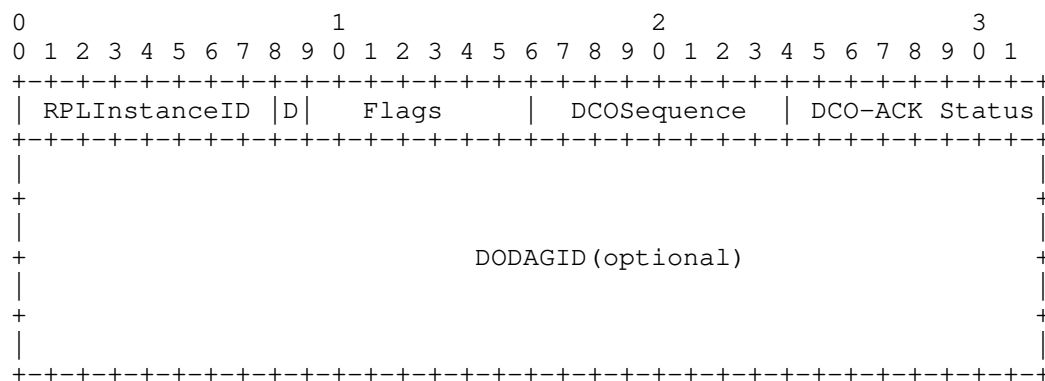


Figure 4: DCO-ACK base object

RPLInstanceID: 8-bit field indicating the topology instance associated with the DODAG, as learned from the DIO.

D: The 'D' flag indicates that the DODAGID field is present. This flag MUST be set when a local RPLInstanceID is used.

Flags: 7-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

DCOSequence: 8-bit field. The DCOSequence in DCO-ACK is copied from the DCOSequence received in the DCO message.

DCO-ACK Status: Indicates the completion. A value of 0 is defined as unqualified acceptance in this specification. A value of 1 is defined as "No routing-entry for the Target found". The remaining status values are reserved as rejection codes.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field MUST be present when the 'D' flag is set and MUST NOT be present when 'D' flag is not set. DODAGID is used when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID.

4.3.5. Secure DCO-ACK

A Secure DCO-ACK message follows the format in [RFC6550] Figure 7, where the base message format is the DCO-ACK message shown in Figure 4.

4.4. DCO Base Rules

1. If a node sends a DCO message with newer or different information than the prior DCO message transmission, it MUST increment the DCOSequence field by at least one. A DCO message transmission that is identical to the prior DCO message transmission MAY increment the DCOSequence field. The DCOSequence counter follows the sequence counter operation as defined in Section 7.2 of [RFC6550].
2. The RPLInstanceID and DODAGID fields of a DCO message MUST be the same value as that of the DAO message in response to which the DCO is generated on the common ancestor node.
3. A node MAY set the 'K' flag in a unicast DCO message to solicit a unicast DCO-ACK in response in order to confirm the attempt.
4. A node receiving a unicast DCO message with the 'K' flag set SHOULD respond with a DCO-ACK. A node receiving a DCO message without the 'K' flag set MAY respond with a DCO-ACK, especially to report an error condition.
5. A node receiving a unicast DCO message MUST verify the stored Path Sequence in context to the given target. If the stored Path Sequence is more fresh, newer than the Path Sequence received in the DCO, then the DCO MUST be dropped.
6. A node that sets the 'K' flag in a unicast DCO message but does not receive DCO-ACK in response MAY reschedule the DCO message transmission for another attempt, up until an implementation specific number of retries.
7. A node receiving a unicast DCO message with its own address in the RPL Target Option MUST strip-off that Target Option. If this Target Option is the only one in the DCO message then the DCO message MUST be dropped.

The scope of DCOSequence values is unique to the node which generates it.

4.5. Unsolicited DCO

A 6LR may generate an unsolicited DCO to unilaterally cleanup the path on behalf of the target entry. The 6LR has all the state information, namely, the Target address and the Path Sequence, required for generating DCO in its routing table. The conditions why 6LR may generate an unsolicited DCO are beyond the scope of this document but some possible reasons could be:

1. On route expiry of an entry, a 6LR may decide to graciously cleanup the entry by initiating DCO.
2. 6LR needs to entertain higher priority entries in case the routing table is full, thus resulting in eviction of an existing routing entry. In this case the eviction can be handled graciously using DCO.

Note that if the 6LR initiates a unilateral path cleanup using DCO and if it has the latest state for the target then the DCO would finally reach the target node. Thus the target node would be informed of its invalidation.

4.6. Other considerations

4.6.1. Dependent Nodes invalidation

Current RPL [RFC6550] does not provide a mechanism for route invalidation for dependent nodes. This document allows the dependent nodes invalidation. Dependent nodes will generate their respective DAOs to update their paths, and the previous route invalidation for those nodes should work in the similar manner described for switching node. The dependent node may set the 'I' flag in the Transit Information Option as part of regular DAO so as to request invalidation of previous route from the common ancestor node.

Dependent nodes do not have any indication regarding if any of their parents in turn have decided to switch their parent. Thus for route invalidation the dependent nodes may choose to always set the 'I' flag in all its DAO message's Transit Information Option. Note that setting the 'I' flag is not counterproductive even if there is no previous route to be invalidated.

4.6.2. NPDAO and DCO in the same network

The current NPDAO mechanism in [RFC6550] can still be used in the same network where DCO is used. The NPDAO messaging can be used, for example, on route lifetime expiry of the target or when the node simply decides to gracefully terminate the RPL session on graceful node shutdown. Moreover, a deployment can have a mix of nodes supporting the DCO and the existing NPDAO mechanism. It is also possible that the same node supports both the NPDAO and DCO signaling for route invalidation.

Section 9.8 of [RFC6550] states, "When a node removes a node from its DAO parent set, it SHOULD send a No-Path DAO message to that removed DAO parent to invalidate the existing router". This document introduces an alternative and more optimized way of route invalidation but it also allows existing NPDAO messaging to work. Thus an implementation has two choices to make when a route invalidation is to be initiated:

1. Use NPDAO to invalidate the previous route and send regular DAO on the new path.
2. Send regular DAO on the new path with the 'I' flag set in the Transit Information Option such that the common ancestor node initiates the DCO message downstream to invalidate the previous route.

This document recommends using option 2 for reasons specified in Section 3 in this document.

This document assumes that all the 6LRs in the network support this specification. If there are 6LRs en-route DCO message path which do not support this document, then the route invalidation for corresponding targets may not work or may work partially i.e., only part of the path supporting DCO may be invalidated. Alternatively, a node could generate an NPDAO if it does not receive a DCO with itself as target within specified time limit. The specified time limit is deployment specific and depends upon the maximum depth of the network and per hop average latency. Note that sending NPDAO and DCO for the same operation would not result in unwanted side-effects because the acceptability of NPDAO or DCO depends upon the Path Sequence freshness.

4.6.3. Considerations for DCO retry

A DCO message could be retried by a sender if it sets the 'K' flag and does not receive a DCO-ACK. The DCO retry time could be dependent on the maximum depth of the network and average per hop latency. This could range from 2 seconds to 120 seconds depending on

the deployment. In case the latency limits are not known, an implementation MUST NOT retry more than once in 3 seconds and MUST NOT retry more than 3 times.

The number of retries could also be set depending on how critical the route invalidation could be for the deployment and the link layer retry configuration. For networks supporting only MP2P and P2MP flows, such as in AMI and telemetry applications, the 6LRs may not be very keen to invalidate routes, unless they are highly memory-constrained. For home and building automation networks which may have substantial P2P traffic, the 6LRs might be keen to invalidate efficiently because it may additionally impact the forwarding efficiency.

4.6.4. DCO with multiple preferred parents

[RFC6550] allows a node to select multiple preferred parents for route establishment. Section 9.2.1 of [RFC6550] specifies, "All DAOs generated at the same time for the same Target MUST be sent with the same Path Sequence in the Transit Information". Subsequently when route invalidation has to be initiated, RPL mentions use of NPDAO which can be initiated with an updated Path Sequence to all the parent nodes through which the route is to be invalidated.

With DCO, the Target node itself does not initiate the route invalidation and it is left to the common ancestor node. A common ancestor node when it discovers an updated DAO from a new next-hop, it initiates a DCO. With multiple preferred parents, this handling does not change. But in this case it is recommended that an implementation initiates a DCO after a time period (DelayDCO) such that the common ancestor node may receive updated DAOs from all possible next-hops. This will help to reduce DCO control overhead i.e., the common ancestor can wait for updated DAOs from all possible directions before initiating a DCO for route invalidation. After timeout, the DCO needs to be generated for all the next-hops for whom the route invalidation needs to be done.

This document recommends using a DelayDCO timer value of 1sec. This value is inspired by the default DelayDAO value of 1sec in [RFC6550]. Here the hypothesis is that the DAOs from all possible parent sets would be received on the common ancestor within this time period.

It is still possible that a DCO is generated before all the updated DAOs from all the paths are received. In this case, the ancestor node would start the invalidation procedure for paths from which the updated DAO is not received. The DCO generated in this case would start invalidating the segments along these paths on which the updated DAOs are not received. But once the DAO reaches these

segments, the routing state would be updated along these segments and should not lead to any inconsistent routing state.

Note that there is no requirement for synchronization between DCO and DAOs. The DelayDCO timer simply ensures that the DCO control overhead can be reduced and is only needed when the network contains nodes using multiple preferred parent.

5. Acknowledgments

Many thanks to Alvaro Retana, Cenk Gundogan, Simon Duquennoy, Georgios Papadopoulos, Peter Van Der Stok for their review and comments. Alvaro Retana helped shape this document's final version with critical review comments.

6. IANA Considerations

IANA is requested to allocate new codes for the DCO and DCO-ACK messages from the RPL Control Codes registry.

Code	Description	Reference
TBD1	Destination Cleanup Object	This document
TBD2	Destination Cleanup Object Acknowledgment	This document
TBD3	Secure Destination Cleanup Object	This document
TBD4	Secure Destination Cleanup Object Acknowledgment	This document

IANA is requested to allocate bit 1 from the Transit Information Option Flags registry for the 'I' flag (Section 4.2)

6.1. New Registry for the Destination Cleanup Object (DCO) Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Flags field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description

- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DCO-ACK request (K)	This document
1	DODAGID field is present (D)	This document

DCO Base Flags

6.2. New Registry for the Destination Cleanup Object Acknowledgment (DCO-ACK) Status field

IANA is requested to create a registry for the 8-bit Destination Cleanup Object Acknowledgment (DCO-ACK) Status field. This registry should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New Status values may be allocated only by an IETF Review. Each value is tracked with the following qualities:

- o Status Code
- o Description
- o Defining RFC

The following values are currently defined:

Status Code	Description	Reference
0	Unqualified acceptance	This document
1	No routing-entry for the indicated Target found	This document

DCO-ACK Status Codes

6.3. New Registry for the Destination Cleanup Object (DCO) Acknowledgment Flags

IANA is requested to create a registry for the 8-bit Destination Cleanup Object (DCO) Acknowledgment Flags field. This registry

should be located in existing category of "Routing Protocol for Low Power and Lossy Networks (RPL)".

New bit numbers may be allocated only by an IETF Review. Each bit is tracked with the following qualities:

- o Bit number (counting from bit 0 as the most significant bit)
- o Capability description
- o Defining RFC

The following bits are currently defined:

Bit number	Description	Reference
0	DODAGID field is present (D)	This document

DCO-ACK Base Flags

7. Security Considerations

This document introduces the ability for a common ancestor node to invalidate a route on behalf of the target node. The common ancestor node could be directed to do so by the target node using the 'I' flag in DCO's Transit Information Option. However, the common ancestor node is in a position to unilaterally initiate the route invalidation since it possesses all the required state information, namely, the Target address and the corresponding Path Sequence. Thus a rogue common ancestor node could initiate such an invalidation and impact the traffic to the target node.

The DCO carries a RPL Status value, which is informative. New Status values may be created over time and a node will ignore an unknown Status value. This enables RPL Status field to be used as a cover channel. But the channel only works once since the message destroys its own medium, that is the existing route that it is removing.

This document also introduces an 'I' flag which is set by the target node and used by the ancestor node to initiate a DCO if the ancestor sees an update in the route adjacency. However, this flag could be spoofed by a malicious 6LR in the path and can cause invalidation of an existing active path. Note that invalidation will happen only if the other conditions such as Path Sequence condition is also met. Having said that, such a malicious 6LR may spoof a DAO on behalf of the (sub) child with the 'I' flag set and can cause route invalidation on behalf of the (sub) child node. Note that, using existing mechanisms offered by [RFC6550], a malicious 6LR might also

spoof a DAO with lifetime of zero or otherwise cause denial of service by dropping traffic entirely, so the new mechanism described in this document does not present a substantially increased risk of disruption.

This document assumes that the security mechanisms as defined in [RFC6550] are followed, which means that the common ancestor node and all the 6LRs are part of the RPL network because they have the required credentials. A non-secure RPL network needs to take into consideration the risks highlighted in this section as well as those highlighted in [RFC6550].

All RPL messages support a secure version of messages which allows integrity protection using either a MAC or a signature. Optionally, secured RPL messages also have encryption protection for confidentiality.

The document adds new messages (DCO, DCO-ACK) which are syntactically similar to existing RPL messages such as DAO, DAO-ACK. Secure versions of DCO and DCO-ACK are added similar to other RPL messages (such as DAO, DAO-ACK).

RPL supports three security modes as mentioned in Section 10.1 of [RFC6550]:

1. Unsecured: In this mode, it is expected that the RPL control messages are secured by other security mechanisms, such as link-layer security. In this mode, the RPL control messages, including DCO, DCO-ACK, do not have Security sections. Also note that unsecured mode does not imply that all messages are sent without any protection.
2. Preinstalled: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.
3. Authenticated: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.

8. Normative References

[I-D.ietf-roll-unaware-leaves]

Thubert, P. and M. Richardson, "Routing for RPL Leaves", draft-ietf-roll-unaware-leaves-14 (work in progress), April 2020.

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

- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Appendix A. Example Messaging

A.1. Example DCO Messaging

In Figure 1, node (D) switches its parent from (B) to (C). This example assumes that Node D has already established its own route via Node B-G-A-6LBR using pathseq=x. The example uses DAO and DCO messaging convention and specifies only the required parameters to explain the example namely, the parameter 'tgt', which stands for Target Option and value of this parameter specifies the address of the target node. The parameter 'pathseq', which specifies the Path Sequence value carried in the Transit Information Option. The parameter 'I_flag' specifies the 'I' flag in the Transit Information Option. sequence of actions is as follows:

1. Node D switches its parent from node B to node C
2. D sends a regular DAO(tgt=D,pathseq=x+1,I_flag=1) in the updated path to C
3. C checks for a routing entry on behalf of D, since it cannot find an entry on behalf of D it creates a new routing entry and forwards the reachability information of the target D to H in a DAO(tgt=D,pathseq=x+1,I_flag=1).
4. Similar to C, node H checks for a routing entry on behalf of D, cannot find an entry and hence creates a new routing entry and forwards the reachability information of the target D to A in a DAO(tgt=D,pathseq=x+1,I_flag=1).
5. Node A receives the DAO(tgt=D,pathseq=x+1,I_flag=1), and checks for a routing entry on behalf of D. It finds a routing entry but checks that the next hop for target D is different (i.e., Node G). Node A checks the I_flag and generates DCO(tgt=D,pathseq=x+1) to previous next hop for target D which is G. Subsequently, Node A updates the routing entry and forwards the reachability information of target D upstream DAO(tgt=D,pathseq=x+1,I_flag=1).
6. Node G receives the DCO(tgt=D,pathseq=x+1). It checks if the received path sequence is later than the stored path sequence. If it is later, Node G invalidates the routing entry of target D

and forwards the (un)reachability information downstream to B in DCO(tgt=D,pathseq=x+1).

7. Similarly, B processes the DCO(tgt=D,pathseq=x+1) by invalidating the routing entry of target D and forwards the (un)reachability information downstream to D.
8. D ignores the DCO(tgt=D,pathseq=x+1) since the target is itself.
9. The propagation of the DCO will stop at any node where the node does not have an routing information associated with the target. If cached routing information is present and the cached Path Sequence is higher than the value in the DCO, then the DCO is dropped.

A.2. Example DCO Messaging with multiple preferred parents

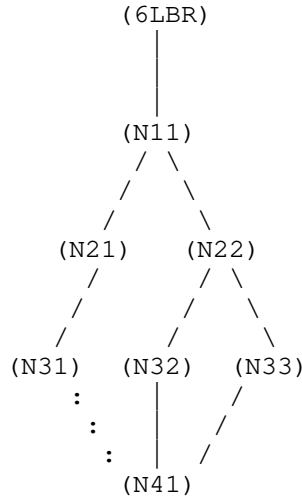


Figure 5: Sample topology 2

In Figure 5, node (N41) selects multiple preferred parents (N32) and (N33). The sequence of actions is as follows:

1. (N41) sends DAO(tgt=N41,PS=x,I_flag=1) to (N32) and (N33). Here I_flag refers to the Invalidation flag and PS refers to Path Sequence in Transit Information option.
2. (N32) sends DAO(tgt=N41,PS=x,I_flag=1) to (N22). (N33) also sends DAO(tgt=N41,PS=x,I_flag=1) to (N22). (N22) learns multiple routes for the same destination (N41) through multiple next-hops. (N22) may receive the DAOs from (N32) and (N33) in any order with the I_flag set. The implementation should use the DelayDCO timer to wait to initiate the DCO. If (N22) receives an updated DAO from all the paths then the DCO need not

- be initiated in this case. Thus the route table at N22 should contain (Dst,NextHop,PS): { (N41,N32,x), (N41,N33,x) }.
3. (N22) sends DAO(tgt=N41,PS=x,I_flag=1) to (N11).
 4. (N11) sends DAO(tgt=N41,PS=x,I_flag=1) to (6LBR). Thus the complete path is established.
 5. (N41) decides to change preferred parent set from { N32, N33 } to { N31, N32 }.
 6. (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N32). (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N31).
 7. (N32) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N22). (N22) has multiple routes to destination (N41). It sees that a new Path Sequence for Target=N41 is received and thus it waits for pre-determined time period (DelayDCO time period) to invalidate another route {(N41),(N33),x}. After time period, (N22) sends DCO(tgt=N41,PS=x+1) to (N33). Also (N22) sends the regular DAO(tgt=N41,PS=x+1,I_flag=1) to (N11).
 8. (N33) receives DCO(tgt=N41,PS=x+1). The received Path Sequence is latest and thus it invalidates the entry associated with target (N41). (N33) then sends the DCO(tgt=N41,PS=x+1) to (N41). (N41) sees itself as the target and drops the DCO.
 9. From Step 6 above, (N31) receives the DAO(tgt=N41,PS=x+1,I_flag=1). It creates a routing entry and sends the DAO(tgt=N41,PS=x+1,I_flag=1) to (N21). Similarly (N21) receives the DAO and subsequently sends the DAO(tgt=N41,PS=x+1,I_flag=1) to (N11).
 10. (N11) receives DAO(tgt=N41,PS=x+1,I_flag=1) from (N21). It waits for DelayDCO timer since it has multiple routes to (N41). (N41) will receive DAO(tgt=N41,PS=x+1,I_flag=1) from (N22) from Step 7 above. Thus (N11) has received regular DAO(tgt=N41,PS=x+1,I_flag=1) from all paths and thus does not initiate DCO.
 11. (N11) forwards the DAO(tgt=N41,PS=x+1,I_flag=1) to 6LBR and the full path is established.

Authors' Addresses

Rahul Arvind Jadhav (editor)
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

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

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
France

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

Rabi Narayan Sahoo
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

Phone: +91-080-49160700
Email: rabinarayans@huawei.com

Zhen Cao
Huawei
W Chang'an Ave
Beijing
P.R. China

Email: zhencao.ietf@gmail.com

ROLL
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2020

R. Jadhav, Ed.
Huawei Tech
P. Thubert
Cisco
M. Richardson
Sandelman Software Works
November 1, 2019

Mode of Operation extension and Capabilities
draft-ietf-roll-mopex-cap-01

Abstract

RPL allows different mode of operations which allows nodes to have a consensus on the basic primitives that must be supported to join the network. The MOP field in RFC6550 is of 3 bits and is fast depleting. This document extends the MOP field specification and adds a notion of capabilities using which the nodes can further advertise their support for, possibly optional, capabilities.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language and Terminology	3
2. Requirements for this document	3
3. Extended MOP Control Message Option	4
3.1. Final MOP	4
3.2. Handling MOPex	5
4. Capabilities	5
4.1. Capability Control Message Option	6
4.2. Capabilities Handshake	7
5. Implementations Consideration	7
6. Acknowledgements	7
7. IANA Considerations	7
7.1. Mode of operation: MOPex	7
7.2. New options: MOPex and Capabilities	8
7.3. New Registry for Extended-MOP-value	8
7.4. New Registry for Capabilities Flags	8
8. Security Considerations	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9
Appendix A. Capability Handshake Example	9
Authors' Addresses	10

1. Introduction

RPL [RFC6550] specifies a proactive distance-vector based routing scheme. The protocol creates a DAG-like structure which operates with a given "Mode of Operation" (MOP) determining the minimal and mandatory set of primitives to be supported by all the participating nodes.

MOP as per [RFC6550] is a 3-bit value carried in DIO messages and is specific to the RPL Instance. The recipient of the DIO message can join the specified network as a router only when it can support the primitives as required by the mode of operation value. For example, in case of MOP=3 (Storing MOP with multicast support) the nodes can join the network as routers only when they can handle the DAO advertisements from the peers and manage routing tables. The 3-bit value is already exhausted and requires replenishment. This document introduces a mechanism to extend mode of operation values.

This document further adds a notion of capabilities using which the nodes in the network could inform its peers about its additional capabilities/features. This document highlights the differences of capabilities from that of Mode of operation and explains the necessity of it.

1.1. Requirements Language and Terminology

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

MOP: Mode of Operation. Identifies the mode of operation of the RPL Instance as administratively provisioned at and distributed by the DODAG root.

MOPex: Extended MOP: This document extends the MOP values over a bigger range. This extension of MOP is called MOPex.

Capabilities: Additional features or capabilities which might possibly be optional that are supported by the node.

DAO: DODAG Advertisement Object. An RPL message used to advertise the target information in order to establish routing adjacencies.

DIO: DODAG Information Object. An RPL message initiated by the root and is used to advertise the network configuration information.

Current parent: Parent 6LR node before switching to the new path.

NPDAO: No-Path DAO. A DAO message which has target with lifetime 0.

MOPex: MOP extension as defined in this document.

This document uses terminology described in [RFC6550]. For the sake of readability all the known relevant terms are repeated in this section.

2. Requirements for this document

Following are the requirements considered for this documents:

REQ1: MOP extension. Current MOP of 3-bit is fast depleting. An MOP extension needs to extend the possibility of adding new MOPs in the future.

REQ2: Backwards compatibility. The new options and new fields in the DIO message should be backward compatible i.e. if there

are nodes which support old MOPs they could still operate in their own instances.

REQ3: Optional capabilities handshake. Capabilities are features, possibly optional, which could be handshaked between the nodes and the root within an RPL Instance.

REQ4: Capabilities handshake could be optionally added with existing MOPs. Capabilities been optional in nature could be put to use with existing MOPs. Capabilities and MOP-extension is mutually independent i.e. a DIO can have a capabilities option, MOP-extension option or both in the same message.

3. Extended MOP Control Message Option

This document reserves existing MOP value 7 to be used as an extender. DIO messages with MOP value of 7 may refer to the Extended MOP (MOPex) option in the DIO message.

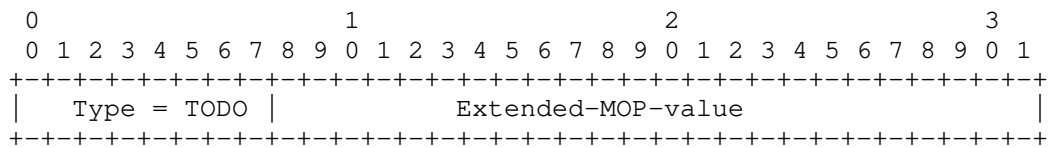


Figure 1: Extended MOP Option

3.1. Final MOP

An implementation supporting this document MUST calculate the final MOP value as the sum of base MOP (as supported in Section 6.3.1. of [RFC6550]) plus the MOPex value. Thus if the MOPex value is 0, it means the final MOP is 7 since the base MOP in this case will be set to 7.

Base MOP	MOPex	Final MOP
0	NA	0
1	NA	1
:	:	:
6	NA	6
7	0	7
7	1	8
7	2	9
:	:	:

Table 1: Final MOP calculation

3.2. Handling MOPex

If the MOPex option is absent in the DIO whose MOP is 7, then the MOPex value can be assumed to be zero (thus the final MOP in this case will be 7). The MOPex value should be referred only if the base MOP value is 7 and if the MOPex option is present. In case the base MOP is 7 and if the MOPex option is present, then the implementation MUST calculate the final MOP after considering the value in MOPex.

Note that [RFC6550] allows the node who does not support the received MOP to still join the network as a leaf node. This semantic continues to be true even in case of MOPex.

4. Capabilities

Currently RPL specification does not have a mechanism whereby a node can signal the set of features that are available on its end. Such a mechanism could help the root to advertise its capabilities and in response also determine some advanced information about the capabilities of the joining nodes. The Mode of Operation field in RPL mandates the operational requirement and does not allow loose coupling of additional capabilities. This document defines Capabilities as additional features which could be supported by the nodes and handshaked as part of RPL signaling. Capabilities are embedded as RPL control message option as defined Section 6.7 of [RFC6550] in the base messages of DIO, DAO and DAO-ACK signaling.

Note that capabilities and MOPex are mutually exclusive and it is possible for an implementation to support either or both of the options.

4.1. Capability Control Message Option

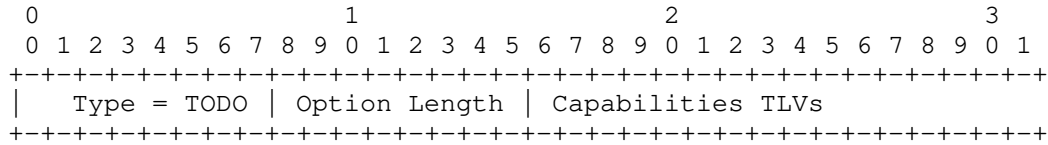


Figure 2: Capabilities Option

Multiple capabilities could be sent in the same message. The length field allows the message parser to skip the capability TLV parsing.

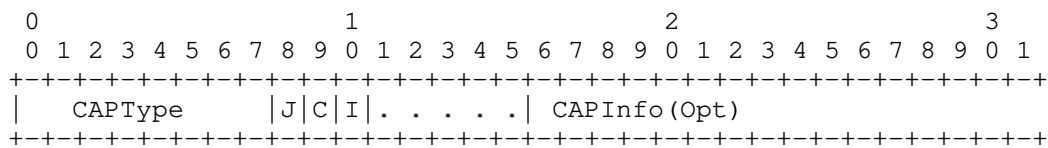


Figure 3: Capabilities TLV

Every capability is identified by its type and it may have an optional Capability Info. Note that a given capability may or may not be disseminated with additional information depending on the 'I' flag.

J = Join only as leaf if capability not understood

C = Copy capability to children

I = Cap Info present

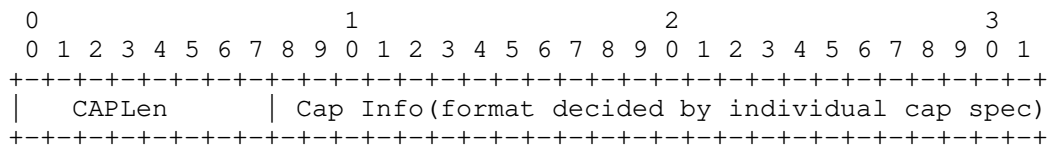


Figure 4: Capabilities Info

Capability Information provides additional information for the given capability. The format of this field should be defined as part the individual capability specification and is beyond the scope of this document. This document provides a container format for carrying the capability and its context information.

4.2. Capabilities Handshake

The root node could advertise the set of capabilities it supports in the DIO message. A node could take advantage of the knowledge that the root supports a particular capability. Similarly a node could advertise its capabilities in the DAO message using the capability control message option defined in this document. Capabilities advertised by non-root nodes are strictly a subset of the capabilities advertised by the root.

In storing MOP, the DAO message from the 6LR could contain multiple target options. The targets of the capabilities option are indicated by one or more Target options that precede the Capabilities Option. This handling is similar to the Transit Information Option as supported in Section 6.7.8. of [RFC6550].

5. Implementations Consideration

The MOP-extension could cause 3-byte increase in memory in the RPL-Instance. The MOP field in the RPL-Instance needs to be upgraded to a 32 bit integer.

[RFC6550], it was possible to discard an unsupported DIO-MOP just by inspecting the base message. With this document, the MOPex is a different control message option and thus the discarding of the DIO message could happen after inspecting the message options.

A node in storing MOP could independently construct a DAO message with target options containing its child/sub-children. Thus with capabilities it needs to reconstruct the capabilities field as well. This may result in increase in the memory requirement on per routing-entry basis.

6. Acknowledgements

Thanks to Georgios Papadopoulos for the review and feedback.

7. IANA Considerations

7.1. Mode of operation: MOPex

IANA is requested to assign a new Mode of Operation, named "MOPex" for MOP extension under the RPL registry. The value of 7 is to be assigned from the "Mode of Operation" space [RFC6550]

Value	Description	Reference
7	MOPex	This document

Mode of Operation

7.2. New options: MOPex and Capabilities

Two new entries are required for new supporting new options "MOPex", "Capabilities" from the "RPL Control Message Options" space [RFC6550].

Value	Meaning	Reference
TBD1	MOPex	This document
TBD2	Capabilities	This document

New options

7.3. New Registry for Extended-MOP-value

IANA is requested to create a registry for the extended-MOP-value (MOPex). This registry should be located in TODO. New MOPex values may be allocated only by an IETF review. Currently no values are defined by this document. Each value is tracked with the following qualities:

- o MOPex value
- o Description
- o Defining RFC

7.4. New Registry for Capabilities Flags

IANA is requested to create a registry for the Capabilities flags as described in Section 4 of this document. This registry should be located in TODO. New Capabilities flags may be allocated only by an IETF review. Currently no flags are defined by this document. Each value is tracked with the following qualities:

- o Flag
- o Description

- o Defining RFC

8. Security Considerations

The options defined in this document are carried in the base message objects as defined in [RFC6550]. The RPL control message options are protected by the same security mechanisms that protect the base messages.

Capabilities flag can reveal that the node has been upgraded or is running a old feature set. This document assumes that the base messages that carry these options are protected by RPL security mechanisms and thus are not visible to a malicious node.

9. References

9.1. Normative References

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

9.2. Informative References

- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.

Appendix A. Capability Handshake Example

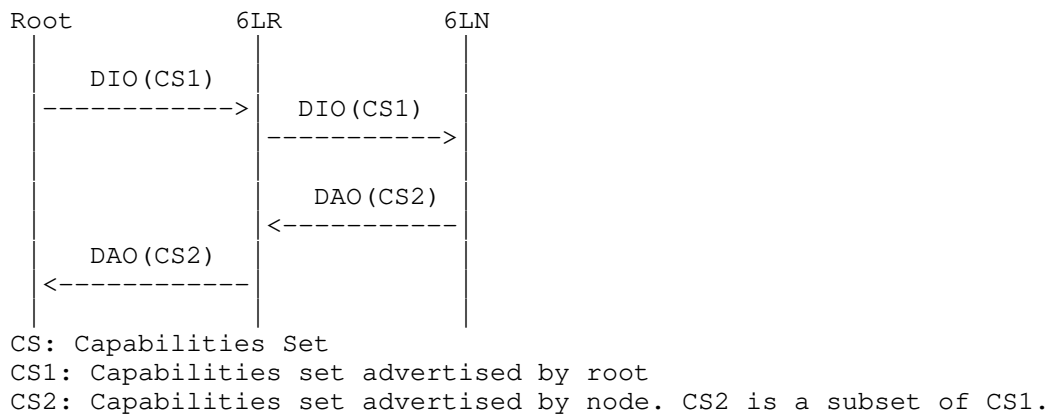


Figure 5: Capabilities Option

Authors' Addresses

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

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

Pascal Thubert
 Cisco Systems, Inc
 Building D
 45 Allee des Ormes - BP1200
 MOUGINS - Sophia Antipolis 06254
 France

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

Michael Richardson
 Sandelman Software Works
 Email: mcr+ietf@sandelman.ca

ROLL
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2020

R. Jadhav, Ed.
R. Sahoo
Y. Wu
Huawei
September 20, 2019

RPL Observations
draft-ietf-roll-rpl-observations-02

Abstract

This document describes RPL protocol design issues, various observations and possible consequences of the design and implementation choices.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Motivation	3
2. Introduction	3
2.1. Requirements Language and Terminology	3
3. DTSN increment in storing MOP	4
3.1. Deliberations	5
4. DAO retransmission and use of DAO-ACK in storing MOP	5
4.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK	6
4.2. Problems with hop-by-hop DAO-ACK	6
4.3. Problems with end-to-end DAO-ACK	6
4.4. Deliberations	6
4.5. Implementation Notes	7
5. Handling resource unavailability	7
5.1. Deliberations	7
6. Handling aggregated targets	7
6.1. Deliberations	8
7. RPL Transit Information in DAO	8
7.1. Deliberations	8
8. Upgrades or Extensions to RPL protocol	9
9. Asymmetric Links and RPL	9
10. Adjacencies probing with RPL	9
10.1. Deliberations	10
11. Control Options eliding mechanism in RPL	10
12. Managing persistent variables across node reboots	10
12.1. Persistent storage and RPL state information	10
12.2. Lollipop Counters	11
12.3. RPL State variables	12
12.3.1. DODAG Version	12
12.3.2. DTSN field in DIO	12
12.3.3. PathSequence	13
12.4. State variables update frequency	13
12.5. Deliberations	13
12.6. Implementation Notes	14
13. Capabilities and its role in RPL	14
13.1. Handshaking node capabilities	14
13.2. How do Capabilities differ from MOP and Configuration Option?	15
13.3. Deliberations	15
14. RPL under-specification	15
15. Acknowledgements	15
16. IANA Considerations	16
17. Security Considerations	16
18. References	16
18.1. Normative References	16
18.2. Informative References	17
Appendix A. Additional Stuff	17

Authors' Addresses	17
--------------------	----

1. Motivation

The primary motivation for this draft is to enlist different issues with RPL operation and invoke a discussion within the working group. This draft by itself is not intended for RFC tracks but as a WG discussion track. This draft may in turn result in other work items taken up by the WG which may improvise on the issues mentioned herewith.

2. Introduction

RPL [RFC6550] specifies a proactive distance-vector routing scheme designed for LLNs (Low Power and Lossy Networks). RPL enables the network to be formed as a DODAG and supports storing mode and non-storing mode of operations. Non-storing mode allows reduced memory resource usage on the nodes by allowing non-BR nodes to operate without managing a routing table and involves use of source routing by the 6LBR to direct the traffic along a specific path. In storing mode of operation intermediate routers maintain routing tables.

This work aims to highlight various issues with RPL which makes it difficult to handle certain scenarios. This work will highlight such issues in context to RPL's mode of operations (storing versus non-storing). There are cases where RPL does not provide clear rules and implementations have to make their choices hindering interoperability and performance.

[I-D.clausen-lln-rpl-experiences] provides some interesting points. Some sections in this draft may overlap with some observations in [clausen], but this is been done to further extend some scenarios or observations. It is highly encouraged that readers should also visit [I-D.clausen-lln-rpl-experiences] for other insights. Regardless, this draft is self-sufficient in a way that it does not expect to have read [clausen-draft].

2.1. Requirements Language and Terminology

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

NS-MOP = RPL Non-storing Mode of Operation

S-MOP = RPL Storing Mode of Operation

This document uses terminology described in [RFC6550] and [RFC6775].

3. DTSN increment in storing MOP

DTSN increment has major impact on the overall RPL control traffic and on the efficiency of downstream route update. DTSN is sent as part of DIO message and signals the downstream nodes to trigger the target advertisement. The 6LR needs to decide when to update the DTSN and usually it should do it in a conservative way. The DTSN update mechanism determines how soon the downward routes are established along the new path. RPL specifications does not provide any clear mechanism on how the DTSN update should happen in case of storing mode.

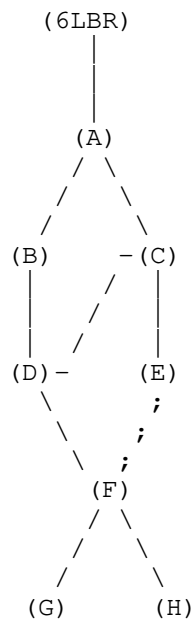


Figure 1: Sample topology

Consider example topology shown in Figure 1, assume that node D switches the parent from node B to C. Ideally the downstream nodes D and its sub-children should send their target advertisement to the new path via node C. To achieve this result in an efficient way is a challenge. Incrementing DTSN is the only way to trigger the DAO on downstream nodes. But this trigger should be sent not only on the first hop but to all the grand-child nodes. Thus DTSN has to be incremented in the complete sub-DODAG rooted at node D thus resulting in DIO/DAO storm along the sub-DODAG. This is specifically a big issue in high density networks where the metric deterioration might happen transiently even though the signal strength is good.

The primary implementation issue is whether a child node increment its own DTSN when it receives DTSN update from its parent node? This would result in DAO-updates in the sub-DODAG, thus the cost could be very high. If not incremented it may result in serious loss of connectivity for nodes in the sub-DODAG.

3.1. Deliberations

- (1) In S-MOP, should the child node increment its DTSN on seeing that its preferred parent has updated its DTSN?
- (2) What are rules for DTSN increment for S-MOP, which multiple implementations can follow thus allowing consistent performance across different implementations?

4. DAO retransmission and use of DAO-ACK in storing MOP

[RFC6550] has an optional DAO-ACK mechanism using which an upstream parent confirms the reception of a DAO from the downstream child. In case of storing mode, the DAO is addressed to the immediate hop upstream parent resulting in DAO-ACK from the parent. There are two implementations possible:

- (1) Hop-by-hop ACK: A parent responds with a DAO-ACK immediately after receiving the DAO.
- (2) End-to-End ACK: A node waits for the upstream parent to send DAO-ACK to respond with a DAO-ACK downstream. The upstream parent may do as many attempts to successfully send this DAO upstream. In other words, the parent node accepts the responsibility of sending the DAO upstream till the point it is ACKed the moment it responds back with its own ACK to the child.

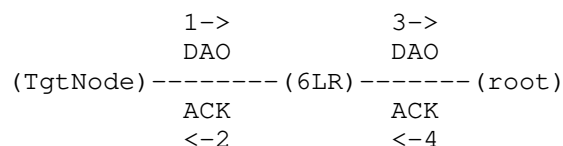


Figure 2: Hop-by-hop DAO-ACK

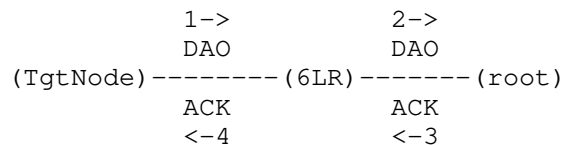


Figure 3: End-to-End DAO-ACK

4.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK

Lot of application traffic patterns requires that the bidirectional path be established between the target node and the root. A typical example is that COAP request with ACK bit set would require an acknowledgement from the end receiver and thus warrants bidirectional path establishment. It is imperative that the target node first ascertains whether such a bidirectional path is established before initiating such application traffic. In case of non-storing MOP, the DAO-ACK works perfectly fine to ascertain such bidirectional connectivity since it is an indication that the root which usually is the direct destination of the DAO has received the DAO. But in case of storing MOP, things are more complicated since DAO is sent hop-by-hop and the DAO-ACK semantics are not clear enough as per the current specification. As mentioned in above section, an implementation can choose to implement hop-by-hop ACK or end-to-end ACK.

4.2. Problems with hop-by-hop DAO-ACK

The primary issue with this mode is that target node cannot ascertain bidirection path connectivity on the reception of the DAO-ACK.

4.3. Problems with end-to-end DAO-ACK

In this case, it is possible for the target node to ascertain if the DAO has indeed reached the root since the reception of DAO-ACK on target node confirms this. However there is extra state information that needs to be maintained on the 6LRs on behalf of all the child nodes. Also it is very difficult for the target node to ascertain a timer value to decide whether the DAO transmission has failed to reach the root.

4.4. Deliberations

- (1) How should an implementation interpret the DAO-ACK semantics?
- (2) What is the best way for the target node to know that the end to end bidirectional path is successfully installed or updated? In NS-MOP, the DAO-ACK provides a clear way to do this. Can the same be achieved for storing-MOP?
- (3) What happens if the DAO-ACK with Status!=0 is responded by ancestor node?
- (4) How to selectively NACK subset of targets in case target containers are aggregated?

4.5. Implementation Notes

Current RPL open source implementations have both types of DAO-ACK implementations. For e.g. RIOT supports hop-by-hop DAO-ACK. Contiki older versions supported hop-by-hop ACK but the recent version have changed to end-to-end ACK implementation.

The sequence of sending no-path DAO and DAO matters when updating the routing adjacencies on a parent switch. If an implementation chooses to send no-path DAO before DAO then it results in significantly more overhead for route invalidation. This is because no-path DAO would traverse all the way up to the BR clearing the routes on the way. In case there is a common ancestor post which the old and new path remains same then it is better to send regular DAO first thus limiting the propagation of subsequent no-path DAO till this common ancestor.

5. Handling resource unavailability

The nodes in the constrained networks have to maintain various records such as neighbor cache entries and routing entries on behalf of other targets to facilitate packet forwarding. Because of the constrained nature of the devices the memory available may be very limited and thus the path selection algorithm may have to take into consideration such resource constraints as well.

RPL currently does not have any mechanism to advertise such resource indicator metrics. The primary tables associated with RPL are routing table and the neighbor cache. Even though neighbor cache is not directly linked with RPL protocol, the maintenance of routing adjacencies results in updates to neighbor cache.

5.1. Deliberations

Is it possible to know that an upstream parent/ancestor cannot hold enough routing entries and thus this path should not be used?

Is it possible to know that an upstream parent cannot hold any more neighbor cache entry and thus this upstream parent should not be used?

6. Handling aggregated targets

RPL allows and defines specific procedures so as to aid target aggregation in DAO. Having said that, the specification does not mandate use of aggregated targets nor does it make any comment on whether a receiving node needs to handle it. Target aggregation is an useful tool and especially helps with link layer technologies that

does not suffer from low MTUs such as PLC. Even if the implementation does not support aggregating targets, it should atleast mandate reception of aggregated targets in DAO.

RPL has a mechanism currently to ACK the DAO but it does not have a mechanism to ACK the target container. Thus in case of aggregated targets in the DAO, if the subset of the targets fail then it is impossible for the DAO-ACK to signal this to the DAO sender.

6.1. Deliberations

Even if the implementation does not support aggregating targets, should it atleast mandate reception and handling of aggregated targets in DAO?

There is a good scope for compressing aggregated targets which can significantly reduce the RPL control overhead.

How to selectively NACK subset of targets in case target containers are aggregated?

The DEFAULT_DAO_DELAY of 1sec does not help much with aggregation. The upstream parent nodes should wait for more time then the child nodes so as to effectively aggregate. Can we have DEFAULT_DAO_DELAY a function of the level/rank the node is at?

7. RPL Transit Information in DAO

RPL allows associating a target or set of targets with a Transit information container which contains attributes for a path to one or more destinations identified by the set of targets. In case of NS-MOP, the transit Information will contain the all critical Parent Address which allows the common ancestor usually the root to identify the source route header for the target node. The Transit Information also contains other information such as Path Sequence and Path Lifetime which are critical for maintaining route adjacencies.

RPL however does not mandate the use of Transit Information container for targets.

7.1. Deliberations

Is it ok to let implementations decide on the inclusion of Transit Information container?

Is it possible to achieve interop without mandating use of Transit Information Container?

If the Transit Information container is sent, should the handling of PathSequence be mandated?

8. Upgrades or Extensions to RPL protocol

RPL extensibility is highly desirable and is controlled by protocol elements within the messaging framework. In the pursuit to keep the signalling overhead less, RPL specification has been restricting in its approach to extend its field ranges, thus in some cases putting extensibility at stakes. Consider for example, the mode of operation bits which is three bits in the RPL specification. These bits are already saturated and it may be difficult to add major upgrades without extending these bits.

9. Asymmetric Links and RPL

Section 3.1 of [I-D.ietf-intarea-adhoc-wireless-com] explains asymmetric link characteristics and what it takes for a protocol to support asymmetric links. RPL depends on bi-directional links for control even though near-perfect symmetry is not expected. The implication of this is that the upstream and downstream path remains same within a given RPL instance for any pair of nodes. There are following questions sprouting of this design:

- (1) Is it possible to detect asymmetric links?
- (2) In the presence of asymmetric links what is the impact on the control overhead and is there a way to possibly mitigate or alleviate any negative impact?

[I-D.ietf-roll-aodv-rpl] defines a mechanism to use a pair of instances which are coupled. This allows disjoint upstream and downstream paths between pair of nodes assuming that the link asymmetry is detected using some outside techniques. The link assumes that the link asymmetry is already known to the nodes in the form of static configuration. In case of 6tisch networks, the availability of transmission slots information can be used to identify link asymmetry. The challenge with regards to detecting link asymmetry arises from scenarios where, for example, the nodes transmit with unequal power levels.

10. Adjacencies probing with RPL

RPL avoids periodic hello messaging as compared to other distance-vector protocols. It uses trickle timer based mechanism to update configuration parameters. This significantly reduces the RPL control overhead. One of the fallout of this design choice is that, in the

absence of regular traffic, the adjacencies could not be tested and repaired if broken.

RPL provides a mechanism in the form of unicast DIS to query a particular node for its DIO. A node receiving a unicast DIS MUST respond with a unicast DIO with Configuration Option. This mechanism could as well be made use of for probing adjacencies and certain implementations such as Contiki uses this. The periodicity of the probing is implementation dependent, but the node is expected to invoke probing only when

- (1) There is no data traffic based on which the links could be tested.
- (2) There is no L2 feedback. In some case, L2 might provide periodic beacons at link layer and the absence of beacons could be used for link tests.

10.1. Deliberations

- (1) Should the probing scheme be standardized? In some cases using multicast based probing may prove advantageous.
- (2) In some cases using multicast based probing may prove advantageous. Currently RPL does not have multicast based probing. Multicast DIS/DIO may not be suitable for probing because it could possibly lead to change of states.

11. Control Options eliding mechanism in RPL

RPL configuration changes are rare and thus various configuration options may not change over a long period of time. RPL provides a way for the configuration options to be elided but there are no clear guidelines on how the eliding should be handled. In the absence of such guidelines, it is possible that certain nodes may end up using stale configuration in the event of transient link failures.

12. Managing persistent variables across node reboots

12.1. Persistent storage and RPL state information

Devices are required to be functional for several years without manual maintenance. Usually battery power consumption is considered key for operating the devices for several (tens of) years. But apart from battery, flash memory endurance may prove to be a lifetime bottleneck in constrained networks. Endurance is defined as maximum number of erase-write cycles that a NAND/NOR cell can undergo before losing its 'gauranteed' write operation. In some cases (cheaper

NAND-MLC/TLC), the endurance can be as less as 2K cycles. Thus for e.g. if a given cell is written 5 times a day, that NAND-flash cell assuming an endurance of 10K cycles may last for less than 6 years.

Wear leveling is a popular technique used in flash memory to minimize the impact of limited cell endurance. Wear leveling works by arranging data so that erasures and re-writes are distributed evenly across the medium. The memory sectors are over-provisioned so that the writes are distributed across multiple sectors. Many IoT platforms do not necessarily consider this over-provisioning and usually provision the memory only to what is required. Some scenarios such as street-lighting may not require the application layer to write any information to the persistent storage and thus the over-provisioning is often ignored. In such cases if the network stack ends up using persistent storage for maintaining its state information then it becomes counter-productive.

In a star topology, the amount of persistent data write done by network protocols is very limited. But ad-hoc networks employing routing protocols such as RPL assume certain state information to be retained across node reboots. In case of IoT devices this storage is mostly floating gate based NAND/NOR based flash memory. The impact of loss of this state information differs depending upon the type (6LN/6LR/6LBR) of the node.

12.2. Lollipop Counters

[RFC6550] Section 7.2. explains sequence counter operation defining lollipop [Perlman83] style counters. Lollipop counters specify mechanism in which even if the counter value wraps, the algorithm would be able to tell whether the received value is the latest or not. This mechanism also helps in "some cases" to recover from node reboot, but is not foolproof.

Consider an e.g. where Node A boots up and initialises the seqcnt to 240 as recommended in [RFC6550]. Node A communicates to Node B using this seqcnt and node B uses this seqcnt to determine whether the information node A sent in the packet is latest. Now lets assume, the counter value reaches 250 after some operations on Node A, and node B keeps receiving updated seqcnt from node A. Now consider that node A reboots, and since it reinitializes the seqcnt value to 240 and sends the information to node B (who has seqcnt of 250 stored on behalf of node A). As per section 7.2. of [RFC6550], when node B receives this packet it will consider the information to be old (since $240 < 250$).

A	B	Output
240	240	A<B, old
240	241	A<B, old
240	::	A<B, old
240	256	A<B, old
240	0	A<B, new
240	1	A>B, new
240	::	A>B, new
240	127	A>B, new

Default values for lollipop counters considered from [RFC6550]
Section 7.2.

Table 1: Example lollipop counter operation

Based on this figure, there is dead zone (240 to 0) in which if A operates after reboot then the seqcnt will always be considered smaller. Thus node A needs to maintain the seqcnt in persistent storage and reuse this on reboot.

12.3. RPL State variables

The impact of loss of RPL state information differs depending upon the node type (6LN/6LR/6LBR). Following sections explain different state variables and the impact in case this information is lost on reboot.

12.3.1. DODAG Version

The tuple (RPLInstanceID, DODAGID, DODAGVersionNumber) uniquely identifies a DODAG Version. DODAGVersionNumber is incremented everytime a global repair is initiated for the instance (global or local). A node receiving an older DODAGVersionNumber will ignore the DIO message assuming it to be from old DODAG version. Thus a 6LBR node (and 6LR node in case of local DODAG) needs to maintain the DODAGVersionNumber in the persistent storage, so as to be available on reboot. In case the 6LBR could not use the latest DODAGVersionNumber the implication are that it won't be able to recover/re-establish the routing table.

12.3.2. DTSN field in DIO

DTSN (Destination advertisement Trigger Sequence Number) is a DIO message field used as part of procedure to maintain Downward routes. A 6LBR/6LR node may increment a DTSN in case it requires the

downstream nodes to send DAO and thus update downward routes on the 6LBR/6LR node. In case of RPL NS-MOP, only the 6LBR maintains the downward routes and thus controls this field update. In case of S-MOP, 6LRs additionally keep downward routes and thus control this field update.

In S-MOP, when a 6LR node switches parent it may have to issue a DIO with incremented DTSN to trigger downstream child nodes to send DAO so that the downward routes are established in all parent/ancestor set. Thus in S-MOP, the frequency of DTSN update might be relatively high (given the node density and hysteresis set by objective function to switch parent).

12.3.3. PathSequence

PathSequence is part of RPL Transit Option, and associated with RPL Target option. A node which owns a target address can associate a PathSequence in the DAO message to denote freshness of the target information. This is especially useful when a node uses multiple paths or multiple parents to advertise its reachability.

Loss of PathSequence information maintained on the target node can result in routing adjacencies been lost on 6LRs/6LBR/6BBR.

12.4. State variables update frequency

State variable	Update frequency	Impacts node type
DODAGVersionNumber	Low	6LBR, 6LR(local DODAG)
DTSN	High(SM), Low(NSM)	6LBR, 6LR
PathSequence	High(SM), Low(NSM)	6LR, 6LN

Low=<5 per day, High=>5 per day; SM=Storing MOP, NSM=Non-Storing MOP

Table 2: RPL State variables

12.5. Deliberations

- (1) Is it possible that RPL reduces the use of persistent storage for maintaining state information?
- (2) In most cases, the node reboots will happen very rarely. Thus doing a persistent storage book-keeping for handling node reboot might not make sense. Is it possible to consider signaling (especially after the node reboots) so as to avoid maintaining

this persistent state? Is it possible to use one-time on-reboot signalling to recover some state information?

- (3) It is necessary that RPL avoids using persistent storage as far as possible. Ideally, extensions to RPL should consider this as a design requirement especially for 6LR and 6LN nodes. DTSN and PathSequence are the primary state variables which have major impact.

12.6. Implementation Notes

An implementation should use a random DAOSequence number on reboot so as to avoid a risk of reusing the same DAOSequence on reboot. Regardless the sequence counter size of 8bits does not provide much guarantees towards choosing a good random number. A parent node will not respond with a DAO-ACK in case it sees a DAO with the same previous DAOSequence.

Write-Before-Use: The state information should be written to the flash before using it in the messaging. If it is done the other way, then the chances are that the node power downs before writing to the persistent storage.

13. Capabilities and its role in RPL

RPL is a distributed protocol and it requires that the participating nodes agree on basic set of primitives to follow. RPL currently handles this using MOP (Mode of Operation) bits in the DIO. MOP bits inform the nodes the basic mode of operation a node MUST support to join the Instance as a 6LR. The MOP is decided and advertised by the root of the RPL Instance. A node not supporting the given MOP may still join the Instance as a leaf node or 6LN.

RPL further uses DIO Configuration Option to advertise the configuration each node needs to use (for e.g., for trickle timer).

13.1. Handshaking node capabilities

Currently there exist no mechanism to handshake capabilities of the root or 6LRs or 6LNs. If a feature is optional and is supported by 6LRs/6LNs then currently there exists no mechanism to signal it. There are several RPL extension proposals which are possibly optional features. Root needs to know if the 6LR/6LN supports these optional features to enable the extension in that path context. Similarly 6LRs and 6LNs need to know whether the root supports certain extensions that it can make use of.

13.2. How do Capabilities differ from MOP and Configuration Option?

Unlike MOP and Configuration Option which are issued by the root of the Instance, Capabilities can be issued by any node. A 6LN/6LR node can advertise its capabilities such that those can be seen by intermediate 6LRs and the root of the Instance.

13.3. Deliberations

- (1) Is it possible for leaf nodes to advertise their set of capabilities, which can be used by root and/or intermediate 6LRs to make run time decisions?
- (2) How should these capabilities be carried? Should it be carried in DAO/DIO/DAO-ACK?
- (3) Should the definition of capabilities be same in both directions (upstream/downstream)?

14. RPL under-specification

- (a) PathSequence: Is it mandatory to use PathSequence in DAO Transit container? RPL mentions that a 6LR/6LBR hosting the routing entry on behalf of target node should refresh the lifetime on reception of a new Path Sequence. But RPL does not necessarily mandate use of Path Sequence. Most of the open source implementation [RIOT] [CONTIKI] currently do not issue Path Sequence in the DAO message.
- (b) Target Container aggregation in DAO: RPL allows multiple targets to be aggregated in a single DAO message and has introduced a notion of DelayDAO using which a 6LR node could delay its DAO to enable such aggregation. But RPL does not have clear text on handling of aggregated DAOs and thus it hinders interoperability.
- (c) DTSN Update: RPL does not clearly define in which cases DTSN should be updated in case of storing mode of operation. More details for this are presented in Section 3.

15. Acknowledgements

Many thanks to Pascal Thubert for hallway chats and for helping understand the existing design rationales. Thanks to Michael Richardson for Unstrung RPL implementation rationale. Thanks to ML discussions, in particular (<https://www.ietf.org/mail-archive/web/roll/current/msg09443.html>).

16. IANA Considerations

This memo includes no request to IANA.

17. Security Considerations

This is an information draft and does not add any changes to the existing specifications.

18. References

18.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <<https://www.rfc-editor.org/info/rfc6551>>.
- [RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <<https://www.rfc-editor.org/info/rfc6552>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks", RFC 6997, DOI 10.17487/RFC6997, August 2013, <<https://www.rfc-editor.org/info/rfc6997>>.

18.2. Informative References

- [I-D.clausen-lln-rpl-experiences]
Clausen, T., Verdiere, A., Yi, J., Herberg, U., and Y. Igarashi, "Observations on RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-clausen-lln-rpl-experiences-11 (work in progress), March 2018.
- [I-D.ietf-intarea-adhoc-wireless-com]
Baccelli, E. and C. Perkins, "Multi-hop Ad Hoc Wireless Communication", draft-ietf-intarea-adhoc-wireless-com-02 (work in progress), July 2016.
- [I-D.ietf-roll-aodv-rpl]
Anamalamudi, S., Zhang, M., Perkins, C., Anand, S., and B. Liu, "Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs)", draft-ietf-roll-aodv-rpl-07 (work in progress), April 2019.
- [Perlman83]
Perlman, R., "Fault-Tolerant Broadcast of Routing Information", North-Holland Computer Networks, Vol.7, December 1983.

Appendix A. Additional Stuff

Authors' Addresses

Rahul Arvind Jadhav (editor)
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

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

Rabi Narayan Sahoo
Huawei
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

Phone: +91-080-49160700
Email: rabinarayans@huawei.com

Yuefeng Wu
Huawei
No.101, Software Avenue, Yuhuatai District,
Nanjing, Jiangsu 210012
China

Phone: +86-15251896569
Email: wuyuefeng@huawei.com

ROLL
Internet-Draft
Intended status: Standards Track
Expires: 2 June 2022

R.A. Jadhav, Ed.

R.N. Sahoo
Juniper
Y. Wu
Huawei
29 November 2021

RPL Observations
draft-ietf-roll-rpl-observations-07

Abstract

This document describes RPL protocol design issues, various observations and possible consequences of the design and implementation choices.

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Motivation	3
2. Introduction	3
2.1. Requirements Language and Terminology	3
3. DTSN increment in storing MOP	4
3.1. Deliberations	5
4. DAO retransmission and use of DAO-ACK in storing MOP	5
4.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK	6
4.2. Problems with hop-by-hop DAO-ACK	6
4.3. Problems with end-to-end DAO-ACK	6
4.4. Deliberations	6
4.5. Implementation Notes	7
5. Interpreting Trickle Timer	7
6. Handling resource unavailability	8
6.1. Deliberations	8
7. Handling aggregated targets	9
7.1. Deliberations	9
8. RPL Transit Information in DAO	9
8.1. Deliberations	10
9. Upgrades or Extensions to RPL protocol	10
10. Path Control bits handling	10
11. Asymmetric Links and RPL	11
12. Adjacencies probing with RPL	12
12.1. Deliberations	12
13. Control Options eliding mechanism in RPL	12
14. Managing persistent variables across node reboots	12
14.1. Persistent storage and RPL state information	13
14.2. Lollipop Counters	13
14.3. RPL State variables	14
14.3.1. DODAG Version	15
14.3.2. DTSN field in DIO	15
14.3.3. PathSequence	15
14.4. State variables update frequency	16
14.5. Deliberations	16
14.6. Implementation Notes	16
15. Capabilities and its role in RPL	17
15.1. Handshaking node capabilities	17
15.2. How do Capabilities differ from MOP and Configuration Option?	17
15.3. Deliberations	17
16. Backward Compatibility issues with RPL Options	18
17. RPL under-specification	18
18. Acknowledgements	18
19. IANA Considerations	19
20. Security Considerations	19
21. References	19

21.1. Normative References	19
21.2. Informative References	19
Appendix A. Additional Stuff	20
Authors' Addresses	20

1. Motivation

The primary motivation for this draft is to enlist different issues with RPL operation and invoke a discussion within the working group. This draft by itself is not intended for RFC tracks but as a WG discussion track. This draft may in turn result in other work items taken up by the WG which may improvise on the issues mentioned herewith.

2. Introduction

RPL [RFC6550] specifies a proactive distance-vector routing scheme designed for LLNs (Low Power and Lossy Networks). RPL enables the network to be formed as a DODAG and supports storing mode and non-storing mode of operations. Non-storing mode allows reduced memory resource usage on the nodes by allowing non-BR nodes to operate without managing a routing table and involves use of source routing by the Root to direct the traffic along a specific path. In storing mode of operation intermediate routers maintain routing tables.

This work aims to highlight various issues with RPL which makes it difficult to handle certain scenarios. This work will highlight such issues in context to RPL's mode of operations (storing versus non-storing). There are cases where RPL does not provide clear rules and implementations have to make their choices hindering interoperability and performance.

[I-D.clausen-lln-rpl-experiences] provides some interesting points. Some sections in this draft may overlap with some observations in [clausen], but this is been done to further extend some scenarios or observations. It is highly encouraged that readers should also visit [I-D.clausen-lln-rpl-experiences] for other insights. Regardless, this draft is self-sufficient in a way that it does not expect to have read [clausen-draft].

2.1. Requirements Language and Terminology

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

NS-MOP = RPL Non-storing Mode of Operation

S-MOP = RPL Storing Mode of Operation

This document uses terminology described in [RFC6550] and [RFC6775].

3. DTSN increment in storing MOP

DTSN increment has major impact on the overall RPL control traffic and on the efficiency of downstream route update. DTSN is sent as part of DIO message and signals the downstream nodes to trigger the target advertisement. The 6LR needs to decide when to update the DTSN and usually it should do it in a conservative way. The DTSN update mechanism determines how soon the downward routes are established along the new path. RPL specifications does not provide any clear mechanism on how the DTSN update should happen in case of storing mode.

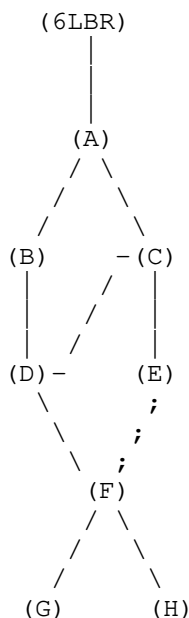


Figure 1: Sample topology

Consider example topology shown in Figure 1, assume that node D switches the parent from node B to C. Ideally the downstream nodes D and its sub-children should send their target advertisement to the new path via node C. To achieve this result in a efficient way is a challenge. Incrementing DTSN is the only way to trigger the DAO on downstream nodes. But this trigger should be sent not only on the first hop but to all the grand-child nodes. Thus DTSN has to be incremented in the complete sub-DODAG rooted at node D thus resulting

in DIO/DAO storm along the sub-DODAG. This is specifically a big issue in high density networks where the metric deterioration might happen transiently even though the signal strength is good.

The primary implementation issue is whether a child node increment its own DTSN when it receives DTSN update from its parent node? This would result in DAO-updates in the sub-DODAG, thus the cost could be very high. If not incremented it may result in serious loss of connectivity for nodes in the sub-DODAG.

3.1. Deliberations

- (1) In S-MOP, should the child node increment its DTSN on seeing that its preferred parent has updated its DTSN?
- (2) What are rules for DTSN increment for S-MOP, which multiple implementations can follow thus allowing consistent performance across different implementations?

4. DAO retransmission and use of DAO-ACK in storing MOP

[RFC6550] has an optional DAO-ACK mechanism using which an upstream parent confirms the reception of a DAO from the downstream child. In case of storing mode, the DAO is addressed to the immediate hop upstream parent resulting in DAO-ACK from the parent. There are two implementations possible:

- (1) Hop-by-hop ACK: A parent responds with a DAO-ACK immediately after receiving the DAO.
- (2) End-to-End ACK: A node waits for the upstream parent to send DAO-ACK to respond with a DAO-ACK downstream. The upstream parent may do as many attempts to successfully send this DAO upstream. In other words, the parent node accepts the responsibility of sending the DAO upstream till the point it is ACKed the moment it responds back with its own ACK to the child.

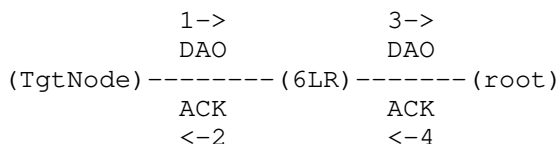


Figure 2: Hop-by-hop DAO-ACK

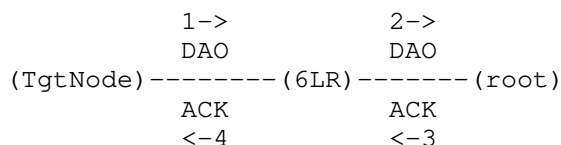


Figure 3: End-to-End DAO-ACK

4.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK

Lot of application traffic patterns requires that the bidirectional path be established between the target node and the root. A typical example is that COAP request with ACK bit set would require an acknowledgement from the end receiver and thus warrants bidirectional path establishment. It is imperative that the target node first ascertains whether such a bidirectional path is established before initiating such application traffic. In case of non-storing MOP, the DAO-ACK works perfectly fine to ascertain such bidirectional connectivity since it is an indication that the root which usually is the direct destination of the DAO has received the DAO. But in case of storing MOP, things are more complicated since DAO is sent hop-by-hop and the DAO-ACK semantics are not clear enough as per the current specification. As mentioned in above section, an implementation can choose to implement hop-by-hop ACK or end-to-end ACK.

4.2. Problems with hop-by-hop DAO-ACK

The primary issue with this mode is that target node cannot ascertain bidirection path connectivity on the reception of the DAO-ACK.

4.3. Problems with end-to-end DAO-ACK

In this case, it is possible for the target node to ascertain if the DAO has indeed reached the root since the reception of DAO-ACK on target node confirms this. However there is extra state information that needs to be maintained on the 6LRs on behalf of all the child nodes. Also it is very difficult for the target node to ascertain a timer value to decide whether the DAO transmission has failed to reach the root.

4.4. Deliberations

- (1) How should an implementation interpret the DAO-ACK semantics?

- (2) What is the best way for the target node to know that the end to end bidirectional path is successfully installed or updated? In NS-MOP, the DAO-ACK provides a clear way to do this. Can the same be achieved for storing-MOP?
- (3) What happens if the DAO-ACK with Status!=0 is responded by ancestor node?
- (4) How to selectively NACK subset of targets in case target options are aggregated?

4.5. Implementation Notes

Current RPL open source implementations have both types of DAO-ACK implementations. For e.g. RIOT supports hop-by-hop DAO-ACK. Contiki older versions supported hop-by-hop ACK but the recent version have changed to end-to-end ACK implementation.

The sequence of sending no-path DAO and DAO matters when updating the routing adjacencies on a parent switch. If an implementation chooses to send no-path DAO before DAO then it results in significantly more overhead for route invalidation. This is because no-path DAO would traverse all the way up to the BR clearing the routes on the way. In case there is a common ancestor post which the old and new path remains same then it is better to send regular DAO first thus limiting the propagation of subsequent no-path DAO till this common ancestor.

5. Interpreting Trickle Timer

Trickle algorithm defines a mechanism to reset the timer. Trickle timer reset is unlike regular periodic timers wherein the timer is simply reset to start again. Reset of trickle timer implies resetting the trickle back to Imin and starting with a new interval as mentioned in Section 4.2 of [RFC6206].

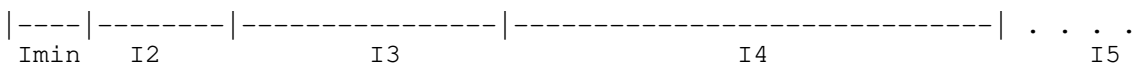


Figure 4: Trickle Timer Operation

The above figure shows an example of trickle intervals. An interval is double that of the previous interval size. Section 4.2. of [RFC6206] states that,

"If Trickle hears a transmission that is "inconsistent" and I is greater than Imin, it resets the Trickle timer. To reset the timer, Trickle sets I to Imin and starts a new interval as in step 2. If I

is equal to Imin when Trickle hears an "inconsistent" transmission, Trickle does nothing. Trickle can also reset its timer in response to external "events".

Thus if the trickle timer has advanced to subsequent intervals i.e., $\geq I2$, then a reset of trickle timer implies going back to Imin. However, if the trickle timer is currently in Imin and if it hears an inconsistent transmission then it does nothing.

In context to multicast DIS/DIO operation, this implies that if the DIO trickle timer is already at Imin and if the node hears a multicast DIS, then the timer does nothing. It MUST NOT reset the timer again in this case.

An implementation MUST never restart the timer within an interval. For e.g., in the above figure, if the timer is in interval I2, the implementation MUST never restart the timer to the beginning of the current interval i.e., I2. If the timer is in interval T2 and if the reset is to be done then the interval is set back to Imin. If the timer is already in Imin, then the reset should do nothing.

6. Handling resource unavailability

The nodes in the constrained networks have to maintain various records such as neighbor cache entries and routing entries on behalf of other targets to facilitate packet forwarding. Because of the constrained nature of the devices the memory available may be very limited and thus the path selection algorithm may have to take into consideration such resource constraints as well.

RPL currently does not have any mechanism to advertise such resource indicator metrics. The primary tables associated with RPL are routing table and the neighbor cache. Even though neighbor cache is not directly linked with RPL protocol, the maintenance of routing adjacencies results in updates to neighbor cache.

6.1. Deliberations

Is it possible to know that an upstream parent/ancestor cannot hold enough routing entries and thus this path should not be used?

Is it possible to know that an upstream parent cannot hold any more neighbor cache entry and thus this upstream parent should not be used?

7. Handling aggregated targets

RPL allows and defines specific procedures so as to aid target aggregation in DAO. Having said that, the specification does not mandate use of aggregated targets nor does it make any comment on whether a receiving node needs to handle it. Target aggregation is an useful tool and especially helps with link layer technologies that does not suffer from low MTUs such as PLC. Even if the implementation does not support aggregating targets, it should atleast mandate reception of aggregated targets in DAO.

RPL has a mechanism currently to ACK the DAO but it does not have a mechanism to ACK the target option. Thus in case of aggregated targets in the DAO, if the subset of the targets fail then it is impossible for the DAO-ACK to signal this to the DAO sender.

7.1. Deliberations

Even if the implementation does not support aggregating targets, should it atleast mandate reception and handling of aggregated targets in DAO?

There is a good scope for compressing aggregated targets which can significantly reduce the RPL control overhead.

How to selectively NACK subset of targets in case target options are aggregated?

The DEFAULT_DAO_DELAY of 1sec does not help much with aggregation. The upstream parent nodes should wait for more time then the child nodes so as to effectively aggregate. Can we have DEFAULT_DAO_DELAY a function of the level/rank the node is at?

8. RPL Transit Information in DAO

RPL allows associating a target or set of targets with a Transit Information Option which contains attributes for a path to one or more destinations identified by the set of targets. In case of NS-MOP, the transit Information will contain the all critical Parent Address which allows the common ancestor usually the root to identify the source route header for the target node. The Transit Information also contains other information such as Path Sequence and Path Lifetime which are critical for maintaining route adjacencies.

RPL however does not mandate the use of Transit Information Option for targets.

8.1. Deliberations

Is it ok to let implementations decide on the inclusion of Transit Information Option?

Is it possible to achieve interop without mandating use of Transit Information Option?

If the Transit Information Option is sent, should the handling of PathSequence be mandated?

9. Upgrades or Extensions to RPL protocol

RPL extensibility is highly desirable and is controlled by protocol elements within the messaging framework. In the pursuit to keep the signalling overhead less, RPL specification has been restricting in its approach to extend its field ranges, thus in some cases putting extensibility at stakes. Consider for example, the mode of operation bits which is three bits in the RPL specification. These bits are already saturated and it may be difficult to add major upgrades without extending these bits.

Addition of new Control Options or new RPL Codes almost certainly results in backward compatibility issues. RFC6550 clearly mentions that a message with an unknown RPL Code MUST be silently discarded. However, no explicit handling is suggested for unknown RPL control option types. In some cases, implementations simply copy-forward an unknown option as it is while in other cases the unknown option is stripped off before forwarding the message.

Deliberations:

- (1) What are the extensibility options RPL could implement? How much overhead would it incur?
- (2) Most of the extensions are in the form of new control options. Should RPL have a mechanism to only handle such extensions in a backward compatible but in a generic manner?

10. Path Control bits handling

RPL uses Path Control bits in the DAO's Transit Information Option for installing multiple downward routes to the nodes. These multiple routes could be used for reliability, latency or traffic load-balancing within a DAG. The path control bits are usable both in storing and non-storing mode of operation.

RFC6550 Section 9.9 bullet point 9 requires a mandatory setting of Path Control bits in all the unicast DAOs sent by the Target node. However, no existing implementation of RPL supports this. There is no reason for a network which only requires a single path to the root to mandatorily support path control bits.

Deliberations:

- (1) Should the mandatory clause for supporting Path Control Bits in RFC6550 Section 9.9 point 9 be removed?
- (2) Handling Path Control Bits may be complex. An implementation guideline explaining the use-cases and resource (memory requirements) assumptions would help implementors decide the utility of this technique.

11. Asymmetric Links and RPL

Section 3.1 of [I-D.ietf-intarea-adhoc-wireless-com] explains asymmetric link characteristics and what it takes for a protocol to support asymmetric links. RPL depends on bi-directional links for control even though near-perfect symmetry is not expected. The implication of this is that the upstream and downstream path remains same within a given RPL instance for any pair of nodes. There are following questions sprouting of this design:

- (1) Is it possible to detect asymmetric links?
- (2) In the presence of asymmetric links what is the impact on the control overhead and is there a way to possibly mitigate or alleviate any negative impact?

[I-D.ietf-roll-aodv-rpl] defines a mechanism to use a pair of instances which are coupled. This allows disjoint upstream and downstream paths between pair of nodes assuming that the link asymmetry is detected using some outside techniques. The link assumes that the link asymmetry is already known to the nodes in the form of static configuration. In case of 6tisch networks, the availability of transmission slots information can be used to identify link asymmetry. The challenge with regards to detecting link asymmetry arises from scenarios where, for example, the nodes transmit with unequal power levels.

12. Adjacencies probing with RPL

RPL avoids periodic hello messaging as compared to other distance-vector protocols. It uses trickle timer based mechanism to update configuration parameters. This significantly reduces the RPL control overhead. One of the fallout of this design choice is that, in the absence of regular traffic, the adjacencies could not be tested and repaired if broken.

RPL provides a mechanism in the form of unicast DIS to query a particular node for its DIO. A node receiving a unicast DIS MUST respond with a unicast DIO with Configuration Option. This mechanism could as well be made use of for probing adjacencies and certain implementations such as Contiki uses this. The periodicity of the probing is implementation dependent, but the node is expected to invoke probing only when

- (1) There is no data traffic based on which the links could be tested.
- (2) There is no L2 feedback. In some case, L2 might provide periodic beacons at link layer and the absence of beacons could be used for link tests.

12.1. Deliberations

- (1) Should the probing scheme be standardized? In some cases using multicast based probing may prove advantageous.
- (2) In some cases using multicast based probing may prove advantageous. Currently RPL does not have multicast based probing. Multicast DIS/DIO may not be suitable for probing because it could possibly lead to change of states.

13. Control Options eliding mechanism in RPL

RPL configuration changes are rare and thus various configuration options may not change over a long period of time. RPL provides a way for the configuration options to be elided but there are no clear guidelines on how the eliding should be handled. In the absence of such guidelines, it is possible that certain nodes may end up using stale configuration in the event of transient link failures.

14. Managing persistent variables across node reboots

14.1. Persistent storage and RPL state information

Devices are required to be functional for several years without manual maintenance. Usually battery power consumption is considered key for operating the devices for several (tens of) years. But apart from battery, flash memory endurance may prove to be a lifetime bottleneck in constrained networks. Endurance is defined as maximum number of erase-write cycles that a NAND/NOR cell can undergo before losing its 'gauranteed' write operation. In some cases (cheaper NAND-MLC/TLC), the endurance can be as less as 2K cycles. Thus for e.g. if a given cell is written 5 times a day, that NAND-flash cell assuming an endurance of 10K cycles may last for less than 6 years.

Wear leveling is a popular technique used in flash memory to minimize the impact of limited cell endurance. Wear leveling works by arranging data so that erasures and re-writes are distributed evenly across the medium. The memory sectors are over-provisioned so that the writes are distributed across multiple sectors. Many IoT platforms do not necessarily consider this over-provisioning and usually provision the memory only to what is required. Some scenarios such as street-lighting may not require the application layer to write any information to the persistent storage and thus the over-provisioning is often ignored. In such cases if the network stack ends up using persistent storage for maintaining its state information then it becomes counter-productive.

In a star topology, the amount of persistent data write done by network protocols is very limited. But ad-hoc networks employing routing protocols such as RPL assume certain state information to be retained across node reboots. In case of IoT devices this storage is mostly floating gate based NAND/NOR based flash memory. The impact of loss of this state information differs depending upon the type (6LN/6LR/6LBR) of the node.

14.2. Lollipop Counters

[RFC6550] Section 7.2. explains sequence counter operation defining lollipop [Perlman83] style counters. Lollipop counters specify mechanism in which even if the counter value wraps, the algorithm would be able to tell whether the received value is the latest or not. This mechanism also helps in "some cases" to recover from node reboot, but is not foolproof.

Consider an e.g. where Node A boots up and initialises the seqcnt to 240 as recommended in [RFC6550]. Node A communicates to Node B using this seqcnt and node B uses this seqcnt to determine whether the information node A sent in the packet is latest. Now lets assume, the counter value reaches 250 after some operations on Node A, and

node B keeps receiving updated seqcnt from node A. Now consider that node A reboots, and since it reinitializes the seqcnt value to 240 and sends the information to node B (who has seqcnt of 250 stored on behalf of node A). As per section 7.2. of [RFC6550], when node B receives this packet it will consider the information to be old (since $240 < 250$).

A	B	Output
240	240	A<B, old
240	241	A<B, old
240	::	A<B, old
240	256	A<B, old
240	0	A<B, new
240	1	A>B, new
240	::	A>B, new
240	127	A>B, new

Table 1: Example
lollipop counter
operation

Default values for lollipop counters considered from [RFC6550] Section 7.2.

Based on this figure, there is dead zone (240 to 0) in which if A operates after reboot then the seqcnt will always be considered smaller. Thus node A needs to maintain the seqcnt in persistent storage and reuse this on reboot.

14.3. RPL State variables

The impact of loss of RPL state information differs depending upon the node type (6LN/6LR/6LBR). Following sections explain different state variables and the impact in case this information is lost on reboot.

14.3.1. DODAG Version

The tuple (RPLInstanceID, DODAGID, DODAGVersionNumber) uniquely identifies a DODAG Version. DODAGVersionNumber is incremented everytime a global repair is initiated for the instance (global or local). A node receiving an older DODAGVersionNumber will ignore the DIO message assuming it to be from old DODAG version. Thus a 6LBR node (and 6LR node in case of local DODAG) needs to maintain the DODAGVersionNumber in the persistent storage, so as to be available on reboot. In case the 6LBR could not use the latest DODAGVersionNumber the implication are that it won't be able to recover/re-establish the routing table.

14.3.2. DTSN field in DIO

DTSN (Destination advertisement Trigger Sequence Number) is a DIO message field used as part of procedure to maintain Downward routes. A 6LBR/6LR node may increment a DTSN in case it requires the downstream nodes to send DAO and thus update downward routes on the 6LBR/6LR node. In case of RPL NS-MOP, only the 6LBR maintains the downward routes and thus controls this field update. In case of S-MOP, 6LRs additionally keep downward routes and thus control this field update.

In S-MOP, when a 6LR node switches parent it may have to issue a DIO with incremented DTSN to trigger downstream child nodes to send DAO so that the downward routes are established in all parent/ancestor set. Thus in S-MOP, the frequency of DTSN update might be relatively high (given the node density and hysteresis set by objective function to switch parent).

14.3.3. PathSequence

PathSequence is part of RPL Transit Option, and associated with RPL Target option. A node whichs owns a target address can associate a PathSequence in the DAO message to denote freshness of the target information. This is especially useful when a node uses multiple paths or multiple parents to advertise its reachability.

Loss of PathSequence information maintained on the target node can result in routing adjacencies been lost on 6LRs/6LBR/6BBR.

14.4. State variables update frequency

State variable	Update frequency	Impacts node type
DODAGVersionNumber	Low	6LBR, 6LR(local DODAG)
DTSN	High (SM), Low (NSM)	6LBR, 6LR
PathSequence	High (SM), Low (NSM)	6LR, 6LN

Table 2: RPL State variables

Low=<5 per day, High=>5 per day; SM=Storing MOP, NSM=Non-Storing MOP

14.5. Deliberations

- (1) Is it possible that RPL removes the use of persistent storage for maintaining state information?
- (2) In most cases, the node reboots will happen very rarely. Thus doing a persistent storage book-keeping for handling node reboot might not make sense. Is it possible to consider signaling (especially after the node reboots) so as to avoid maintaining this persistent state? Is it possible to use one-time on-reboot signalling to recover some state information?
- (3) It is necessary that RPL avoids using persistent storage as far as possible. Ideally, extensions to RPL should consider this as a design requirement especially for 6LR and 6LN nodes. DTSN and PathSequence are the primary state variables which have major impact.

14.6. Implementation Notes

An implementation should use a random DAOSequence number on reboot so as to avoid a risk of reusing the same DAOSequence on reboot. Regardless the sequence counter size of 8bits does not provide much guarantees towards choosing a good random number. A parent node will not respond with a DAO-ACK in case it sees a DAO with the same previous DAOSequence.

Write-Before-Use: The state information should be written to the flash before using it in the messaging. If it is done the other way, then the chances are that the node power downs before writing to the persistent storage.

15. Capabilities and its role in RPL

RPL is a distributed protocol and it requires that the participating nodes agree on basic set of primitives to follow. RPL currently handles this using MOP (Mode of Operation) bits in the DIO. MOP bits inform the nodes the basic mode of operation a node MUST support to join the Instance as a 6LR. The MOP is decided and advertised by the root of the RPL Instance. A node not supporting the given MOP may still join the Instance as a leaf node or 6LN.

RPL further uses DIO Configuration Option to advertise the configuration each node needs to use (for e.g., for trickle timer).

15.1. Handshaking node capabilities

Currently there exist no mechanism to handshake capabilities of the root or 6LRs or 6LNs. If a feature is optional and is supported by 6LRs/6LNs then currently there exists no mechanism to signal it. There are several RPL extension proposals which are possibly optional features. Root needs to know if the 6LR/6LN supports these optional features to enable the extension in that path context. Similarly 6LRs and 6LNs need to know whether the root supports certain extensions that it can make use of.

15.2. How do Capabilities differ from MOP and Configuration Option?

Unlike MOP and Configuration Option which are issued by the root of the Instance, Capabilities can be issued by any node. A 6LN/6LR node can advertise its capabilities such that those can be seen by intermediate 6LRs and the root of the Instance.

15.3. Deliberations

- (1) Is it possible for leaf nodes to advertise their set of capabilities, which can be used by root and/or intermediate 6LRs to make run time decisions?
- (2) How should these capabilities be carried? Should it be carried in DAO/DIO/DAO-ACK?
- (3) Should the definition of capabilities be same in both directions (upstream/downstream)?

16. Backward Compatibility issues with RPL Options

Most of the new work in ROLL requires addition of new control options. Everytime a new control option is added, it is required that all the nodes upgrade to support this option. In many cases, the new specification declares using a Flag day to switch to the new functionality.

New control options may not require mandatory handling on every node but it requires at-least some processing. For e.g., assume that a new control option is added to DIO message. The option does not require any handling on the nodes not supporting it but it requires at-least for these nodes to forward this new control option downstream. Currently the new control option may be stripped off.

It should be possible for the unknown control options to be copied as-is to the downstream/upstream node(s). The specification defining the new control option will decide whether a node should strip-off or copy the unknown control option.

17. RPL under-specification

- (a) PathSequence: Is it mandatory to use PathSequence in DAO Transit Information Option? RPL mentions that a 6LR/6LBR hosting the routing entry on behalf of target node should refresh the lifetime on reception of a new Path Sequence. But RPL does not necessarily mandate use of Path Sequence. Most of the open source implementation [RIOT] [CONTIKI] currently do not issue Path Sequence in the DAO message.
- (b) Target Option aggregation in DAO: RPL allows multiple targets to be aggregated in a single DAO message and has introduced a notion of DelayDAO using which a 6LR node could delay its DAO to enable such aggregation. But RPL does not have clear text on handling of aggregated DAOs and thus it hinders interoperability.
- (c) DTSN Update: RPL does not clearly define in which cases DTSN should be updated in case of storing mode of operation. More details for this are presented in Section 3.

18. Acknowledgements

Many thanks to Pascal Thubert for hallway chats and for helping understand the existing design rationales. Thanks to Michael Richardson for Unstrung RPL implementation rationale. Thanks to ML discussions, in particular (<https://www.ietf.org/mail-archive/web/roll/current/msg09443.html>).

19. IANA Considerations

This memo includes no request to IANA.

20. Security Considerations

This is an information draft and does not add any changes to the existing specifications.

21. References

21.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <<https://www.rfc-editor.org/info/rfc6206>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.

21.2. Informative References

- [I-D.clausen-lln-rpl-experiences]
Clausen, T., Verdiere, A. C. D., Yi, J., Herberg, U., and Y. Igarashi, "Observations on RPL: IPv6 Routing Protocol for Low power and Lossy Networks", Work in Progress, Internet-Draft, draft-clausen-lln-rpl-experiences-11, 27 March 2018, <<https://www.ietf.org/archive/id/draft-clausen-lln-rpl-experiences-11.txt>>.
- [I-D.ietf-intarea-adhoc-wireless-com]
Baccelli, E. and C. E. Perkins, "Multi-hop Ad Hoc Wireless Communication", Work in Progress, Internet-Draft, draft-

ietf-intarea-adhoc-wireless-com-02, 20 July 2016,
<<https://www.ietf.org/archive/id/draft-ietf-intarea-adhoc-wireless-com-02.txt>>.

[I-D.ietf-roll-aodv-rpl]

Anamalamudi, S., Perkins, C. E., Anand, S., and B. Liu,
"Supporting Asymmetric Links in Low Power Networks: AODV-
RPL", Work in Progress, Internet-Draft, draft-ietf-roll-
aodv-rpl-11, 16 September 2021,
<<https://www.ietf.org/archive/id/draft-ietf-roll-aodv-rpl-11.txt>>.

[Perlman83]

Perlman, R., "Fault-Tolerant Broadcast of Routing
Information", North-Holland Computer Networks, Vol.7,
December 1983.

Appendix A. Additional Stuff

Authors' Addresses

Rahul Arvind Jadhav (editor)
Marathahalli
Bangalore 560037
Karnataka
India

Email: rahul.ietf@gmail.com

Rabi Narayan Sahoo
Juniper
Whitefield
Bangalore 560037
Karnataka
India

Email: rabinarayans0828@gmail.com

Yuefeng Wu
Huawei
No.101, Software Avenue, Yuhuatai District,
Nanjing
Jiangsu, 210012
China

Phone: +86-15251896569

Email: wuyuefeng@huawei.com

ROLL
Internet-Draft
Updates: 6550, 6775, 8505 (if approved)
Intended status: Standards Track
Expires: 26 July 2021

P. Thubert, Ed.
Cisco Systems
M. Richardson
Sandelman
22 January 2021

Routing for RPL Leaves
draft-ietf-roll-unaware-leaves-30

Abstract

This specification updates RFC6550, RFC6775, and RFC8505. It provides a mechanism for a host that implements a routing-agnostic interface based on 6LoWPAN Neighbor Discovery to obtain reachability services across a network that leverages RFC6550 for its routing operations.

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 July 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	6
2.1. Requirements Language	6
2.2. Glossary	6
2.3. References	7
3. RPL External Routes and Dataplane Artifacts	8
4. 6LoWPAN Neighbor Discovery	9
4.1. RFC 6775 Address Registration	9
4.2. RFC 8505 Extended Address Registration	10
4.2.1. R Flag	10
4.2.2. TID, "I" Field and Opaque Fields	11
4.2.3. Route Ownership Verifier	11
4.3. RFC 8505 Extended DAR/DAC	11
4.3.1. RFC 7400 Capability Indication Option	12
5. Requirements on the RPL-Unware leaf	13
5.1. Support of 6LoWPAN ND	13
5.2. Support of IPv6 Encapsulation	14
5.3. Support of the Hop-by-Hop Header	14
5.4. Support of the Routing Header	14
6. Enhancements to RFC 6550	14
6.1. Updated RPL Target Option	15
6.2. Additional Flag in the RPL DODAG Configuration Option . .	17
6.3. Updated RPL Status	18
7. Enhancements to draft-ietf-roll-efficient-npdao	20
8. Enhancements to RFC6775 and RFC8505	20
9. Protocol Operations for Unicast Addresses	20
9.1. General Flow	21
9.2. Detailed Operation	24
9.2.1. Perspective of the 6LN Acting as RUL	24
9.2.2. Perspective of the 6LR Acting as Border router . . .	25
9.2.3. Perspective of the RPL Root	30
9.2.4. Perspective of the 6LBR	31
10. Protocol Operations for Multicast Addresses	31
11. Security Considerations	34
12. IANA Considerations	35
12.1. Fixing the Address Registration Option Flags	35
12.2. Resizing the ARO Status values	36
12.3. New RPL DODAG Configuration Option Flag	36
12.4. RPL Target Option Registry	36
12.5. New Subregistry for RPL Non-Rejection Status values . .	37
12.6. New Subregistry for RPL Rejection Status values	37
13. Acknowledgments	38
14. Normative References	38
15. Informative References	39
Appendix A. Example Compression	41
Authors' Addresses	42

1. Introduction

The design of Low Power and Lossy Networks (LLNs) is generally focused on saving energy, which is the most constrained resource of all. Other design constraints, such as a limited memory capacity, duty cycling of the LLN devices and low-power lossy transmissions, derive from that primary concern.

The IETF produced the "Routing Protocol for Low Power and Lossy Networks" [RFC6550] (RPL) to provide IPv6 [RFC8200] routing services within such constraints. RPL belongs to the class of Distance-Vector protocols, which, compared to link-state protocols, limit the amount of topological knowledge that needs to be installed and maintained in each node, and does not require convergence to avoid micro-loops.

To save signaling and routing state in constrained networks, RPL allows a path stretch (see [RFC6687]), whereby routing is only performed along a Destination-Oriented Directed Acyclic Graph (DODAG) that is optimized to reach a Root node, as opposed to along the shortest path between 2 peers, whatever that would mean in a given LLN. This trades the quality of peer-to-peer (P2P) paths for a vastly reduced amount of control traffic and routing state that would be required to operate an any-to-any shortest path protocol. Additionally, broken routes may be fixed lazily and on-demand, based on dataplane inconsistency discovery, which avoids wasting energy in the proactive repair of unused paths.

For many of the nodes, though not all, the DODAG provides multiple forwarding solutions towards the Root of the topology via so-called parents. RPL is designed to adapt to fuzzy connectivity, whereby the physical topology cannot be expected to reach a stable state, with a lazy control that creates the routes proactively, but may only fix them reactively, upon actual traffic. The result is that RPL provides reachability for most of the LLN nodes, most of the time, but may not converge in the classical sense.

RPL can be deployed in conjunction with IPv6 Neighbor Discovery (ND) [RFC4861] [RFC4862] and 6LoWPAN ND [RFC6775] [RFC8505] to maintain reachability within a Non-Broadcast Multiple-Access (NBMA) Multi-Link subnet.

In that mode, IPv6 addresses are advertised individually as host routes. Some nodes may act as routers and participate in the forwarding operations whereas others will only receive/originate packets, acting as hosts in the data-plane. In [RFC6550] terms, an IPv6 host [RFC8504] that is reachable over the RPL network is called a leaf.

Section 2 of [USEofRPLinfo] defines the terms RPL leaf, RPL-Aware-leaf (RAL) and RPL-Unaware Leaf (RUL). A RPL leaf is a host attached to one or more RPL router(s); as such, it relies on the RPL router(s) to forward its traffic across the RPL domain but does not forward traffic from another node. As opposed to the RAL, the RUL does not participate to RPL, and relies on its RPL router(s) also to inject the routes to its IPv6 addresses in the RPL domain.

A RUL may be unable to participate because it is very energy-constrained, code-space constrained, or because it would be unsafe to let it inject routes in RPL. Using 6LoWPAN ND as opposed to RPL as the host-to-router interface limits the surface of the possible attacks by the RUL against the RPL domain. If all RULs and RANs use 6LoWPAN ND for Neighbor Discovery, it is also possible to protect the address ownership of all nodes, including the RULs.

This document specifies how the router injects the host routes in the RPL domain on behalf of the RUL. Section 5 details how the RUL can leverage 6LoWPAN ND to obtain the routing services from the router. In that model, the RUL is also a 6LoWPAN Node (6LN) and the RPL-Aware router is also a 6LoWPAN Router (6LR). Using the 6LoWPAN ND Address Registration mechanism, the RUL signals that the router must inject a host route for the Registered Address.

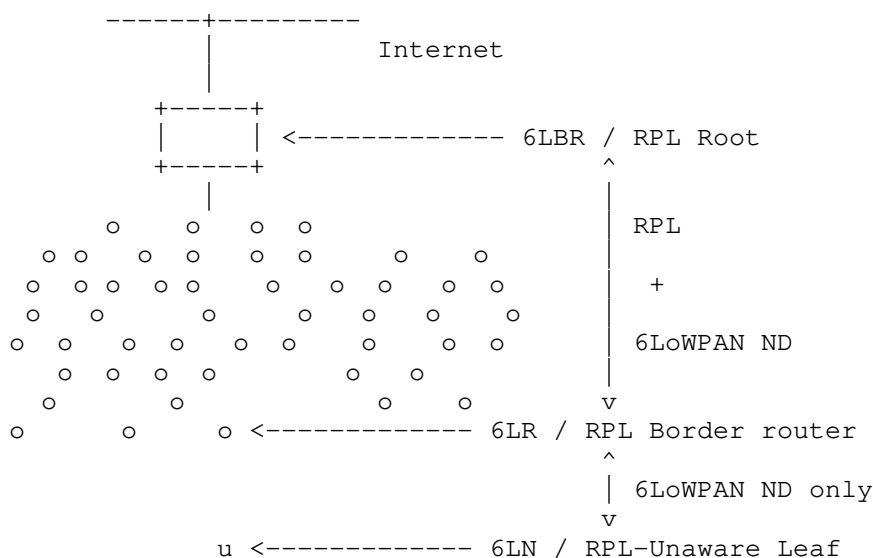


Figure 1: Injecting Routes on behalf of RULs

The RPL Non-Storing Mode mechanism is used to extend the routing state with connectivity to the RULs even when the DODAG is operated in Storing Mode. The unicast packet forwarding operation by the 6LR serving a RUL is described in section 4.1 of [USEofRPLinfo].

Examples of possible RULs include severely energy constrained sensors such as window smash sensor (alarm system), and kinetically powered light switches. Other applications of this specification may include a smart grid network that controls appliances - such as washing machines or the heating system - in the home. Appliances may not participate to the RPL protocol operated in the Smartgrid network but can still interact with the Smartgrid for control and/or metering.

This specification can be deployed incrementally in a network that implements [USEofRPLinfo]. Only the Root and the 6LRs that connect the RULs need to be upgraded. The RPL routers on path will only see unicast IPv6 traffic between the Root and the 6LR.

This document is organized as follows:

- * Section 3 and Section 4 present in a non-normative fashion the salient aspects of RPL and 6LoWPAN ND, respectively, that are leveraged in this specification to provide connectivity to a 6LN acting as a RUL across a RPL network.
- * Section 5 lists the requirements that a RUL needs to match in order to be served by a RPL router that complies with this specification.
- * Section 6 presents the changes made to [RFC6550]; a new behavior is introduced whereby the 6LR advertises the 6LN's addresses in a RPL DAO message based on the ND registration by the 6LN, and the RPL root performs the EDAR/EDAC exchange with the 6LoWPAN Border Router (6LBR) on behalf of the 6LR; modifications are introduced to some RPL options and to the RPL Status to facilitate the integration of the protocols.
- * Section 7 presents the changes made to [EFFICIENT-NPDAO]; the use of the DCO message is extended to the Non-Storing MOP to report asynchronous issues from the Root to the 6LR.
- * Section 8 presents the changes made to [RFC6775] and [RFC8505]; The range of the ND status codes is reduced down to 64 values, and the remaining bits in the original status field are now reserved.
- * Section 9 and Section 10 present the operation of this specification for unicast and multicast flows, respectively, and Section 11 presents associated security considerations.

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. Glossary

This document uses the following acronyms:

6CIO: 6LoWPAN Capability Indication Option
6LN: 6LoWPAN Node (a Low Power host or router)
6LR: 6LoWPAN router
6LBR: 6LoWPAN Border router
(E)ARO: (Extended) Address Registration Option
(E)DAR: (Extended) Duplicate Address Request
(E)DAC: (Extended) Duplicate Address Confirmation
DAD: Duplicate Address Detection
DAO: Destination Advertisement Object (a RPL message)
DCO: Destination Cleanup Object (a RPL message)
DIO: DODAG Information Object (a RPL message)
DODAG: Destination-Oriented Directed Acyclic Graph
LLN: Low-Power and Lossy Network
MOP: RPL Mode of Operation
NA: Neighbor Advertisement
NCE: Neighbor Cache Entry
ND: Neighbor Discovery
NS: Neighbor Solicitation
RA: router Advertisement
ROVR: Registration Ownership Verifier
RPI: RPL Packet Information
RAL: RPL-aware Leaf
RAN: RPL-Aware Node (either a RPL router or a RPL-aware Leaf)
RUL: RPL-Unaware Leaf
SRH: Source-Routing Header
TID: Transaction ID (a sequence counter in the EARO)
TIO: Transit Information Option

2.3. References

The Terminology used in this document is consistent with and incorporates that described in "Terms Used in Routing for Low-Power and Lossy Networks (LLNs)" [RFC7102]. A glossary of classical 6LoWPAN acronyms is given in Section 2.2. Other terms in use in LLNs are found in "Terminology for Constrained-Node Networks" [RFC7228]. This specification uses the terms 6LN and 6LR to refer specifically to nodes that implement the 6LN and 6LR roles in 6LoWPAN ND and does not expect other functionality such as 6LoWPAN Header Compression [RFC6282] from those nodes.

"RPL", the "RPL Packet Information" (RPI), "RPL Instance" (indexed by a RPLInstanceID), "up", "down" are defined in "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550]. The RPI is the abstract information that RPL defines to be placed in data packets, e.g., as the RPL Option [RFC6553] within the IPv6 Hop-By-Hop Header. By extension, the term "RPI" is often used to refer to the RPL Option itself. The Destination Advertisement Object (DAO) and DODAG Information Object (DIO) messages are also specified in [RFC6550]. The Destination Cleanup Object (DCO) message is defined in [EFFICIENT-NPDAO].

This document uses the terms RPL-Unaware Leaf (RUL), RPL-Aware Node (RAN) and RPL aware Leaf (RAL) consistently with [USEofRPLinfo]. A RAN is either a RAL or a RPL router. As opposed to a RUL, a RAN manages the reachability of its addresses and prefixes by injecting them in RPL by itself.

In this document, readers will encounter terms and concepts that are discussed in the following documents:

Classical IPv6 ND: "Neighbor Discovery for IP version 6" [RFC4861] and "IPv6 Stateless Address Autoconfiguration" [RFC4862],

6LoWPAN: "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing" [RFC6606] and "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], and

6LoWPAN ND: Neighbor Discovery Optimization for Low-Power and Lossy Networks [RFC6775], "Registration Extensions for 6LoWPAN Neighbor Discovery" [RFC8505], "Address Protected Neighbor Discovery for Low-power and Lossy Networks" [RFC8928], and "IPv6 Backbone Router" [RFC8929].

3. RPL External Routes and Dataplane Artifacts

RPL was initially designed to build stub networks whereby the only border router would be the RPL Root (typically collocated with the 6LBR) and all the nodes in the stub would be RPL-Aware. But [RFC6550] was also prepared to be extended for external routes (targets in RPL parlance) with the External 'E' flag in the Transit Information Option (TIO). External targets enable to reach destinations that are outside the RPL domain and connected to the RPL domain via RPL border routers that are not the Root. Section 4.1 of [USEofRPLinfo] provides a set of rules summarized below that must be followed for routing packets to and from an external destination. A RUL is a special case of an external target that is also a host directly connected to the RPL domain.

A 6LR that acts as a border router for external routes advertises them using Non-Storing Mode DAO messages that are unicast directly to the Root, even if the DODAG is operated in Storing Mode. Non-Storing Mode routes are not visible inside the RPL domain and all packets are routed via the Root. The RPL Root tunnels the data packets directly to the 6LR that advertised the external route, which decapsulates and forwards the original (inner) packets.

The RPL Non-Storing MOP signaling and the associated IPv6-in-IPv6 encapsulated packets appear as normal traffic to the intermediate routers. The support of external routes only impacts the Root and the 6LR. It can be operated with legacy intermediate routers and does not add to the amount of state that must be maintained in those routers. A RUL is an example of a destination that is reachable via an external route that happens to be also a host route.

The RPL data packets typically carry a Hop-by-Hop Header with a RPL Option [RFC6553] that contains the Packet Information (RPI) defined in section 11.2 of [RFC6550]. Unless the RUL already placed a RPL Option in outer header chain, the packets from and to the RUL are encapsulated using an IPv6-in-IPv6 tunnel between the Root and the 6LR that serves the RUL (see sections 7 and 8 of [USEofRPLinfo] for details). If the packet from the RUL has an RPI, the 6LR as a RPL border router rewrites the RPI to indicate the selected Instance and set the flags, but it does not need to encapsulate the packet (see Section 9.2.2) .

In Non-Storing Mode, packets going down carry a Source Routing Header (SRH). The IPv6-in-IPv6 encapsulation, the RPI and the SRH are collectively called the "RPL artifacts" and can be compressed using [RFC8138]. Appendix A presents an example compressed format for a packet forwarded by the Root to a RUL in a Storing Mode DODAG.

The inner packet that is forwarded to the RUL may carry some RPL artifacts, e.g., an RPI if the original packet was generated with it, and an SRH in a Non-Storing Mode DODAG. [USEofRPLinfo] expects the RUL to support the basic "IPv6 Node Requirements" [RFC8504] and in particular the mandates in Sections 4.2 and 4.4 of [RFC8200]. As such, the RUL is expected to ignore the RPL artifacts that may be left over, either an SRH with zero Segments Left or a RPL Option in the Hop-by-Hop Header, which can be skipped when not recognized, see Section 5 for more.

A RUL is not expected to support the compression method defined in [RFC8138]. For that reason, the border router (the 6LR here) uncompresses the packet before forwarding it over an external route to a RUL [USEofRPLinfo].

4. 6LoWPAN Neighbor Discovery

This section goes through the 6LoWPAN ND mechanisms that this specification leverages, as a non-normative reference to the reader. The full normative text is to be found in [RFC6775], [RFC8505], and [RFC8928].

4.1. RFC 6775 Address Registration

The classical "IPv6 Neighbor Discovery (IPv6 ND) Protocol" [RFC4861] [RFC4862] was defined for serial links and transit media such as Ethernet. It is a reactive protocol that relies heavily on multicast operations for Address Discovery (aka Lookup) and Duplicate Address Detection (DAD).

"Neighbor Discovery Optimizations for 6LoWPAN networks" [RFC6775] adapts IPv6 ND for operations over energy-constrained LLNs. The main functions of [RFC6775] are to proactively establish the Neighbor Cache Entry (NCE) in the 6LR and to prevent address duplication. To that effect, [RFC6775] introduces a new unicast Address Registration mechanism that contributes to reducing the use of multicast messages compared to the classical IPv6 ND protocol.

[RFC6775] defines a new Address Registration Option (ARO) that is carried in the unicast Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages between the 6LoWPAN Node (6LN) and the 6LoWPAN router (6LR). It also defines the Duplicate Address Request (DAR) and Duplicate Address Confirmation (DAC) messages between the 6LR and the 6LBR). In an LLN, the 6LBR is the central repository of all the Registered Addresses in its domain and the source of truth for uniqueness and ownership.

4.2. RFC 8505 Extended Address Registration

"Registration Extensions for 6LoWPAN Neighbor Discovery" [RFC8505] updates RFC 6775 into a generic Address Registration mechanism that can be used to access services such as routing and ND proxy. To that effect, [RFC8505] defines the Extended Address Registration Option (EARO), shown in Figure 2:

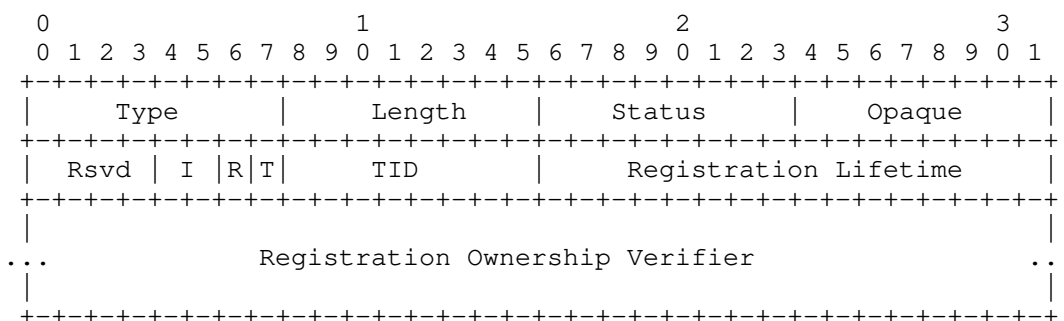


Figure 2: EARO Option Format

4.2.1. R Flag

[RFC8505] introduces the R Flag in the EARO. The Registering Node sets the R Flag to indicate whether the 6LR should ensure reachability for the Registered Address. If the R Flag is set to 0, then the Registering Node handles the reachability of the Registered Address by other means. In a RPL network, this means that either it is a RAN that injects the route by itself or that it uses another RPL router for reachability services.

This document specifies how the R Flag is used in the context of RPL. A RPL leaf that implements the 6LN functionality from [RFC8505] requires reachability services for an IPv6 address if and only if it sets the R Flag in the NS(EARO) used to register the address to a 6LR acting as a RPL border router. Upon receiving the NS(EARO), the RPL router generates a DAO message for the Registered Address if and only if the R flag is set to 1.

Section 9.2 specifies additional operations when R flag is set to 1 in an EARO that is placed either in an NS or an NA message.

4.2.2. TID, "I" Field and Opaque Fields

When the T Flag is set to 1, the EARO includes a sequence counter called Transaction ID (TID), that is needed to fill the Path Sequence Field in the RPL Transit Option. This is the reason why the support of [RFC8505] by the RUL, as opposed to only [RFC6775], is a prerequisite for this specification); this requirement is fully explained in Section 5.1. The EARO also transports an Opaque field and an associated "I" field that describes what the Opaque field transports and how to use it.

Section 9.2.1 specifies the use of the "I" field and the Opaque field by a RUL.

4.2.3. Route Ownership Verifier

Section 5.3 of [RFC8505] introduces the Registration Ownership Verifier (ROVR) field of variable length from 64 to 256 bits. The ROVR is a replacement of the EUI-64 in the ARO [RFC6775] that was used to identify uniquely an Address Registration with the Link-Layer address of the owner but provided no protection against spoofing.

"Address Protected Neighbor Discovery for Low-power and Lossy Networks" [RFC8928] leverages the ROVR field as a cryptographic proof of ownership to prevent a rogue third party from registering an address that is already owned. The use of ROVR field enables the 6LR to block traffic that is not sourced at an owned address.

This specification does not address how the protection by [RFC8928] could be extended for use in RPL. On the other hand, it adds the ROVR to the DAO to build the proxied EDAR at the Root (see Section 6.1), which means that nodes that are aware of the host route are also aware of the ROVR associated to the Target Address.

4.3. RFC 8505 Extended DAR/DAC

[RFC8505] updates the DAR/DAC messages into the Extended DAR/DAC to carry the ROVR field. The EDAR/EDAC exchange takes place between the 6LR and the 6LBR. It is triggered by an NS(EARO) message from a 6LN to create, refresh, and delete the corresponding state in the 6LBR. The exchange is protected by the retry mechanism specified in Section 8.2.6 of [RFC6775], though in an LLN, a duration longer than the default value of the RetransTimer (RETRANS_TIMER) [RFC4861] of 1 second may be necessary to cover the round trip delay between the 6LR and the 6LBR.

RPL [RFC6550] specifies a periodic DAO from the 6LN all the way to the Root that maintains the routing state in the RPL network for the lifetime indicated by the source of the DAO. This means that for each address, there are two keep-alive messages that traverse the whole network, one to the Root and one to the 6LBR.

This specification avoids the periodic EDAR/EDAC exchange across the LLN. The 6LR turns the periodic NS(EARO) from the RUL into a DAO message to the Root on every refresh, but it only generates the EDAR upon the first registration, for the purpose of DAD, which must be verified before the address is injected in RPL. Upon the DAO message, the Root proxies the EDAR exchange to refresh the state at the 6LBR on behalf of the 6LR, as illustrated in Figure 8 in Section 9.1.

4.3.1. RFC 7400 Capability Indication Option

"6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)" [RFC7400] defines the 6LoWPAN Capability Indication Option (6CIO) that enables a node to expose its capabilities in router Advertisement (RA) messages.

[RFC8505] defines a number of bits in the 6CIO, in particular:

L: Node is a 6LR.

E: Node is an IPv6 ND Registrar -- i.e., it supports registrations based on EARO.

P: Node is a Routing Registrar, -- i.e., an IPv6 ND Registrar that also provides reachability services for the Registered Address.

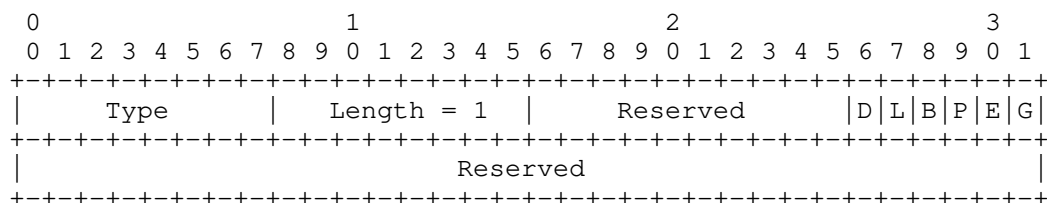


Figure 3: 6CIO flags

A 6LR that provides reachability services for a RUL in a RPL network as specified in this document includes a 6CIO in its RA messages and set the L, P and E flags to 1 as prescribed by [RFC8505]; this is fully explained in Section 9.2.

5. Requirements on the RPL-Unware leaf

This document describes how RPL routing can be extended to reach a RUL. This section specifies the minimal RPL-independent functionality that the RUL needs to implement to obtain routing services for its addresses.

5.1. Support of 6LoWPAN ND

To obtain routing services from a router that implements this specification, a RUL needs to implement [RFC8505] and sets the "R" and "T" flags in the EARO to 1 as discussed in Section 4.2.1 and Section 4.2.2, respectively. Section 9.2.1 specifies new behaviors for the RUL, e.g., when the R Flag set to 1 in a NS(EARO) is not echoed in the NA(EARO), which indicates that the route injection failed.

The RUL is expected to request routing services from a router only if that router originates RA messages with a 6CIO that has the L, P, and E flags all set to 1 as discussed in Section 4.3.1, unless configured to do so. It is suggested that the RUL also implements [RFC8928] to protect the ownership of its addresses.

A RUL that may attach to multiple 6LRs is expected to prefer those that provide routing services. The RUL needs to register to all the 6LRs from which it desires routing services.

Parallel Address Registrations to several 6LRs should be performed in a rapid sequence, using the same EARO for the same Address. Gaps between the Address Registrations will invalidate some of the routes till the Address Registration finally shows on those routes.

[RFC8505] introduces error Status values in the NA(EARO) which can be received synchronously upon an NS(EARO) or asynchronously. The RUL needs to support both cases and refrain from using the address when the Status value indicates a rejection (see Section 6.3).

5.2. Support of IPv6 Encapsulation

Section 2.1 of [USEofRPLInfo] defines the rules for tunneling either to the final destination (e.g., a RUL) or to its attachment router (designated as 6LR). In order to terminate the IPv6-in-IPv6 tunnel, the RUL, as an IPv6 host, would have to be capable of decapsulating the tunneled packet and either drop the encapsulated packet if it is not the final destination, or pass it to the upper layer for further processing. As indicated in section 4.1 of [USEofRPLInfo], this is not mandated by [RFC8504], and the IPv6-in-IPv6 tunnel from the Root is terminated at the parent 6LR. It is thus not necessary for a RUL to support IPv6-in-IPv6 decapsulation.

5.3. Support of the Hop-by-Hop Header

A RUL is expected to process an Option Type in a Hop-by-Hop Header as prescribed by section 4.2 of [RFC8200]. An RPI with an Option Type of 0x23 [USEofRPLInfo] is thus skipped when not recognized.

5.4. Support of the Routing Header

A RUL is expected to process an unknown Routing Header Type as prescribed by section 4.4 of [RFC8200]. This implies that the Source Routing Header, which has a Routing Type of 3 [RFC6554], is ignored when the Segments Left is zero. When the Segments Left is non-zero, the RUL discards the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

6. Enhancements to RFC 6550

This document specifies a new behavior whereby a 6LR injects DAO messages for unicast addresses (see Section 9) and multicast addresses (see Section 10) on behalf of leaves that are not aware of RPL. The RUL addresses are exposed as external targets [RFC6550]. Conforming to [USEofRPLInfo], an IPv6-in-IPv6 encapsulation between the 6LR and the RPL Root is used to carry the RPL artifacts and remove them when forwarding outside the RPL domain, e.g., to a RUL.

This document also synchronizes the liveness monitoring at the Root and the 6LBR. The same value of lifetime is used for both, and a single keep-alive message, the RPL DAO, traverses the RPL network. A new behavior is introduced whereby the RPL Root proxies the EDAR message to the 6LBR on behalf of the 6LR (see Section 8), for any leaf node that implements the 6LN functionality in [RFC8505].

Section 6.7.7 of [RFC6550] introduces the RPL Target Option, which can be used in RPL Control messages such as the DAO message to signal a destination prefix. This document adds the capabilities to transport the ROVR field (see Section 4.2.3) and the IPv6 Address of the prefix advertiser when the Target is a shorter prefix. Their use is signaled respectively by a new ROVR Size field being non-zero and a new "Advertiser address in Full" 'F' flag set to 1, see Section 6.1.

This specification defines a new flag, "Root Proxies EDAR/EDAC" (P), in the RPL DODAG Configuration option, see Section 6.2.

The RPL Status defined in section 6.5.1 of [RFC6550] for use in the DAO-ACK message is extended to be placed in DCO messages [EFFICIENT-NPDAO] as well. Furthermore, this specification enables to carry the EARO Status defined for 6LoWPAN ND in RPL DAO and DCO messages, embedded in a RPL Status, see Section 6.3.

Section 12 of [RFC6550] details the RPL support for multicast flows when the RPLInstance is operated in the MOP of 3 ("Storing Mode of Operation with multicast support"). This specification extends the RPL Root operation to proxy-relay the MLDv2 [RFC3810] operation between the RUL and the 6LR, see Section 10.

6.1. Updated RPL Target Option

This specification updates the RPL Target Option to transport the ROVR that was also defined for 6LoWPAN ND messages. This enables the RPL Root to generate the proxied EDAR message to the 6LBR.

The Target Prefix of the RPL Target Option is left (high bit) justified and contains the advertised prefix; its size may be smaller than 128 when it indicates a Prefix route. The Prefix Length field signals the number of bits that correspond to the advertised Prefix; it is 128 for a host route or less in the case of a Prefix route. This remains unchanged.

This specification defines the new 'F' flag. When it is set to 1, the size of the Target Prefix field MUST be 128 bits and it MUST contain an IPv6 address of the advertising node taken from the advertised Prefix. In that case, the Target Prefix field carries two distinct pieces of information: a route that can be a host route or a Prefix route depending on the Prefix Length, and an IPv6 address that can be used to reach the advertising node and validate the route.

If the 'F' flag is set to 0, the Target Prefix field can be shorter than 128 bits and it MUST be aligned to the next byte boundary after the end of the prefix. Any additional bits in the rightmost octet are filled with padding bits. Padding bits are reserved and set to 0 as specified in section 6.7.7 of [RFC6550].

With this specification the ROVR is the remainder of the RPL Target Option. The size of the ROVR is indicated in a new ROVR Size field that is encoded to map one-to-one with the Code Suffix in the EDAR message (see table 4 of [RFC8505]). The ROVR Size field is taken from the flags field, which is an update to the RPL Target Option Flags IANA registry.

The updated format is illustrated in Figure 4. It is backward compatible with the Target Option in [RFC6550]. It is recommended that the updated format be used as a replacement in new implementations in all MOPs in preparation for upcoming Route Ownership Validation mechanisms based on the ROVR, unless the device or the network is so constrained that this is not feasible.

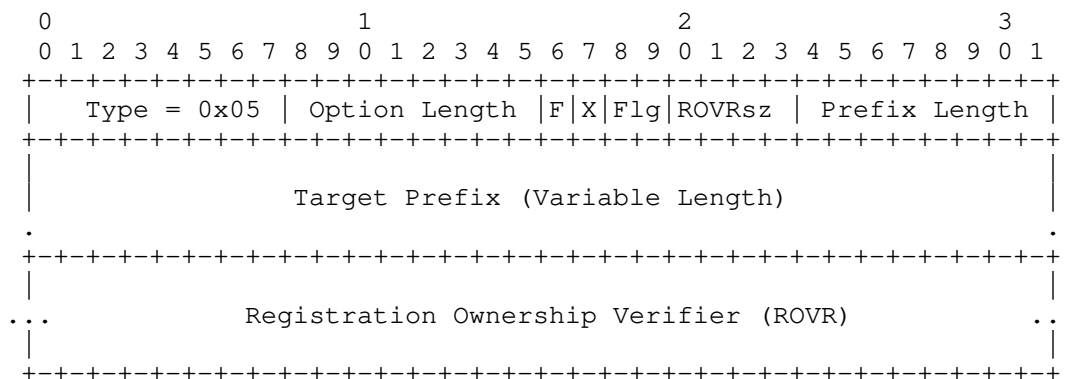


Figure 4: Updated Target Option

New fields:

F: 1-bit flag. Set to 1 to indicate that Target Prefix field contains the complete (128 bit) IPv6 address of the advertising node.

X: 1-bit flag. Set to 1 to request that the Root performs a proxy EDAR/EDAC exchange.

The 'X' flag can only be set to 1 if the DODAG is operating in Non-Storing Mode and if the Root sets the "Root Proxies EDAR/EDAC (P)" flag to 1 in the DODAG Configuration Option, see Section 6.2.

The 'X' flag can be set for host routes to RULs and RANs; it can also be set for internal prefix routes if the 'F' flag is set, using the node's address in the Target Prefix field to form the EDAR, but it cannot be used otherwise.

Flg (Flags): The 2 bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

ROVRsz (ROVR Size): Indicates the Size of the ROVR. It MUST be set to 1, 2, 3, or 4, indicating a ROVR size of 64, 128, 192, or 256 bits, respectively.

If a legacy Target Option is used, then the value must remain 0, as specified in [RFC6550].

In case of a value above 4, the size of the ROVR is undetermined and this node cannot validate the ROVR; an implementation SHOULD propagate the whole Target Option upwards as received to enable the verification by an ancestor that would support the upgraded ROVR.

Registration Ownership Verifier (ROVR): This is the same field as in the EARO, see [RFC8505]

6.2. Additional Flag in the RPL DODAG Configuration Option

The DODAG Configuration Option is defined in Section 6.7.6 of [RFC6550]. Its purpose is extended to distribute configuration information affecting the construction and maintenance of the DODAG, as well as operational parameters for RPL on the DODAG, through the DODAG. This Option was originally designed with 4 bit positions reserved for future use as Flags.

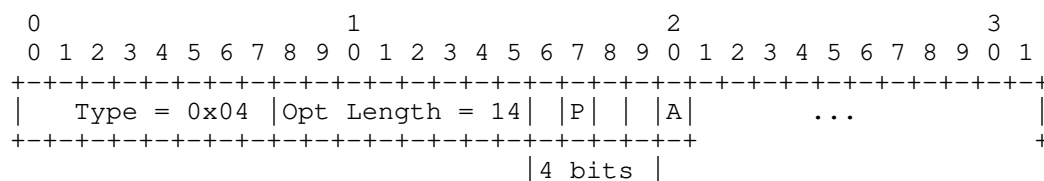


Figure 5: DODAG Configuration Option (Partial View)

This specification defines a new flag "Root Proxies EDAR/EDAC" (P). The 'P' flag is encoded in bit position 1 of the reserved Flags in the DODAG Configuration Option (counting from bit 0 as the most significant bit) and it is set to 0 in legacy implementations as specified respectively in Sections 20.14 and 6.7.6 of [RFC6550].

The 'P' flag is set to 1 to indicate that the Root performs the proxy operation, which implies that it supports this specification and the updated RPL Target Option (see Section 6.1).

Section 4.3 of [USEofRPLinfo] updates [RFC6550] to indicate that the definition of the Flags applies to Mode of Operation (MOP) values from zero (0) to six (6) only. For a MOP value of 7, the implementation MUST consider that the Root performs the proxy operation.

The RPL DODAG Configuration Option is typically placed in a DODAG Information Object (DIO) message. The DIO message propagates down the DODAG to form and then maintain its structure. The DODAG Configuration Option is copied unmodified from parents to children. [RFC6550] states that "Nodes other than the DODAG Root MUST NOT modify this information when propagating the DODAG Configuration option". Therefore, a legacy parent propagates the 'P' Flag as set by the Root, and when the 'P' Flag is set to 1, it is transparently flooded to all the nodes in the DODAG.

6.3. Updated RPL Status

The RPL Status is defined in section 6.5.1 of [RFC6550] for use in the DAO-ACK message and values are assigned as follows:

Range	Meaning
0	Success/Unqualified acceptance
1-127	Not an outright rejection
128-255	Rejection

Table 1: RPL Status per RFC 6550

The 6LoWPAN ND Status was defined for use in the EARO, see section 4.1 of [RFC8505]. This specification adds a capability to allow the carriage of 6LoWPAN ND Status values in RPL DAO and DCO messages, embedded in the RPL Status field.

To achieve this, the range of the ARO/EARO Status values is reduced to 0-63, which updates the IANA registry created for [RFC6775]. This reduction ensures that the values fit within a RPL Status as shown in Figure 6. See Section 12.2, Section 12.5, and Section 12.6 for the respective IANA declarations. This ask is reasonable because the associated registry relies on standards action for registration and only values up to 10 are currently allocated.

```

      0 1 2 3 4 5 6 7
    +-+---+---+---+---+
    |E|A|StatusValue|
    +-+---+---+---+---+

```

Figure 6: RPL Status Format

This specification updates the RPL Status with subfields as indicated below:

E: 1-bit flag. set to 1 to indicate a rejection. When set to 0, a Status value of 0 indicates Success/Unqualified acceptance and other values indicate "not an outright rejection" as per RFC 6550.

A: 1-bit flag. Indicates the type of the RPL Status value.

Status Value: 6-bit unsigned integer.

If the 'A' flag is set to 1 this field transports a value defined for the 6LoWPAN ND EARO Status.

When the 'A' flag is set to 0, this field transports a Status Value defined for RPL.

When building a DCO or a DAO-ACK message upon an IPv6 ND NA or a EDAC message, the RPL Root MUST copy the 6LoWPAN ND status code unchanged in the RPL Status Value and set the 'A' flag to 1. The RPL Root MUST set the 'E' flag to 1 for all rejection and unknown status codes. The status codes in the 1-10 range [RFC8505] are all considered rejections.

Reciprocally, upon a DCO or a DAO-ACK message from the RPL Root with a RPL Status that has the 'A' flag set, the 6LR MUST copy the RPL Status value unchanged in the Status field of the EARO when generating an NA to the RUL.

7. Enhancements to draft-ietf-roll-efficient-npdao

[EFFICIENT-NPDAO] defines the DCO message for RPL Storing Mode only, with a link-local scope. All nodes in the RPL network are expected to support the specification since the message is processed hop-by-hop along the path that is being cleaned up.

This specification extends the use of the DCO message to the Non-Storing MOP, whereby the DCO is sent end-to-end by the Root directly to the RAN that injected the DAO message for the considered target. In that case, intermediate nodes do not need to support [EFFICIENT-NPDAO]; they forward the DCO message as a plain IPv6 packet between the Root and the RAN.

In the case of a RUL, the 6LR that serves the RUL acts as the RAN that receives the Non-Storing DCO. This specification leverages the Non-Storing DCO between the Root and the 6LR that serves as attachment router for a RUL. A 6LR and a Root that support this specification MUST implement the Non-Storing DCO.

8. Enhancements to RFC6775 and RFC8505

This document updates [RFC6775] and [RFC8505] to reduce the range of the ND status codes down to 64 values. The two most significant (leftmost) bits if the original ND status field are now reserved, they MUST be set to zero by the sender and ignored by the receiver.

This document also updates the behavior of a 6LR acting as RPL router and of a 6LN acting as RUL in the 6LoWPAN ND Address Registration as follows:

- * If the RPL Root advertises the capability to proxy the EDAR/EDAC exchange to the 6LBR, the 6LR refrains from sending the keep-alive EDAR message. If it is separated from the 6LBR, the Root regenerates the EDAR message to the 6LBR periodically, upon a DAO message that signals the liveness of the address.
- * The use of the R Flag is extended to the NA(EO) to confirm whether the route was installed.

9. Protocol Operations for Unicast Addresses

The description below assumes that the Root sets the 'P' flag in the DODAG Configuration Option and performs the EDAR proxy operation presented in Section 4.3 .

If the 'P' flag is set to 0, the 6LR MUST generate the periodic EDAR messages and process the returned status as specified in [RFC8505]. If the EDAC indicates success, the rest of the flow takes place as presented but without the proxied EDAR/EDAC exchange.

Section 9.1 provides an overview of the route injection in RPL, whereas Section 9.2 offers more details from the perspective of the different nodes involved in the flow.

9.1. General Flow

This specification eliminates the need to exchange keep-alive Extended Duplicate Address messages, EDAR and EDAC, all the way from a 6LN to the 6LBR across a RPL mesh. Instead, the EDAR/EDAC exchange with the 6LBR is proxied by the RPL Root upon the DAO message that refreshes the RPL routing state. The first EDAR upon a new Registration cannot be proxied, though, as it serves for the purpose of DAD, which must be verified before the address is injected in RPL.

In a RPL network where the function is enabled, refreshing the state in the 6LBR is the responsibility of the Root. Consequently, only addresses that are injected in RPL will be kept alive at the 6LBR by the RPL Root. Since RULs are advertised using Non-Storing Mode, the DAO message flow and the keep alive EDAR/EDAC can be nested within the Address (re)Registration flow. Figure 7 illustrates that, for the first Registration, both the DAD and the keep-alive EDAR/EDAC exchanges happen in the same sequence.

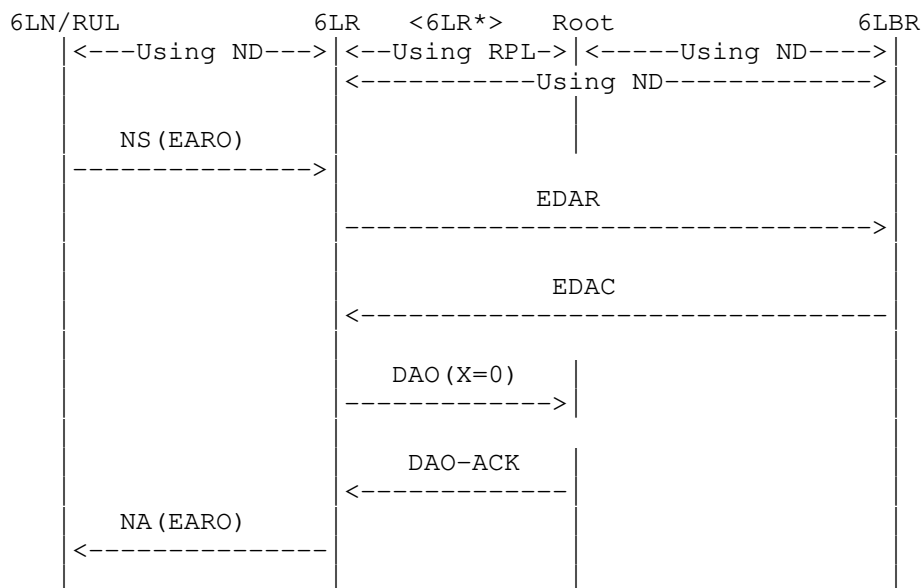


Figure 7: First RUL Registration Flow

This flow requires that the lifetimes and sequence counters in 6LoWPAN ND and RPL are aligned.

To achieve this, the Path Sequence and the Path Lifetime in the DAO message are taken from the Transaction ID and the Address Registration lifetime in the NS(EARO) message from the 6LN.

On the first Address Registration, illustrated in Figure 7 for RPL Non-Storing Mode, the Extended Duplicate Address exchange takes place as prescribed by [RFC8505]. If the exchange fails, the 6LR returns an NA message with a non-zero status to the 6LN, the NCE is not created, and the address is not injected in RPL. Otherwise, the 6LR creates an NCE and injects the Registered Address in the RPL routing using a DAO/DAO-ACK exchange with the RPL DODAG Root.

An Address Registration refresh is performed by the 6LN to keep the NCE in the 6LR alive before the lifetime expires. Upon the refresh of a registration, the 6LR reinjects the corresponding route in RPL before it expires, as illustrated in Figure 8.

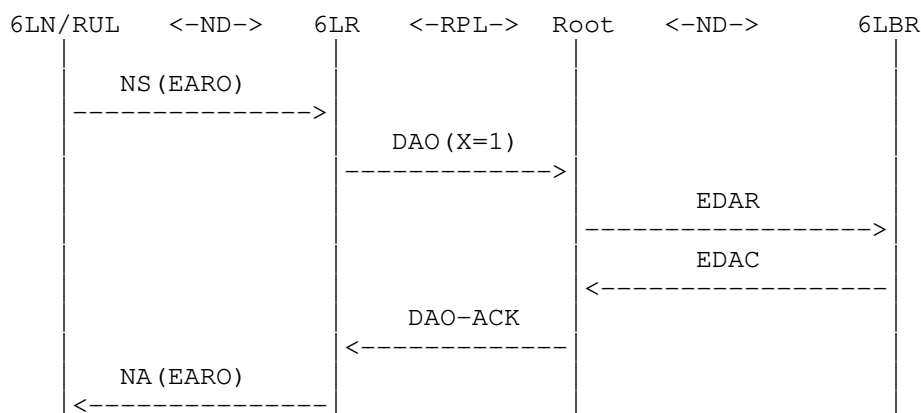


Figure 8: Next RUL Registration Flow

This is what causes the RPL Root to refresh the state in the 6LBR, using an EDAC message. In case of an error in the proxied EDAR flow, the error is returned in the DAO-ACK using a RPL Status with the 'A' flag set to 1 that imbeds a 6LoWPAN Status value as discussed in Section 6.3.

The 6LR may receive a requested DAO-ACK after it received an asynchronous Non-Storing DCO, but the non-zero status in the DCO supersedes a positive Status in the DAO-ACK regardless of the order in which they are received. Upon the DAO-ACK - or the DCO if one arrives first - the 6LR responds to the RUL with an NA(EARO).

An issue may be detected later, e.g., the address moves to a different DODAG with the 6LBR attached to a different 6LoWPAN Backbone router (6BBR), see Figure 5 in section 3.3 of [RFC8929]. The 6BBR may send a negative ND status, e.g., in an asynchronous NA(EARO) to the 6LBR.

[RFC8929] expects that the 6LBR is collocated with the RPL Root, but if not, the 6LBR MUST forward the status code to the originator of the EDAR, either the 6LR or the RPL Root that proxies for it. The ND status code is mapped in a RPL Status value by the RPL Root, and then back by the 6LR. Note that a legacy RAN that receives a Non-Storing DCO that it does not support will ignore it silently, as specified in section 6 of [RFC6550]. The result is that it may ignore for a while that it is no more reachable. The situation will be cleared upon the next Non-Storing DAO exchange if the error is returned in a DAO-ACK.

Figure 9 illustrates this in the case where the 6LBR and the Root are not collocated, and the Root proxies the EDAR/EDAC flow.

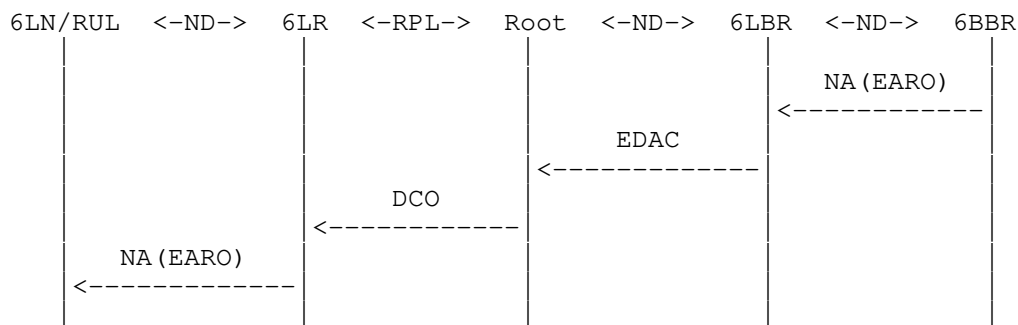


Figure 9: Asynchronous Issue

If the Root does not proxy, then the EDAC with a non-zero status reaches the 6LR directly. In that case, the 6LR MUST clean up the route using a DAO with a Lifetime of zero, and it MUST propagate the status back to the RUL in a NA(EARO) with the R Flag set to 0.

The RUL may terminate the registration at any time by using a Registration Lifetime of 0. This specification requires that the RPL Target Option transports the ROVR. This way, the same flow as the heartbeat flow is sufficient to inform the 6LBR using the Root as proxy, as illustrated in Figure 8.

Any combination of the logical functions of 6LR, Root, and 6LBR might be collapsed in a single node.

9.2. Detailed Operation

The following section specify respectively the behaviour of the 6LN Acting as RUL, the 6LR Acting as Border router and serving the 6LN, the RPL Root and the 6LBR in the control flows that enable RPL routing back to the RUL.

9.2.1. Perspective of the 6LN Acting as RUL

This specification builds on the operation of a 6LoWPAN ND-compliant 6LN/RUL, which is expected to operate as follows:

1. The 6LN selects a 6LR that provides reachability services for a RUL. This is signaled by a 6CIO in the RA messages with the L, P and E flags set to 1 as prescribed by [RFC8505].
2. The 6LN obtains an IPv6 global address, either using Stateless Address Autoconfiguration (SLAAC) [RFC4862] based on a Prefix Information Option (PIO) [RFC4861] found in an RA message, or some other means, such as DHCPv6 [RFC8415].
3. Once it has formed an address, the 6LN registers its address and refreshes its registration periodically, early enough within the Lifetime of the previous Address Registration, as prescribed by [RFC6775], to refresh the NCE before the lifetime indicated in the EARO expires. It sets the T Flag to 1 as prescribed in [RFC8505]. The TID is incremented each time and wraps in a lollipop fashion (see section 5.2.1 of [RFC8505], which is fully compatible with section 7.2 of [RFC6550]).
4. As stated in section 5.2 of [RFC8505], the 6LN can register to more than one 6LR at the same time. In that case, it uses the same EARO for all of the parallel Address Registrations, with the exception of the Registration Lifetime field and the setting of the R flag that may differ. The 6LN may cancel a subset of its registrations, or transfer a registration from one or more old 6LR(s) to one or more new 6LR(s). To do so, the 6LN sends a series of NS(EARO) messages, all with the same TID, with a zero Registration Lifetime to the old 6LR(s) and with a non-zero

Registration Lifetime to the new 6LR(s). In that process, the 6LN SHOULD send the NS(EARO) with a non-zero Registration Lifetime and ensure that at least one succeeds before it sends an NS(EARO) that terminates another registration. This avoids the churn related to transient route invalidation in the RPL network above the common parent of the involved 6LRs.

5. Following section 5.1 of [RFC8505], a 6LN acting as a RUL sets the R Flag in the EARO of its registration(s) for which it requires routing services. If the R Flag is not echoed in the NA, the RUL MUST consider that establishing the routing services via this 6LR failed and it SHOULD attempt to use another 6LR. The RUL SHOULD ensure that one registration succeeds before setting the R Flag to 0. In case of a conflict with the preceding rule on lifetime, the rule on lifetime has precedence.
6. The 6LN may use any of the 6LRs to which it registered as the default gateway. Using a 6LR to which the 6LN is not registered may result in packets dropped at the 6LR by a Source Address Validation function (SAVI) [RFC7039] so it is not recommended.

Even without support for RPL, the RUL may be configured with an opaque value to be provided to the routing protocol. If the RUL has knowledge of the RPL Instance the packet should be injected into, then it SHOULD set the Opaque field in the EARO to the RPLInstanceID, otherwise it MUST leave the Opaque field as zero.

Regardless of the setting of the Opaque field, the 6LN MUST set the "I" field to zero to signal "topological information to be passed to a routing process", as specified in section 5.1 of [RFC8505].

A RUL is not expected to produce RPL artifacts in the data packets, but it may do so. For instance, if the RUL has minimal awareness of the RPL Instance then it can build an RPI. A RUL that places an RPI in a data packet SHOULD indicate the RPLInstanceID of the RPL Instance where the packet should be forwarded. It is up to the 6LR (e.g., by policy) to use the RPLInstanceID information provided by the RUL or rewrite it to the selected RPLInstanceID for forwarding inside the RPL domain. All the flags and the Rank field are set to 0 as specified by section 11.2 of [RFC6550].

9.2.2. Perspective of the 6LR Acting as Border router

A 6LR that provides reachability services for a RUL in a RPL network as specified in this document MUST include a 6CIO in its RA messages and set the L, P and E flags to 1 as prescribed by [RFC8505].

As prescribed by [RFC8505], the 6LR generates an EDAR message upon reception of a valid NS(EARO) message for the registration of a new IPv6 address by a 6LN. If the initial EDAR/EDAC exchange succeeds, then the 6LR installs an NCE for the Registration Lifetime.

If the R Flag is set to 1 in the NS(EARO), the 6LR SHOULD inject the host route in RPL, unless this is barred for other reasons, such as the saturation of the RPL parents. The 6LR MUST use a RPL Non-Storing Mode signaling and the updated Target Option (see Section 6.1). The 6LR SHOULD refrain from setting the 'X' flag to avoid a redundant EDAR/EDAC flow to the 6LBR. The 6LR MUST request a DAO-ACK by setting the 'K' flag in the DAO message. Success injecting the route to the RUL's address is indicated by the 'E' flag set to 0 in the RPL status of the DAO-ACK message.

For the registration refreshes, if the RPL Root sets the 'P' flag in the DODAG Configuration Option to 1, then the 6LR MUST refrain from sending the keep-alive EDAR; instead, it MUST set the 'X' flag to 1 in the Target Option of the DAO messages, to request that the Root proxies the keep-alive EDAR/EDAC exchange with the 6LBR (see Section 6); if the 'P' flag is set to 0 then the 6LR MUST set the 'X' flag to 0 and handle the EDAR/EDAC flow itself.

The Opaque field in the EARO provides a means to signal which RPL Instance is to be used for the DAO advertisements and the forwarding of packets sourced at the Registered Address when there is no RPI in the packet.

As described in [RFC8505], if the "I" field is zero, then the Opaque field is expected to carry the RPLInstanceID suggested by the 6LN; otherwise, there is no suggested Instance. If the 6LR participates in the suggested RPL Instance, then the 6LR MUST use that RPL Instance for the Registered Address.

If there is no suggested RPL Instance or else if the 6LR does not participate to the suggested Instance, it is expected that the packets coming from the 6LN "can unambiguously be associated to at least one RPL Instance" [RFC6550] by the 6LR, e.g., using a policy that maps the 6-tuple into an Instance.

The DAO message advertising the Registered Address MUST be constructed as follows:

1. The Registered Address is signaled as the Target Prefix in the updated Target Option in the DAO message; the Prefix Length is set to 128 but the 'F' flag is set to 0 since the advertiser is not the RUL. The ROVR field is copied unchanged from the EARO (see Section 6.1).

2. The 6LR indicates one of its global or unique-local IPv6 unicast addresses as the Parent Address in the TIO associated with the Target Option
3. The 6LR sets the External 'E' flag in the TIO to indicate that it is redistributing an external target into the RPL network
4. The Path Lifetime in the TIO is computed from the Registration Lifetime in the EARO. This operation converts seconds to the Lifetime Units used in the RPL operation. This creates the deployment constraint that the Lifetime Unit is reasonably compatible with the expression of the Registration Lifetime; e.g., a Lifetime Unit of 0x4000 maps the most significant byte of the Registration Lifetime to the Path Lifetime.

In that operation, the Path Lifetime must be set to ensure that the path has a longer lifetime than the registration and covers in addition the round trip time to the Root.

Note that if the Registration Lifetime is 0, then the Path Lifetime is also 0 and the DAO message becomes a No-Path DAO, which cleans up the routes down to the RUL's address; this also causes the Root as a proxy to send an EDAR message to the 6LBR with a Lifetime of 0.

5. the Path Sequence in the TIO is set to the TID value found in the EARO option.

Upon receiving or timing out the DAO-ACK after an implementation-specific number of retries, the 6LR MUST send the corresponding NA(EARO) to the RUL. Upon receiving an asynchronous DCO message, it MUST send an asynchronous NA(EARO) to the RUL immediately, but still be capable of processing the DAO-ACK if one is pending.

The 6LR MUST set the R Flag to 1 in the NA(EARO) back if and only if the 'E' flag in the RPL Status is set to 0, indicating that the 6LR injected the Registered Address in the RPL routing successfully and that the EDAR proxy operation succeeded.

If the 'A' flag in the RPL Status is set to 1, the embedded Status value is passed back to the RUL in the EARO Status. If the 'E' flag is also set to 1, the registration failed for 6LoWPAN-ND-related reasons, and the NCE is removed.

An error injecting the route causes the 'E' flag to be set to 1. If the error is not related to ND, the 'A' flag is set to 0. In that case, the registration succeeds, but the RPL route is not installed. So the NA(EARO) is returned with a status indicating success but the R Flag set to 0, which means that the 6LN obtained a binding but no route.

If the 'A' flag is set to 0 in the RPL Status of the DAO-ACK, then the 6LoWPAN ND operation succeeded, and an EARO Status of 0 (Success) MUST be returned to the 6LN. The EARO Status of 0 MUST also be used if the 6LR did not attempt to inject the route but could create the binding after a successful EDAR/EDAC exchange or refresh it.

If the 'E' flag is set to 1 in the RPL Status of the DAO-ACK, then the route was not installed and the R flag MUST be set to 0 in the NA(EARO). The R flag MUST be set to 0 if the 6LR did not attempt to inject the route.

In a network where Address Protected Neighbor Discovery (AP-ND) is enabled, in case of a DAO-ACK or a DCO transporting an EARO Status value of 5 (Validation Requested), the 6LR MUST challenge the 6LN for ownership of the address, as described in section 6.1 of [RFC8928], before the Registration is complete. This flow, illustrated in Figure 10, ensures that the address is validated before it is injected in the RPL routing.

If the challenge succeeds, then the operations continue as normal. In particular, a DAO message is generated upon the NS(EARO) that proves the ownership of the address. If the challenge failed, the 6LR rejects the registration as prescribed by AP-ND and may take actions to protect itself against DoS attacks by a rogue 6LN, see Section 11.

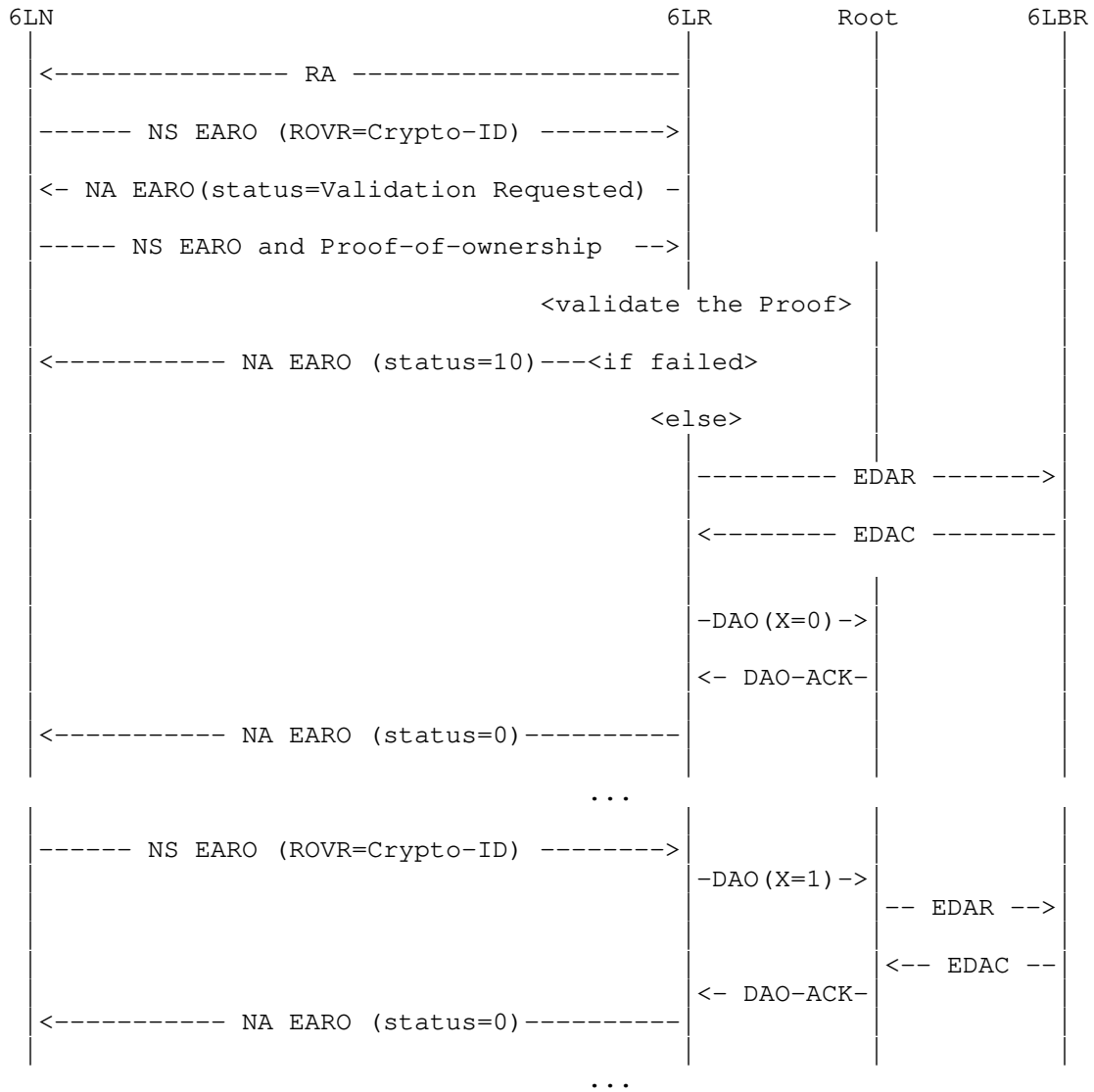


Figure 10: Address Protection

The 6LR may at any time send a unicast asynchronous NA(EARO) with the R Flag set to 0 to signal that it stops providing routing services, and/or with the EARO Status 2 "Neighbor Cache full" to signal that it removes the NCE. It may also send a final RA, unicast or multicast, with a router Lifetime field of zero, to signal that it is ceasing to serve as router, as specified in section 6.2.5 of [RFC4861]. This may happen upon a DCO or a DAO-ACK message indicating the path is already removed; else the 6LR MUST remove the host route to the 6LN using a DAO message with a Path Lifetime of zero.

A valid NS(EARO) message with the R Flag set to 0 and a Registration Lifetime that is not zero signals that the 6LN wishes to maintain the binding but does not require the routing services from the 6LR (any more). Upon this message, if, due to previous NS(EARO) with the R Flag set to 1, the 6LR was injecting the host route to the Registered Address in RPL using DAO messages, then the 6LR MUST invalidate the host route in RPL using a DAO with a Path Lifetime of zero. It is up to the Registering 6LN to maintain the corresponding route from then on, either keeping it active via a different 6LR or by acting as a RAN and managing its own reachability.

When forwarding a packet from the RUL into the RPL domain, if the packet does not have an RPI then the 6LR MUST encapsulate the packet to the Root, and add an RPI. If there is an RPI in the packet, the 6LR MUST rewrite the RPI but it does not need to encapsulate.

9.2.3. Perspective of the RPL Root

A RPL Root MUST set the 'P' flag to 1 in the RPL DODAG Configuration Option of the DIO messages that it generates (see Section 6) to signal that it proxies the EDAR/EDAC exchange and supports the Updated RPL Target option.

Upon reception of a DAO message, for each updated RPL Target Option (see Section 6.1) with the 'X' flag set to 1, the Root MUST notify the 6LBR by using a proxied EDAR/EDAC exchange; if the RPL Root and the 6LBR are integrated, an internal API can be used instead.

The EDAR message MUST be constructed as follows:

1. The Target IPv6 address from the RPL Target Option is placed in the Registered Address field of the EDAR message;
2. the Registration Lifetime is adapted from the Path Lifetime in the TIO by converting the Lifetime Units used in RPL into units of 60 seconds used in the 6LoWPAN ND messages;

3. The TID value is set to the Path Sequence in the TIO and indicated with an ICMP code of 1 in the EDAR message;
4. The ROVR in the RPL Target Option is copied as is in the EDAR and the ICMP Code Suffix is set to the appropriate value as shown in Table 4 of [RFC8505] depending on the size of the ROVR field.

Upon receiving an EDAC message from the 6LBR, if a DAO is pending, then the Root MUST send a DAO-ACK back to the 6LR. Otherwise, if the Status in the EDAC message is not "Success", then it MUST send an asynchronous DCO to the 6LR.

In either case, the EDAC Status is embedded in the RPL Status with the 'A' flag set to 1.

The proxied EDAR/EDAC exchange MUST be protected with a timer of an appropriate duration and a number of retries, that are implementation-dependent, and SHOULD be configurable since the Root and the 6LBR are typically nodes with a higher capacity and manageability than 6LRs. Upon timing out, the Root MUST send an error back to the 6LR as above, either using a DAO-ACK or a DCO, as appropriate, with the 'A' and 'E' flags set to 1 in the RPL status, and a RPL Status value of "6LBR Registry Saturated" [RFC8505].

9.2.4. Perspective of the 6LBR

The 6LBR is unaware that the RPL Root is not the new attachment 6LR of the RUL, so it is not impacted by this specification.

Upon reception of an EDAR message, the 6LBR acts as prescribed by [RFC8505] and returns an EDAC message to the sender.

10. Protocol Operations for Multicast Addresses

Section 12 of [RFC6550] details the RPL support for multicast flows. This support is activated by the MOP of 3 ("Storing Mode of Operation with multicast support") in the DIO messages that form the DODAG. This section also applies if and only if the MOP of the RPLInstance is 3.

The RPL support of multicast is not source-specific and only operates as an extension to the Storing Mode of Operation for unicast packets. Note that it is the RPL model that the multicast packet is passed as a Layer-2 unicast to each of the interested children. This remains true when forwarding between the 6LR and the listener 6LN.

"Multicast Listener Discovery Version 2 (MLDv2) for IPv6" [RFC3810] provides an interface for a listener to register to multicast flows. In the MLD model, the router is a "querier", and the host is a multicast listener that registers to the querier to obtain copies of the particular flows it is interested in.

The equivalent of the first Address Registration happens as illustrated in Figure 11. The 6LN, as an MLD listener, sends an unsolicited Report to the 6LR. This enables it to start receiving the flow immediately, and causes the 6LR to inject the multicast route in RPL.

This specification does not change MLD but will operate more efficiently if the asynchronous messages for unsolicited Report and Done are sent by the 6LN as Layer-2 unicast to the 6LR, in particular on wireless.

The 6LR acts as a generic MLD querier and generates a DAO with the Multicast Address as the Target Prefix as described in section 12 of [RFC6550]. As for the Unicast host routes, the Path Lifetime associated to the Target is mapped from the Query Interval, and set to be larger to account for variable propagation delays to the Root. The Root proxies the MLD exchange as a listener with the 6LBR acting as the querier, so as to get packets from a source external to the RPL domain.

Upon a DAO with a Target option for a multicast address, the RPL Root checks if it is already registered as a listener for that address, and if not, it performs its own unsolicited Report for the multicast address as described in section 5.1 of [RFC3810]. The report is source independent, so there is no Source Address listed.

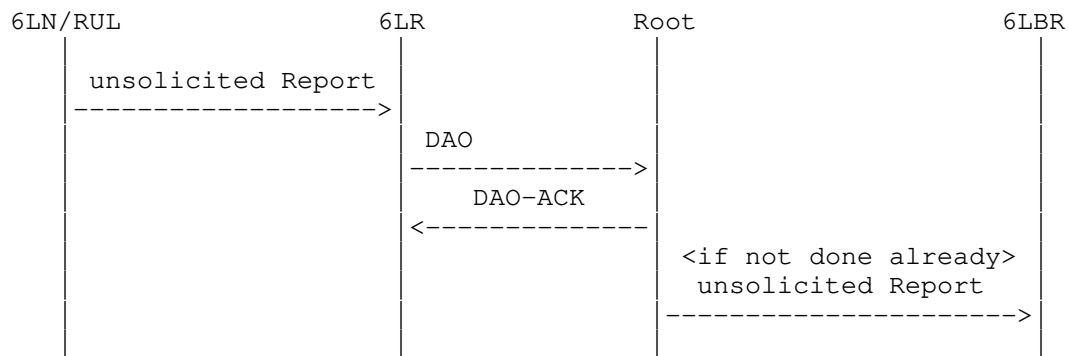


Figure 11: First Multicast Registration Flow

The equivalent of the registration refresh is pulled periodically by the 6LR acting as querier. Upon the timing out of the Query Interval, the 6LR sends a Multicast Address Specific Query to each of its listeners, for each Multicast Address, and gets a Report back that is mapped into a DAO one by one. Optionally, the 6LR MAY send a General Query, where the Multicast Address field is set to zero. In that case, the multicast packet is passed as a Layer-2 unicast to each of the interested children. .

Upon a Report, the 6LR generates a DAO with as many Target Options as there are Multicast Address Records in the Report message, copying the Multicast Address field in the Target Prefix of the RPL Target Option. The DAO message is a Storing Mode DAO, passed to a selection of the 6LR's parents.

Asynchronously to this, a similar procedure happens between the Root and a router such as the 6LBR that serves multicast flows on the Link where the Root is located. Again the Query and Report messages are source independent. The Root lists exactly once each Multicast Address for which it has at least one active multicast DAO state, copying the multicast address in the DAO state in the Multicast Address field of the Multicast Address Records in the Report message.

This is illustrated in Figure 12:

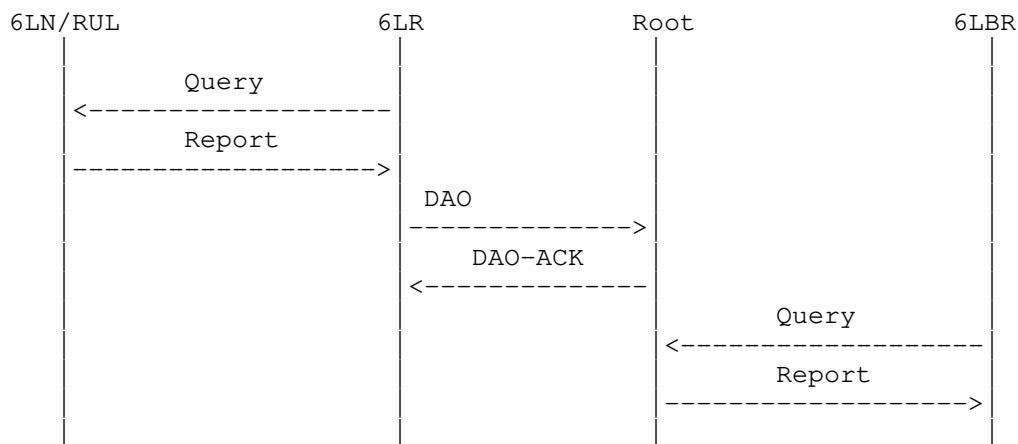


Figure 12: Next Registration Flow

Note that any of the functions 6LR, Root and 6LBR might be collapsed in a single node, in which case the flow above happens internally, and possibly through internal API calls as opposed to messaging.

11. Security Considerations

It is worth noting that with [RFC6550], every node in the LLN is RPL-aware and can inject any RPL-based attack in the network. This specification improves the situation by isolating edge nodes that can only interact with the RPL routers using 6LoWPAN ND, meaning that they cannot perform RPL insider attacks.

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), Denial-Of-Service attacks whereby a rogue 6LR creates a high churn in the RPL network by advertising and removing many forged addresses, or bombing attack whereby an impersonated 6LBR would destroy state in the network by using the status code of 4 ("Removed").

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 [RFC6550], [RFC7416] [RFC6775], and [RFC8505] apply to this specification as well.

The Link-Layer security is needed in particular to prevent Denial-Of-Service attacks whereby a rogue 6LN creates a high churn in the RPL network by constantly registering and deregistering addresses with the R Flag set to 1 in the EARO.

[RFC8928] updated 6LoWPAN ND with the called Address-Protected Neighbor Discovery (AP-ND). AP-ND protects the owner of an address against address theft and impersonation attacks in a Low-Power and Lossy Network (LLN). Nodes supporting the extension compute a cryptographic identifier (Crypto-ID), and use it with one or more of their Registered Addresses. The Crypto-ID identifies the owner of the Registered Address and can be used to provide proof of ownership of the Registered Addresses. Once an address is registered with the Crypto-ID and a proof of ownership is provided, only the owner of that address can modify the registration information, thereby enforcing Source Address Validation. [RFC8928] reduces even more the attack perimeter that is available to the edge nodes and its use is suggested in this specification.

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 6LBR or a RPL Root is entitled to do so.

The Opaque field in the EARO enables the RUL to suggest a RPLInstanceID where its traffic is placed. It is also possible for an attacker RUL to include an RPI in the packet. This opens to attacks where a RPL instance would be reserved for critical traffic, e.g., with a specific bandwidth reservation, that the additional traffic generated by a rogue may disrupt. The attack may be alleviated by traditional access control and traffic shaping mechanisms where the 6LR controls the incoming traffic from the 6LN. More importantly, the 6LR is the node that injects the traffic in the RPL domain, so it has the final word on which RPLInstance is to be used for the traffic coming from the RUL, per its own policy. In particular, a policy can override the formal language that forces to use the Opaque field or to rewrite the RPI provided by the RUL, in a situation where the network administrator finds it relevant.

At the time of this writing, RPL does not have a Route Ownership Validation model whereby it is possible to validate the origin of an address that is injected in a DAO. This specification makes a first step in that direction by allowing the Root to challenge the RUL via the 6LR that serves it.

Section 6.1 indicates that when the length of the ROVR field is unknown, the RPL Target Option must be passed on as received in RPL storing Mode. This creates a possible opening for using DAO messages as a covert channel. Note that DAO messages are rare and overusing that channel could be detected. An implementation SHOULD notify the network management when a RPL Target Option is received with an unknown ROVR field size, to ensure that the situation is known to the network administrator.

[EFFICIENT-NPDAO] introduces the ability for a rogue common ancestor node to invalidate a route on behalf of the target node. In this case, the RPL Status in the DCO has the 'A' flag set to 0, and a NA(EARO) is returned to the 6LN with the R flag set to 0. This encourages the 6LN to try another 6LR. If a 6LR exists that does not use the rogue common ancestor, then the 6LN will eventually succeed gaining reachability over the RPL network in spite of the rogue node.

12. IANA Considerations

12.1. Fixing the Address Registration Option Flags

Section 9.1 of [RFC8505] creates a Registry for the 8-bit Address Registration Option Flags field. IANA is requested to rename the first column of the table from "ARO Status" to "Bit number".

12.2. Resizing the ARO Status values

Section 12 of [RFC6775] creates the Address Registration Option Status values Registry with a range 0-255.

This specification reduces that range to 0-63, see Section 6.3.

IANA is requested to modify the Address Registration Option Status values Registry so that the upper bound of the unassigned values is 63. This document should be added as a reference. The registration procedure does not change.

12.3. New RPL DODAG Configuration Option Flag

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

Bit Number	Capability Description	Reference
1 (suggested)	Root Proxies EDAR/EDAC (P)	THIS RFC

Table 2: New DODAG Configuration Option Flag

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

12.4. RPL Target Option Registry

This document modifies the "RPL Target Option Flags" registry initially created in Section 20.15 of [RFC6550]. The registry now includes only 4 bits (Section 6.1) and should point to this document as an additional reference. The registration procedure does not change.

Section 6.1 also defines 2 new entries in the Registry as follows:

Bit Number	Capability Description	Reference
0 (suggested)	Advertiser address in Full (F)	THIS RFC
1 (suggested)	Proxy EDAR Requested (X)	THIS RFC

Table 3: RPL Target Option Registry

12.5. New Subregistry for RPL Non-Rejection Status values

This specification creates a new Subregistry for the RPL Non-Rejection Status values for use in the RPL DAO-ACK, DCO, and DCO-ACK messages with the 'A' flag set to 0, under the RPL registry.

- * Possible values are 6-bit unsigned integers (0..63).
- * Registration procedure is "IETF Review" [RFC8126].
- * Initial allocation is as indicated in Table 4:

Value	Meaning	Reference
0	Unqualified acceptance	THIS RFC / RFC 6550
1..63	Unassigned	

Table 4: Acceptance values of the RPL Status

12.6. New Subregistry for RPL Rejection Status values

This specification creates a new Subregistry for the RPL Rejection Status values for use in the RPL DAO-ACK and DCO messages with the 'A' flag set to 0, under the RPL registry.

- * Possible values are 6-bit unsigned integers (0..63).
- * Registration procedure is "IETF Review" [RFC8126].
- * Initial allocation is as indicated in Table 5:

Value	Meaning	Reference
0	Unqualified rejection	THIS RFC
1 (suggested in [EFFICIENT-NPDAO])	No routing entry	[EFFICIENT-NPDAO]
2..63	Unassigned	

Table 5: Rejection values of the RPL Status

13. Acknowledgments

The authors wish to thank Ines Robles, Georgios Papadopoulos and especially Rahul Jadhav and Alvaro Retana for their reviews and contributions to this document. Also many thanks to Eric Vyncke, Erik Kline, Murray Kucherawy, Peter Van der Stok, Carl Wallace, Barry Leiba, Julien Meuric, and especially Benjamin Kaduk and Elwyn Davies, for their reviews and useful comments during the IETF Last Call and the IESG review sessions.

14. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [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>>.
- [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <<https://www.rfc-editor.org/info/rfc7400>>.

- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <<https://www.rfc-editor.org/info/rfc8504>>.
- [RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <<https://www.rfc-editor.org/info/rfc8505>>.
- [RFC8928] Thubert, P., Ed., Sarikaya, B., Sethi, M., and R. Struik, "Address-Protected Neighbor Discovery for Low-Power and Lossy Networks", RFC 8928, DOI 10.17487/RFC8928, November 2020, <<https://www.rfc-editor.org/info/rfc8928>>.
- [USEofRPLinfo] Robles, 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", Work in Progress, Internet-Draft, draft-ietf-roll-useofrplinfo-43, 10 January 2021, <<https://tools.ietf.org/html/draft-ietf-roll-useofrplinfo-43>>.
- [EFFICIENT-NPDAO] Jadhav, R., Thubert, P., Sahoo, R., and Z. Cao, "Efficient Route Invalidation", Work in Progress, Internet-Draft, draft-ietf-roll-efficient-npdao-18, 15 April 2020, <<https://tools.ietf.org/html/draft-ietf-roll-efficient-npdao-18>>.

15. Informative References

- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <<https://www.rfc-editor.org/info/rfc4919>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [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>>.
- [RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, DOI 10.17487/RFC6606, May 2012, <<https://www.rfc-editor.org/info/rfc6606>>.
- [RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <<https://www.rfc-editor.org/info/rfc7039>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [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>>.

- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [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>>.
- [RFC6687] Tripathi, J., Ed., de Oliveira, J., Ed., and JP. Vasseur, Ed., "Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6687, DOI 10.17487/RFC6687, October 2012, <<https://www.rfc-editor.org/info/rfc6687>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<https://www.rfc-editor.org/info/rfc8025>>.
- [RFC8929] Thubert, P., Ed., Perkins, C.E., and E. Levy-Abegnoli, "IPv6 Backbone Router", RFC 8929, DOI 10.17487/RFC8929, November 2020, <<https://www.rfc-editor.org/info/rfc8929>>.

Appendix A. Example Compression

Figure 13 illustrates the case in Storing Mode where the packet is received from the Internet, then the Root encapsulates the packet to insert the RPI and deliver to the 6LR that is the parent and last hop to the final destination, which is not known to support [RFC8138].

```

+-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ...
|11110001|SRH-6LoRH|RPI- |IP-in-IP|NH=1 |11110001|UDP |UDP
|Page 1 |Type1 S=0|6LoRH |6LoRH |LOWPAN_IPHC|UDP |hdr |Payld
+-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ... +-+ ...
          <-4 bytes->          <- RFC 6282 ->
                               <- No RPL artifact ...

```

Figure 13: Encapsulation to Parent 6LR in Storing Mode

The difference with the example presented in Figure 19 of [RFC8138] is the addition of a SRH-6LoRH before the RPI-6LoRH to transport the compressed address of the 6LR as the destination address of the outer IPv6 header. In the [RFC8138] example the destination IP of the outer header was elided and was implicitly the same address as the destination of the inner header. Type 1 was arbitrarily chosen, and the size of 0 denotes a single address in the SRH.

In Figure 13, the source of the IPv6-in-IPv6 encapsulation is the Root, so it is elided in the IPv6-in-IPv6 6LoRH. The destination is the parent 6LR of the destination of the encapsulated packet so it cannot be elided. If the DODAG is operated in Storing Mode, it is the single entry in the SRH-6LoRH and the SRH-6LoRH Size is encoded as 0. The SRH-6LoRH is the first 6LoRH in the chain. In this particular example, the 6LR address can be compressed to 2 bytes so a Type of 1 is used. It results that the total length of the SRH-6LoRH is 4 bytes.

In Non-Storing Mode, the encapsulation from the Root would be similar to that represented in Figure 13 with possibly more hops in the SRH-6LoRH and possibly multiple SRH-6LoRHs if the various addresses in the routing header are not compressed to the same format. Note that on the last hop to the parent 6LR, the RH3 is consumed and removed from the compressed form, so the use of Non-Storing Mode vs. Storing Mode is indistinguishable from the packet format.

The SRH-6LoRHs are followed by RPI-6LoRH and then the IPv6-in-IPv6 6LoRH. When the IPv6-in-IPv6 6LoRH is removed, all the 6LoRH Headers that precede it are also removed. The Paging Dispatch [RFC8025] may also be removed if there was no previous Page change to a Page other than 0 or 1, since the LOWPAN_IPHC is encoded in the same fashion in the default Page 0 and in Page 1. The resulting packet to the destination is the encapsulated packet compressed with [RFC6282].

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

Michael C. Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
URI: <http://www.sandelman.ca/>

ROLL Working Group
Internet-Draft
Updates: 6553, 6550, 8138 (if approved)
Intended status: Standards Track
Expires: July 19, 2021

M. Robles
UTN-FRM/Aalto
M. Richardson
SSW
P. Thubert
Cisco
January 15, 2021

Using RPI Option Type, Routing Header for Source Routes and IPv6-in-IPv6
encapsulation in the RPL Data Plane
draft-ietf-roll-useofrplinfo-44

Abstract

This document looks at different data flows through LLN (Low-Power and Lossy Networks) where RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) is used to establish routing. The document enumerates the cases where RFC6553 (RPI Option Type), RFC6554 (Routing Header for Source Routes) and IPv6-in-IPv6 encapsulation is required in data plane. This analysis provides the basis on which to design efficient compression of these headers. This document updates RFC6553 adding a change to the RPI Option Type. Additionally, this document updates RFC6550 defining a flag in the DIO Configuration option to indicate about this change and updates RFC8138 as well to consider the new Option Type when the RPL Option is decompressed.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 19, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Overview	4
2. Terminology and Requirements Language	5
3. RPL Overview	6
4. Updates to RFC6550, RFC6553 and RFC8138	7
4.1. Updates to RFC6550	7
4.1.1. Advertising External Routes with Non-Storing Mode Signaling.	7
4.1.2. Configuration Options and Mode of Operation	8
4.1.3. Indicating the new RPI in the DODAG Configuration option Flag.	9
4.2. Updates to RFC6553: Indicating the new RPI Option Type.	10
4.3. Updates to RFC8138: Indicating the way to decompress with the new RPI Option Type.	13
5. Sample/reference topology	14
6. Use cases	16
7. Storing mode	19
7.1. Storing Mode: Interaction between Leaf and Root	20
7.1.1. SM: Example of Flow from RAL to Root	21
7.1.2. SM: Example of Flow from Root to RAL	22
7.1.3. SM: Example of Flow from Root to RUL	22
7.1.4. SM: Example of Flow from RUL to Root	24
7.2. SM: Interaction between Leaf and Internet.	25
7.2.1. SM: Example of Flow from RAL to Internet	25
7.2.2. SM: Example of Flow from Internet to RAL	27
7.2.3. SM: Example of Flow from RUL to Internet	28
7.2.4. SM: Example of Flow from Internet to RUL.	29
7.3. SM: Interaction between Leaf and Leaf	30
7.3.1. SM: Example of Flow from RAL to RAL	30
7.3.2. SM: Example of Flow from RAL to RUL	31

7.3.3.	SM: Example of Flow from RUL to RAL	33
7.3.4.	SM: Example of Flow from RUL to RUL	34
8.	Non Storing mode	35
8.1.	Non-Storing Mode: Interaction between Leaf and Root . . .	37
8.1.1.	Non-SM: Example of Flow from RAL to root	37
8.1.2.	Non-SM: Example of Flow from root to RAL	38
8.1.3.	Non-SM: Example of Flow from root to RUL	39
8.1.4.	Non-SM: Example of Flow from RUL to root	40
8.2.	Non-Storing Mode: Interaction between Leaf and Internet .	41
8.2.1.	Non-SM: Example of Flow from RAL to Internet	41
8.2.2.	Non-SM: Example of Flow from Internet to RAL	43
8.2.3.	Non-SM: Example of Flow from RUL to Internet	44
8.2.4.	Non-SM: Example of Flow from Internet to RUL	45
8.3.	Non-SM: Interaction between leaves	46
8.3.1.	Non-SM: Example of Flow from RAL to RAL	46
8.3.2.	Non-SM: Example of Flow from RAL to RUL	49
8.3.3.	Non-SM: Example of Flow from RUL to RAL	51
8.3.4.	Non-SM: Example of Flow from RUL to RUL	52
9.	Operational Considerations of supporting RUL-leaves	53
10.	Operational considerations of introducing 0x23	54
11.	IANA Considerations	54
11.1.	Option Type in RPL Option	54
11.2.	Change to the DODAG Configuration Options Flags registry	55
11.3.	Change MOP value 7 to Reserved	55
12.	Security Considerations	56
13.	Acknowledgments	59
14.	References	59
14.1.	Normative References	60
14.2.	Informative References	61
	Authors' Addresses	63

1. Introduction

RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [RFC6550] is a routing protocol for constrained networks. [RFC6553] defines the RPL Option carried within the IPv6 Hop-by-Hop Header to carry the RPLInstanceID and quickly identify inconsistencies (loops) in the routing topology. The RPL Option is commonly referred to as the RPL Packet Information (RPI) though the RPI is the routing information that is defined in [RFC6550] and transported in the RPL Option. RFC6554 [RFC6554] defines the "RPL Source Route Header" (RH3), an IPv6 Extension Header to deliver datagrams within a RPL routing domain, particularly in non-storing mode.

These various items are referred to as RPL artifacts, and they are seen on all of the data-plane traffic that occurs in RPL routed networks; they do not in general appear on the RPL control plane

traffic at all which is mostly Hop-by-Hop traffic (one exception being DAO messages in non-storing mode).

It has become clear from attempts to do multi-vendor interoperability, and from a desire to compress as many of the above artifacts as possible that not all implementers agree when artifacts are necessary, or when they can be safely omitted, or removed.

The ROLL WG analyzed how [RFC2460] rules apply to storing and non-storing use of RPL. The result was 24 data plane use cases. They are exhaustively outlined here in order to be completely unambiguous. During the processing of this document, new rules were published as [RFC8200], and this document was updated to reflect the normative changes in that document.

This document updates [RFC6553], changing the value of the Option Type of the RPL Option to make [RFC8200] routers ignore this option when not recognized.

A Routing Header Dispatch for 6LoWPAN (6LoRH) ([RFC8138]) defines a mechanism for compressing RPL Option information and Routing Header type 3 (RH3) [RFC6554], as well as an efficient IPv6-in-IPv6 technique.

Most of the use cases described herein require the use of IPv6-in-IPv6 packet encapsulation. When encapsulating and decapsulating packets, [RFC6040] MUST be applied to map the setting of the explicit congestion notification (ECN) field between inner and outer headers. Additionally, [I-D.ietf-intarea-tunnels] is recommended reading to explain the relationship of IP tunnels to existing protocol layers and the challenges in supporting IP tunneling.

Non-constrained uses of RPL are not in scope of this document, and applicability statements for those uses may provide different advice, E.g. [I-D.ietf-anima-autonomic-control-plane].

1.1. Overview

The rest of the document is organized as follows: Section 2 describes the used terminology. Section 3 provides a RPL Overview. Section 4 describes the updates to RFC6553, RFC6550 and RFC 8138. Section 5 provides the reference topology used for the uses cases. Section 6 describes the use cases included. Section 7 describes the storing mode cases and section 8 the non-storing mode cases. Section 9 describes the operational considerations of supporting RPL-unaware-leaves. Section 10 depicts operational considerations for the proposed change on RPL Option Type, section 11 the IANA considerations and then section 12 describes the security aspects.

2. Terminology and Requirements Language

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

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

Consumed: A Routing Header is consumed when the Segments Left field is zero, which indicates that the destination in the IPv6 header is the final destination of the packet and that the hops in the Routing Header have been traversed.

RPL Leaf: An IPv6 host that is attached to a RPL router and obtains connectivity through a RPL Destination Oriented Directed Acyclic Graph (DODAG). As an IPv6 node, a RPL Leaf is expected to ignore a consumed Routing Header and as an IPv6 host, it is expected to ignore a Hop-by-Hop header. It results that a RPL Leaf can correctly receive a packet with RPL artifacts. On the other hand, a RPL Leaf is not expected to generate RPL artifacts or to support IP-in-IP encapsulation. For simplification, this document uses the standalone term leaf to mean a RPL leaf.

RPL Packet Information (RPI): The information defined abstractly in [RFC6550] to be placed in IP packets. The term is commonly used, including in this document, to refer to the RPL Option [RFC6553] that transports that abstract information in an IPv6 Hop-by-Hop Header. [RFC8138] provides an alternate (more compressed) formatting for the same abstract information.

RPL-aware-node (RAN): A device which implements RPL. Please note that the device can be found inside the LLN or outside LLN.

RPL-Aware-Leaf(RAL): A RPL-aware-node that is also a RPL Leaf.

RPL-unaware-node: A device which does not implement RPL, thus the device is not-RPL-aware. Please note that the device can be found inside the LLN.

RPL-Unaware-Leaf(RUL): A RPL-unaware-node that is also a RPL Leaf.

6LoWPAN Node (6LN): [RFC6775] defines it as: "A 6LoWPAN node is any host or router participating in a LoWPAN. This term is used when referring to situations in which either a host or router can play the role described.". In this document, a 6LN acts as a leaf.

6LoWPAN Router (6LR): [RFC6775] defines it as:" An intermediate router in the LoWPAN that is able to send and receive Router Advertisements (RAs) and Router Solicitations (RSs) as well as forward and route IPv6 packets. 6LoWPAN routers are present only in route-over topologies."

6LoWPAN Border Router (6LBR): [RFC6775] defines it as:"A border router located at the junction of separate 6LoWPAN networks or between a 6LoWPAN network and another IP network. There may be one or more 6LBRs at the 6LoWPAN network boundary. A 6LBR is the responsible authority for IPv6 prefix propagation for the 6LoWPAN network it is serving. An isolated LoWPAN also contains a 6LBR in the network, which provides the prefix(es) for the isolated network."

Flag Day: A Flag Day is caused when a network is reconfigured in a way that nodes running the older configuration can not communicate with nodes running the new configuration. For instance, when the ARPANET changed from IP version 3 to IP version 4 on January 1, 1983 ([RFC0801]). In the context of this document, a switch from RPI Option Type (0x63) and Option Type (0x23) presents as a disruptive changeover. In order to reduce the amount of time for such a changeover, Section 4.1.3 provides a mechanism to allow nodes to be incrementally upgraded.

Non-Storing Mode (Non-SM): RPL mode of operation in which the RPL-aware-nodes send information to the root about their parents. Thus, the root knows the topology. Because the root knows the topology, the intermediate 6LRs do not maintain routing state and source routing is needed.

Storing Mode (SM): RPL mode of operation in which RPL-aware-nodes (6LRs) maintain routing state (of the children) so that source routing is not needed.

Note: Due to lack of space in some figures (tables) we refer to IPv6-in-IPv6 as IP6-IP6.

3. RPL Overview

RPL defines the RPL Control messages (control plane), a new ICMPv6 [RFC4443] message with Type 155. DIS (DODAG Information Solicitation), DIO (DODAG Information Object) and DAO (Destination Advertisement Object) messages are all RPL Control messages but with different Code values. A RPL Stack is shown in Figure 1.

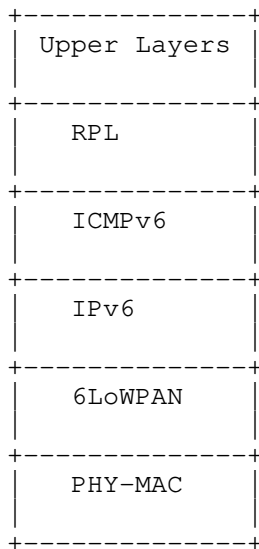


Figure 1: RPL Stack.

RPL supports two modes of Downward internal traffic: in storing mode (SM), it is fully stateful; in non-storing mode (Non-SM), it is fully source routed. A RPL Instance is either fully storing or fully non-storing, i.e. a RPL Instance with a combination of a fully storing and non-storing nodes is not supported with the current specifications at the time of writing this document. External routes are advertised with non-storing-mode messaging even in a storing mode network, see Section 4.1.1

4. Updates to RFC6550, RFC6553 and RFC8138

4.1. Updates to RFC6550

4.1.1. Advertising External Routes with Non-Storing Mode Signaling.

Section 6.7.8. of [RFC6550] introduces the 'E' flag that is set to indicate that the 6LR that generates the DAO redistributes external targets into the RPL network. An external Target is a Target that has been learned through an alternate protocol, for instance a route to a prefix that is outside the RPL domain but reachable via a 6LR. Being outside of the RPL domain, a node that is reached via an external target cannot be guaranteed to ignore the RPL artifacts and cannot be expected to process the [RFC8138] compression correctly. This means that the RPL artifacts should be contained in an IP-in-IP encapsulation that is removed by the 6LR, and that any remaining

compression should be expanded by the 6LR before it forwards a packet outside the RPL domain.

This specification updates [RFC6550] to RECOMMEND that external targets are advertised using Non-Storing Mode DAO messaging even in a Storing-Mode network. This way, external routes are not advertised within the DODAG and all packets to an external target reach the Root like normal Non-Storing Mode traffic. The Non-Storing Mode DAO informs the Root of the address of the 6LR that injects the external route, and the root uses IP-in-IP encapsulation to that 6LR, which terminates the IP-in-IP tunnel and forwards the original packet outside the RPL domain free of RPL artifacts.

In the other direction, for traffic coming from an external target into the LLN, the parent (6LR) that injects the traffic always encapsulates to the root. This whole operation is transparent to intermediate routers that only see traffic between the 6LR and the Root, and only the Root and the 6LRs that inject external routes in the network need to be upgraded to add this function to the network.

A RUL is a special case of external target when the target is actually a host and it is known to support a consumed Routing Header and to ignore a Hop-by-Hop header as prescribed by [RFC8200]. The target may have been learned through an external routing protocol or may have been registered to the 6LR using [RFC8505].

In order to enable IP-in-IP all the way to a 6LN, it is beneficial that the 6LN supports decapsulating IP-in-IP, but that is not assumed by [RFC8504]. If the 6LN is a RUL, the Root that encapsulates a packet SHOULD terminate the tunnel at a parent 6LR unless it is aware that the RUL supports IP-in-IP decapsulation.

A node that is reachable over an external route is not expected to support [RFC8138]. Whether a decapsulation took place or not and even when the 6LR is delivering the packet to a RUL, the 6LR that injected an external route MUST uncompress the packet before forwarding over that external route.

4.1.2. Configuration Options and Mode of Operation

Section 6.7.6 of RFC6550 describes the DODAG Configuration Option as containing a series of Flags in the first octet of the payload.

Anticipating future work to revise RPL relating to how the LLN and DODAG are configured, this document renames the DODAG Configuration Option Flags registry so that it applies to Mode of Operation (MOP) values zero (0) to six (6) only, leaving the flags unassigned for MOP value seven (7). The MOP is described in RFC6550 section 6.3.1.

In addition, this document reserves MOP value 7 for future expansion.

See Sections 11.2 and 11.3.

4.1.3. Indicating the new RPI in the DODAG Configuration option Flag.

In order to avoid a Flag Day caused by lack of interoperation between new RPI Option Type (0x23) and old RPI Option Type (0x63) nodes, this section defines a flag in the DIO Configuration option, to indicate when the new RPI Option Type can be safely used. This means, the flag is going to indicate the value of Option Type that the network will be using for the RPL Option. Thus, when a node joins to a network it will know which value to use. With this, RPL-capable nodes know if it is safe to use 0x23 when creating a new RPL Option. A node that forwards a packet with an RPI MUST NOT modify the Option Type of the RPL Option.

This is done using a DODAG Configuration option flag which will signal "RPI 0x23 enable" and propagate through the network. Section 6.3.1. of [RFC6550] defines a 3-bit Mode of Operation (MOP) in the DIO Base Object. The flag is defined only for MOP value between 0 to 6.

For a MOP value of 7, a node MUST use the RPI 0x23 option.

As stated in [RFC6550] the DODAG Configuration option is present in DIO messages. The DODAG Configuration option distributes configuration information. It is generally static, and does not change within the DODAG. This information is configured at the DODAG root and distributed throughout the DODAG with the DODAG Configuration option. Nodes other than the DODAG root do not modify this information when propagating the DODAG Configuration option.

Currently, the DODAG Configuration option in [RFC6550] states: "the unused bits MUST be initialized to zero by the sender and MUST be ignored by the receiver". If the flag is received with a value zero (which is the default), then new nodes will remain in RFC6553 Compatible Mode; originating traffic with the old-RPI Option Type (0x63) value. If the flag is received with a value of 1, then the value for the RPL Option MUST be set to 0x23.

Bit number three of the flag field in the DODAG Configuration option is to be used as shown in Figure 2 (which is the same as Figure 39 in Section 11 and is shown here for convenience):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 2: DODAG Configuration option Flag to indicate the RPI-flag-day.

In the case of reboot, the node (6LN or 6LR) does not remember the RPI Option Type (i.e., whether or not the flag is set), so the node will not trigger DIO messages until a DIO message is received indicating the RPI value to be used. The node will use the value 0x23 if the network supports this feature.

4.2. Updates to RFC6553: Indicating the new RPI Option Type.

This modification is required in order to be able to send, for example, IPv6 packets from a RPL-Aware-Leaf to a RPL-unaware node through Internet (see Section 7.2.1), without requiring IPv6-in-IPv6 encapsulation.

[RFC6553] (Section 6, Page 7) states as shown in Figure 3, that in the Option Type field of the RPL Option, the two high order bits must be set to '01' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node must discard the packet if it doesn't recognize the Option Type, and the third bit indicates that the Option Data may change in route. The remaining bits serve as the Option Type.

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x63	01	1	00011	RPL Option	[RFC6553]

Figure 3: Option Type in RPL Option.

This document illustrates that it is not always possible to know for sure at the source that a packet will only travel within the RPL domain or may leave it.

At the time [RFC6553] was published, leaking a Hop-by-Hop header in the outer IPv6 header chain could potentially impact core routers in the internet. So at that time, it was decided to encapsulate any

packet with a RPL Option using IPv6-in-IPv6 in all cases where it was unclear whether the packet would remain within the RPL domain. In the exception case where a packet would still leak, the Option Type would ensure that the first router in the Internet that does not recognize the option would drop the packet and protect the rest of the network.

Even with [RFC8138], where the IPv6-in-IPv6 header is compressed, this approach yields extra bytes in a packet; this means consuming more energy, more bandwidth, incurring higher chances of loss and possibly causing a fragmentation at the 6LoWPAN level. This impacts the daily operation of constrained devices for a case that generally does not happen and would not heavily impact the core anyway.

While intention was and remains that the Hop-by-Hop header with a RPL Option should be confined within the RPL domain, this specification modifies this behavior in order to reduce the dependency on IPv6-in-IPv6 and protect the constrained devices. Section 4 of [RFC8200] clarifies the behaviour of routers in the Internet as follows: "it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so".

When unclear about the travel of a packet, it becomes preferable for a source not to encapsulate, accepting the fact that the packet may leave the RPL domain on its way to its destination. In that event, the packet should reach its destination and should not be discarded by the first node that does not recognize the RPL Option. But with the current value of the Option Type, if a node in the Internet is configured to process the Hop-by-Hop header, and if such node encounters an option with the first two bits set to 01 and conforms to [RFC8200], it will drop the packet. Host systems should do the same, irrespective of the configuration.

Thus, this document updates the Option Type of the RPL Option [RFC6553], naming it RPI Option Type for simplicity, to (Figure 4): the two high order bits MUST be set to '00' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node MUST skip over this option and continue processing the header ([RFC8200] Section 4.2) if it doesn't recognize the Option Type, and the third bit continues to be set to indicate that the Option Data may change en route. The rightmost five bits remain at 0x3(00011). This ensures that a packet that leaves the RPL domain of an LLN (or that leaves the LLN entirely) will not be discarded when it contains the RPL Option.

With the new Option Type, if an IPv6 (intermediate) node (RPL-not-capable) receives a packet with a RPL Option, it should ignore the

Hop-by-Hop RPL Option (skip over this option and continue processing the header). This is relevant, as it was mentioned previously, in the case that there is a flow from RAL to Internet (see Section 7.2.1).

This is a significant update to [RFC6553].

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x23	00	1	00011	RPL Option	[RFCXXXX] (*)

Figure 4: Revised Option Type in RPL Option. (*) represents this document

Without the signaling described below, this change would otherwise create a lack of interoperation (flag day) for existing networks which are currently using 0x63 as the RPI Option Type value. A move to 0x23 will not be understood by those networks. It is suggested that RPL implementations accept both 0x63 and 0x23 when processing the header.

When forwarding packets, implementations SHOULD use the same value of RPI Type as was received. This is required because the RPI Option Type does not change en route ([RFC8200] - Section 4.2). It allows the network to be incrementally upgraded and allows the DODAG root to know which parts of the network have been upgraded.

When originating new packets, implementations should have an option to determine which value to originate with, this option is controlled by the DIO Configuration option (Section Section 4.1.3).

The change of RPI Option Type from 0x63 to 0x23, makes all [RFC8200] Section 4.2 compliant nodes tolerant of the RPL artifacts. There is no longer a need to remove the artifacts when sending traffic to the Internet. This change clarifies when to use IPv6-in-IPv6 headers, and how to address them: The Hop-by-Hop Options header containing the RPI MUST always be added when 6LRs originate packets (without IPv6-in-IPv6 headers), and IPv6-in-IPv6 headers MUST always be added when a 6LR finds that it needs to insert a Hop-by-Hop Options header containing the RPL Option. The IPv6-in-IPv6 header is to be addressed to the RPL root when on the way up, and to the end-host when on the way down.

In the non-storing case, dealing with not-RPL aware leaf nodes is much easier as the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node.

The 6LBR can recognize not-RPL aware leaf nodes because it will receive a DAO about that node from the 6LR immediately above that not-RPL aware node.

The non-storing mode case does not require the type change from 0x63 to 0x23, as the root can always create the right packet. The type change does not adversely affect the non-storing case. (see Section 4.1.3)

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

This modification is required in order to be able to decompress the RPL Option with the new Option Type of 0x23.

RPI-6LoRH header provides a compressed form for the RPL RPI; see [RFC8138], Section 6. A node that is decompressing this header MUST decompress using the RPI Option Type that is currently active: that is, a choice between 0x23 (new) and 0x63 (old). The node will know which to use based upon the presence of the flag in the DODAG Configuration option defined in Section 4.1.3. E.g. If the network is in 0x23 mode (by DIO option), then it should be decompressed to 0x23.

[RFC8138] section 7 documents how to compress the IPv6-in-IPv6 header.

There are potential significant advantages to having a single code path that always processes IPv6-in-IPv6 headers with no conditional branches.

In Storing Mode, the scenarios where the flow goes from RAL to RUL and RUL to RUL include compression of the IPv6-in-IPv6 and RPI headers. The use of the IPv6-in-IPv6 header is MANDATORY in this case, and it SHOULD be compressed with [RFC8138] section 7. Figure 5 illustrates the case in Storing mode where the packet is received from the Internet, then the root encapsulates the packet to insert the RPI. In that example, the leaf is not known to support RFC 8138, and the packet is encapsulated to the 6LR that is the parent and last hop to the final destination.

```

+-+ ... +-+ ... +-+ ... +-+ +-+ +-+ +-+ +-+ +-+ ... +-+ +-+ ... -+++ ... +-...
|11110001|SRH-6LoRH| RPI- |IP-in-IP| NH=1 |11110001| UDP | UDP
|Page 1 |Type1 S=0| 6LoRH |6LoRH | LOWPAN_IPHC| UDP | hdr | Payld
+-+ ... +-+ ... +-+ ... +-+ +-+ +-+ +-+ +-+ +-+ ... +-+ +-+ ... -+ ... +-...
                <-4bytes->                <- RFC 6282 ->
                                           No RPL artifact

```

Figure 5: RPI Inserted by the Root in Storing Mode

In Figure 5, the source of the IPv6-in-IPv6 encapsulation is the Root, so it is elided in the IP-in-IP 6LoRH. The destination is the parent 6LR of the destination of the inner packet so it cannot be elided. It is placed as the single entry in an SRH-6LoRH as the first 6LoRH. There is a single entry so the SRH-6LoRH Size is 0. In that example, the type is 1 so the 6LR address is compressed to 2 bytes. It results that the total length of the SRH-6LoRH is 4 bytes. Follows the RPI-6LoRH and then the IP-in-IP 6LoRH. When the IP-in-IP 6LoRH is removed, all the router headers that precede it are also removed. The Paging Dispatch [RFC8025] may also be removed if there was no previous Page change to a Page other than 0 or 1, since the LOWPAN_IPHC is encoded in the same fashion in the default Page 0 and in Page 1. The resulting packet to the destination is the inner packet compressed with [RFC6282].

5. Sample/reference topology

A RPL network in general is composed of a 6LBR, a Backbone Router (6BBR), a 6LR and a 6LN as a leaf logically organized in a DODAG structure.

Figure 6 shows the reference RPL Topology for this document. The letters above the nodes are there so that they may be referenced in subsequent sections. In the figure, 6LR represents a full router node. The 6LN is a RPL aware router, or host (as a leaf). Additionally, for simplification purposes, it is supposed that the 6LBR has direct access to Internet and is the root of the DODAG, thus the 6BBR is not present in the figure.

The 6LN leaves (RAL) marked as (F, H and I) are RPL nodes with no children hosts.

The leaves marked as RUL (G and J) are devices that do not speak RPL at all (not-RPL-aware), but use Router-Advertisements, 6LowPAN DAR/DAC and 6LoWPAN ND only to participate in the network [RFC8505]. In the document these leaves (G and J) are also referred to as a RUL.

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

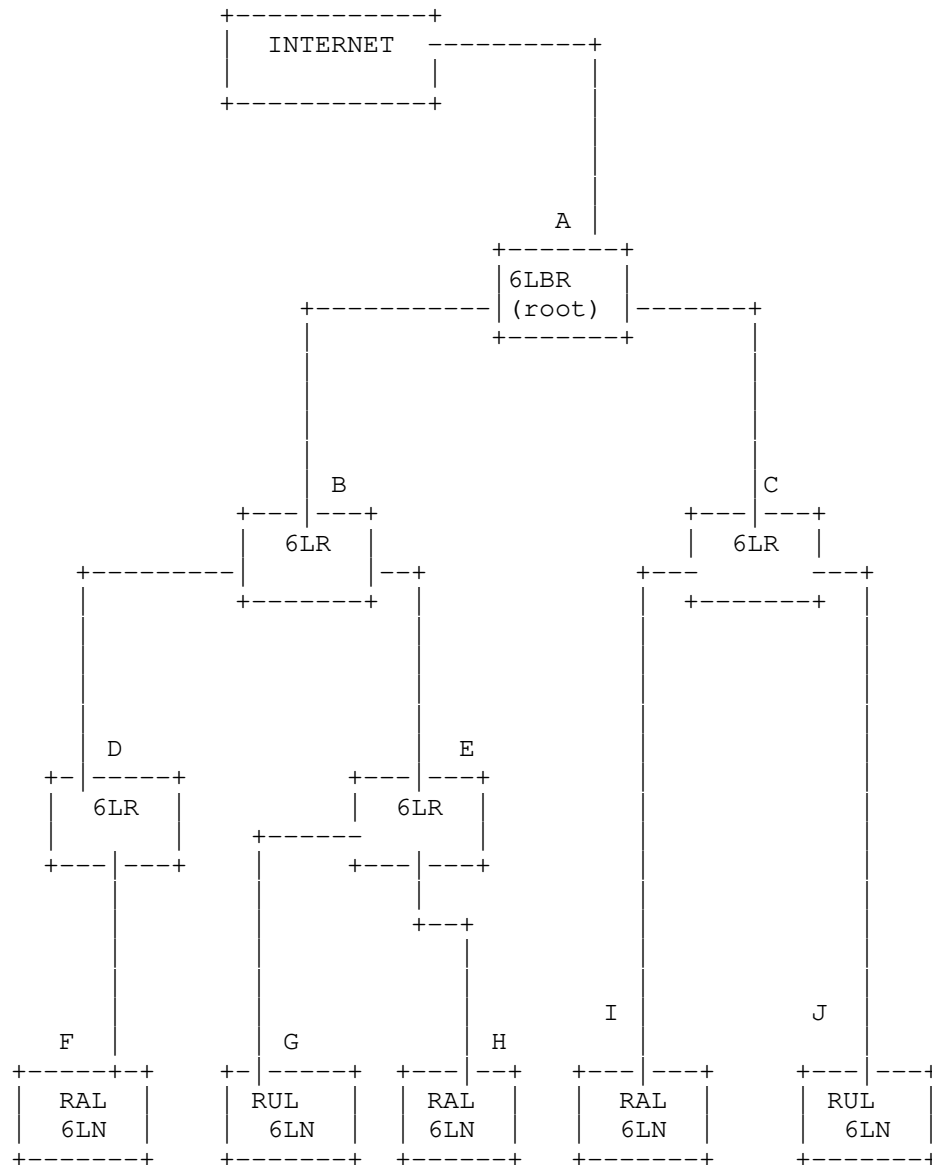


Figure 6: A reference RPL Topology.

6. Use cases

In the data plane a combination of RFC6553, RFC6554 and IPv6-in-IPv6 encapsulation are going to be analyzed for a number of representative traffic flows.

The use cases describe the communication in the following cases: - Between RPL-aware-nodes with the root (6LBR) - Between RPL-aware-nodes with the Internet - Between RUL nodes within the LLN (e.g. see Section 7.1.4) - Inside of the LLN when the final destination address resides outside of the LLN (e.g. see Section 7.2.3).

The use cases are as follows:

Interaction between Leaf and Root:

RAL to root

root to RAL

RUL to root

root to RUL

Interaction between Leaf and Internet:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

Interaction between leaves:

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

This document is consistent with the rule that a Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed. This is a fundamental precept of the IPv6 architecture as outlined in [RFC8200].

As the rank information in the RPI artifact is changed at each hop, it will typically be zero when it arrives at the DODAG root. The DODAG root MUST force it to zero when passing the packet out to the Internet. The Internet will therefore not see any SenderRank information.

Despite being legal to leave the RPI artifact in place, an intermediate router that needs to add an extension header (e.g. RH3 or RPL Option) MUST still encapsulate the packet in an (additional) outer IP header. The new header is placed after this new outer IP header.

A corollary is that an intermediate router can remove an RH3 or RPL Option only if it is placed in an encapsulating IPv6 Header that is addressed TO this intermediate router. When doing the above, the whole encapsulating header must be removed. (A replacement may be added). This sometimes can result in outer IP headers being addressed to the next hop router using link-local address.

Both the RPL Option and the RH3 headers may be modified in very specific ways by routers on the path of the packet without the need to add and remove an encapsulating header. Both headers were designed with this modification in mind, and both the RPL RH3 and the RPL Option are marked mutable but recoverable: so an IPsec AH security header can be applied across these headers, but it can not secure the values which mutate.

The RPI MUST be present in every single RPL data packet.

Prior to [RFC8138], there was significant interest in creating an exception to this rule and removing the RPI for downward flows in non-storing mode. This exception covered a very small number of cases, and caused significant interoperability challenges while adding significant interest in the code and tests. The ability to compress the RPI down to three bytes or less removes much of the pressure to optimize this any further [I-D.ietf-anima-autonomic-control-plane].

Throughout the following subsections, the examples are described in more details in the first subsections, and more concisely in the later ones.

The uses cases are delineated based on the following IPV6 and RPL mandates:

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

- Because of the above requirement, packets from the Internet have to be encapsulated.
- A Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed.
- Extension headers may not be added or removed except by the sender or the receiver.
- RPI and RH3 headers may be modified by routers on the path of the packet without the need to add and remove an encapsulating header.
- an RH3 or RPL Option can only be removed by an intermediate router if it is placed in an encapsulating IPv6 Header, which is addressed to the intermediate router.
- Non-storing mode requires downstream encapsulation by root for RH3.

The uses cases are delineated based on the following assumptions:

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

- Each IPv6 node (including Internet routers) obeys [RFC8200], so that 0x23 RPI Option Type can be safely inserted.
- All 6LRs obey [RFC8200].
- The RPI is ignored at the IPv6 dst node (RUL).
- In the uses cases, we assume that the RAL supports IP-in-IP encapsulation.
- In the uses cases, we don't assume that the RUL supports IP-in-IP encapsulation.
- For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags.
- The description for RALs applies to RAN in general.
- Non-constrained uses of RPL are not in scope of this document.
- Compression is based on [RFC8138].

- The flow label [RFC6437] is not needed in RPL.

7. Storing mode

In storing mode (SM) (fully stateful), the sender can determine if the destination is inside the LLN by looking if the destination address is matched by the DIO's Prefix Information Option (PIO) option.

The following table (Figure 7) itemizes which headers are needed in each of the following scenarios. It indicates whether an IPv6-in-IPv6 header must be added and what destination it must be addressed to: (1) the final destination (the RAL node that is the target (tgt)), (2) the "root", or (3) the 6LR parent of a RUL.

In cases where no IPv6-in-IPv6 header is needed, the column states "No", and the destination is N/A (Not Applicable). If the IPv6-in-IPv6 header is needed, the column shows "must".

In all cases, the RPI is needed, since it identifies inconsistencies (loops) in the routing topology. In general, the RH3 is not needed because it is not used in storing mode. However, there is one scenario (from the root to the RUL in SM) where the RH3 can be used to point at the RUL (Figure 11).

The leaf can be a router 6LR or a host, both indicated as 6LN. The root refers to the 6LBR (see Figure 6).

Interaction between	Use Case	IPv6-in-IPv6	IPv6-in-IPv6 dst
Leaf - Root	RAL to root	No	N/A
	root to RAL	No	N/A
	root to RUL	must	6LR
	RUL to root	must	root
Leaf - Internet	RAL to Int	may	root
	Int to RAL	must	RAL (tgt)
	RUL to Int	must	root
	Int to RUL	must	6LR
Leaf - Leaf	RAL to RAL	No	N/A
	RAL to RUL	No (up)	N/A
		must (down)	6LR
	RUL to RAL	must (up)	root
		must (down)	RAL
	RUL to RUL	must (up)	root
		must (down)	6LR

Figure 7: Table of IPv6-in-IPv6 encapsulation in Storing mode.

7.1. Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in storing mode (SM) between,

RAL to root

root to RAL

RUL to root

root to RUL

7.1.1. SM: Example of Flow from RAL to Root

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

In this case the flow comprises:

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

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

The RAL (Node F) inserts the RPI, and sends the packet to 6LR (Node D) which decrements the rank in the RPI and sends the packet up. When the packet arrives at 6LBR (Node A), the RPI is removed and the packet is processed.

No IPv6-in-IPv6 header is required.

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

The Figure 8 summarizes what headers are needed for this use case.

Header	RAL src	6LR_i	6LBR dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 8: SM: Summary of the use of headers from RAL to root

7.1.2. SM: Example of Flow from Root to RAL

In this case the flow comprises:

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

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

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

No IPv6-in-IPv6 header is required.

The Figure 9 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	RAL dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 9: SM: Summary of the use of headers from root to RAL

7.1.3. SM: Example of Flow from Root to RUL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

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

6LR_i (Node B) represents the intermediate routers from the source (6LBR) to the destination (RUL), $1 \leq i \leq n$, where n is the total

number of routers (6LR) that the packet goes through from the 6LBR (Node A) to the RUL (Node G).

The 6LBR will encapsulate the packet in an IPv6-in-IPv6 header, and prepend an RPI. The IPv6-in-IPv6 header is addressed to the 6LR parent of the RUL (6LR_n). The 6LR parent of the RUL removes the header and sends the packet to the RUL.

The Figure 10 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	6LR_n	RUL dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 10: SM: Summary of the use of headers from root to RUL

IP-in-IP encapsulation may be avoided for Root to RUL communication. In SM, it can be replaced by a loose RH3 header that indicates the RUL, in which case the packet is routed to the 6LR as a normal SM operation, then the 6LR forwards to the RUL based on the RH3, and the RUL ignores both the consumed RH3 and the RPI, as in Non-Storing Mode.

The Figure 11 summarizes what headers are needed for this scenario.

Header	6LBR src	6LR _i <i>i</i> =(1,...,n-1)	6LR _n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI	RPI RH3 (consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	RH3	--	RPI, RH3 (both ignored)

Figure 11: SM: Summary of the use of headers from root to RUL without encapsulation

7.1.4. SM: Example of Flow from RUL to Root

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR)

For example, a communication flow could be: Node G (RUL) --> Node E (6LR₁) --> Node B (6LR_i) --> Node A root (6LBR)

6LR_i represents the intermediate routers from the source (RUL) to the destination (6LBR), $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RUL to the 6LBR.

When the packet arrives from the RUL (Node G) to 6LR₁ (Node E), the 6LR₁ will encapsulate the packet in an IPv6-in-IPv6 header with an RPI. The IPv6-in-IPv6 header is addressed to the root (Node A). The root removes the header and processes the packet.

The Figure 12 shows the table that summarizes what headers are needed for this use case where the IPv6-in-IPv6 header is addressed to the root (Node A).

Header	RUL src node	6LR_1	6LR_i	6LBR dst
Added headers	--	IP6-IP6 RPI	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	---	IP6-IP6 RPI
Untouched headers	--	--	IP6-IP6	--

Figure 12: SM: Summary of the use of headers from RUL to root.

7.2. SM: Interaction between Leaf and Internet.

In this section is described the communication flow in storing mode (SM) between,

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

7.2.1. SM: Example of Flow from RAL to Internet

In this case the flow comprises:

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

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

6LR_i represents the intermediate routers from the source (RAL) to the root (6LBR), $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RAL to the 6LBR.

RPL information from RFC 6553 may go out to Internet as it will be ignored by nodes which have not been configured to be RPI aware. No IPv6-in-IPv6 header is required.

On the other hand, the RAL may insert the RPI encapsulated in a IPv6-in-IPv6 header to the root. Thus, the root removes the RPI and send the packet to the Internet.

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

The Figure 13 summarizes what headers are needed for this use case when there is no encapsulation. Note that the RPI is modified by 6LBR to set the SenderRank to zero in case that it is not already zero. The Figure 14 summarizes what headers are needed when encapsulation to the root takes place.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

Figure 13: SM: Summary of the use of headers from RAL to Internet with no encapsulation

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 14: SM: Summary of the use of headers from RAL to Internet with encapsulation to the root (6LBR).

7.2.2. SM: Example of Flow from Internet to RAL

In this case the flow comprises:

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

For example, a communication flow could be: Internet --> Node A
root(6LBR) --> Node B (6LR_1) --> Node D (6LR_n) --> Node F (RAL)

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

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

Header	Internet src	6LBR	6LR_i	RAL dst
Added headers	--	IP6-IP6 (RPI)	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	--	IP6-IP6 (RPI)
Untouched headers	--	--	--	--

Figure 15: SM: Summary of the use of headers from Internet to RAL.

7.2.3. SM: Example of Flow from RUL to Internet

In this case the flow comprises:

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

For example, a communication flow could be: Node G (RUL) --> Node E (6LR_1) --> Node B (6LR_i) --> Node A root (6LBR) --> Internet

The node 6LR_1 (i=1) will add an IPv6-in-IPv6(RPI) header addressed to the root such that the root can remove the RPI before passing upwards. In the intermediate 6LR, the rank in the RPI is modified.

The originating node will ideally leave the IPv6 flow label as zero so that the packet can be better compressed through the LLN. The 6LBR will set the flow label of the packet to a non-zero value when sending to the Internet, for details check [RFC6437].

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

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

Figure 16: SM: Summary of the use of headers from RUL to Internet.

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

In this case the flow comprises:

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

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

The 6LBR will have to add an RPI within an IPv6-in-IPv6 header. The IPv6-in-IPv6 is addressed to the 6LR parent of the RUL.

Further details about this are mentioned in [I-D.ietf-roll-unaware-leaves], which specifies RPL routing for a 6LN acting as a plain host and not being aware of RPL.

The 6LBR may set the flow label on the inner IPv6-in-IPv6 header to zero in order to aid in compression [RFC8138][RFC6437].

The Figure 17 shows the table that summarizes what headers are needed for this use case.

Header	Inter- net src	6LBR	6LR_i [i=1,...,n-1]	6LR_n	RUL dst
Inserted headers	--	IP6-IP6 (RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6 (RPI)	--
Untouched headers	--	--	--	--	--

Figure 17: SM: Summary of the use of headers from Internet to RUL.

7.3. SM: Interaction between Leaf and Leaf

In this section is described the communication flow in storing mode (SM) between,

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

7.3.1. SM: Example of Flow from RAL to RAL

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

When the nodes are not directly connected, then in storing mode, the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> RAL dst (6LN)

For example, a communication flow could be: Node F (RAL src)--> Node D (6LR_ia)--> Node B (6LR_x) --> Node E (6LR_id) --> Node H (RAL dst)

6LR_ia (Node D) represents the intermediate routers from source to the common parent (6LR_x) (Node B), $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL (Node F) to the common parent 6LR_x (Node B).

6LR_id (Node E) represents the intermediate routers from the common parent (6LR_x) (Node B) to destination RAL (Node H), $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination RAL (Node H).

It is assumed that the two nodes are in the same RPL domain (that they share the same DODAG root). At the common parent (Node B), the direction flag ('O' flag) of the RPI is changed (from decreasing ranks to increasing ranks).

While the 6LR nodes will update the RPI, no node needs to add or remove the RPI, so no IPv6-in-IPv6 headers are necessary.

The Figure 18 summarizes what headers are needed for this use case.

Header	RAL src	6LR_ia	6LR_x (common parent)	6LR_id	RAL dst
Added headers	RPI	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Removed headers	--	--	--	--	RPI
Untouched headers	--	--	--	--	--

Figure 18: SM: Summary of the Use of Headers from RAL to RAL

7.3.2. SM: Example of Flow from RAL to RUL

In this case the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LBR - The root-) -->
6LR_id --> RUL (IPv6 dst node)

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

6LR_ia represents the intermediate routers from source (RAL) to the common parent (the Root), $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL to the Root.

6LR_id (Node E) represents the intermediate routers from the Root (Node B) to destination RUL (Node G). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the Root down to the destination RUL.

In this case, the packet from the RAL goes to 6LBR because the route to the RUL is not injected into the RPL-SM. Thus, the RAL inserts an RPI (RPI1) addressed to the root (6LBR). The root does not remove the RPI1 (the root cannot remove an RPI if there is no encapsulation). The root inserts an IPv6-IPv6 encapsulation with an RPI2 and sends it to the 6LR parent of the RUL, which removes the encapsulation and RPI2 before passing the packet to the RUL.

The Figure 19 summarizes what headers are needed for this use case.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	RPI1	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 19: SM: Summary of the Use of Headers from RAL to RUL

7.3.3. SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_ia --> 6LBR --> 6LR_id --> RAL dst (6LN)

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

6LR_ia (Node E) represents the intermediate routers from source (RUL) (Node G) to the root (Node A). In this case, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from source to the root.

6LR_id represents the intermediate routers from the root (Node A) to destination RAL (Node F). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the root to the destination RAL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI1) encapsulated in a IPv6-in-IPv6 header to the root. The root removes the outer header including the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the destination RAL (Node F).

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

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--	--
Modified headers	--	--	RPI1	--	RPI2	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)
Untouched headers	--	--	--	--	--	--

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

7.3.4. SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> 6LBR --> 6LR_id --> RUL
(IPv6 dst node)

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

Internal nodes 6LR_ia (e.g: Node E or Node B) is the intermediate router from the RUL source (Node G) to the root (6LBR) (Node A). In this case, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from the RUL to the root. 6LR_1 refers when $ia=1$.

6LR_id (Node C) represents the intermediate routers from the root (Node A) to the destination RUL dst node (Node J). In this case, $1 \leq id \leq m$, where m is the total number of routers (6LR) that the packet goes through from the root to destination RUL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI), encapsulated in an IPv6-in-IPv6 header directed to the root. The root removes the outer header including

the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the 6LR father of the RUL.

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

Header	RUL src	6LR_1	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst
Added Headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 21: SM: Summary of the use of headers from RUL to RUL

8. Non Storing mode

In Non Storing Mode (Non-SM) (fully source routed), the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node. Thus, there is no need for all nodes to know about the existence of RPL-unaware nodes. Only the 6LBR needs to act if compensation is necessary for not-RPL aware receivers.

The table (Figure 22) summarizes what headers are needed in the following scenarios, and indicates when the RPI, RH3 and IPv6-in-IPv6 header are to be inserted. The last column depicts the target destination of the IPv6-in-IPv6 header: 6LN (indicated by "RAL"), 6LR (parent of a RUL) or the root. In cases where no IPv6-in-IPv6 header is needed, the column indicates "No". There is no expectation on RPL that RPI can be omitted, because it is needed for routing, quality of service and compression. This specification expects that an RPI is always present. The term "may(up)" means that the IPv6-in-IPv6 header may be necessary in the upwards direction. The term "must(up)" means that the IPv6-in-IPv6 header must be present in the upwards direction. The term "must(down)" means that the IPv6-in-IPv6 header must be present in the downward direction.

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 6). In the table (Figure 22) the (1) indicates a 6tisch case [RFC8180], where the RPI may still be needed for the RPLInstanceID to be available for priority/channel selection at each hop.

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

Figure 22: Table that shows headers needed in Non-Storing mode: RPI, RH3, IPv6-in-IPv6 encapsulation.

8.1. Non-Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RAL to root

root to RAL

RUL to root

root to RUL

8.1.1. Non-SM: Example of Flow from RAL to root

In non-storing mode the leaf node uses default routing to send traffic to the root. The RPI must be included since it contains the rank information, which is used to avoid/detect loops.

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

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

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RAL) to destination (6LBR).

This situation is the same case as storing mode.

The Figure 23 summarizes what headers are needed for this use case.

Header	RAL src	6LR_i	6LBR dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 23: Non-SM: Summary of the use of headers from RAL to root

8.1.2. Non-SM: Example of Flow from root to RAL

In this case the flow comprises:

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

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

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (6LBR) to destination (RAL).

The 6LBR inserts an RH3, and an RPI. No IPv6-in-IPv6 header is necessary as the traffic originates with a RPL aware node, the 6LBR. The destination is known to be RPL-aware because the root knows the whole topology in non-storing mode.

The Figure 24 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	RAL dst
Added headers	RPI, RH3	--	--
Modified headers	--	RPI, RH3	--
Removed headers	--	--	RPI, RH3
Untouched headers	--	--	--

Figure 24: Non-SM: Summary of the use of headers from root to RAL

8.1.3. Non-SM: Example of Flow from root to RUL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

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

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (6LBR) to destination (RUL).

In the 6LBR, the RH3 is added; it is then modified at each intermediate 6LR (6LR_1 and so on), and it is fully consumed in the last 6LR (6LR_n) but is left in place. When the RPI is added, the RUL, which does not understand the RPI, will ignore it (per [RFC8200]); thus, encapsulation is not necessary.

The Figure 25 depicts the table that summarizes what headers are needed for this use case.

Header	6LBR src	6LR _i <i>i</i> =(1,...,n-1)	6LR _n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI, RH3	RPI, RH3 (consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI, RH3 (both ignored)

Figure 25: Non-SM: Summary of the use of headers from root to RUL

8.1.4. Non-SM: Example of Flow from RUL to root

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR) dst

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

6LR_i represents the intermediate routers from source to destination. In this case, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RUL) to destination (6LBR). For example, 6LR₁ ($i=1$) is the router that receives the packets from the RUL.

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

The Figure 26 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_i	6LBR dst
Added headers	--	IPv6-in-IPv6 (RPI)	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	--	IPv6-in-IPv6 (RPI)
Untouched headers	--	--	--	--

Figure 26: Non-SM: Summary of the use of headers from RUL to root

8.2. Non-Storing Mode: Interaction between Leaf and Internet

This section will describe the communication flow in Non Storing Mode (Non-SM) between:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

8.2.1. Non-SM: Example of Flow from RAL to Internet

In this case the flow comprises:

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

For example, a communication flow could be: Node F (RAL) --> Node D --> Node B --> Node A --> Internet. Having the RAL information about the RPL domain, the packet may be encapsulated to the root when the destination is not in the RPL domain of the RAL.

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from source (RAL) to 6LBR.

In this case, the encapsulation from the RAL to the root is optional. The simplest case is when the RPI gets to the Internet (as the Figure 27 shows it), knowing that the Internet is going to ignore it.

The IPv6 flow label should be set to zero to aid in compression [RFC8138], and the 6LBR will set it to a non-zero value when sending towards the Internet [RFC6437].

The Figure 27 summarizes what headers are needed for this use case when no encapsulation is used. The Figure 28 summarizes what headers are needed for this use case when encapsulation to the root is used.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

Figure 27: Non-SM: Summary of the use of headers from RAL to Internet with no encapsulation

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	IPv6-in-IPv6 (RPI)	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI)	--
Untouched headers	--	--	--	--

Figure 28: Non-SM: Summary of the use of headers from RAL to Internet with encapsulation to the root

8.2.2. Non-SM: Example of Flow from Internet to RAL

In this case the flow comprises:

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

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

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from 6LBR to destination (RAL).

The 6LBR must add an RH3 header. As the 6LBR will know the path and address of the target node, it can address the IPv6-in-IPv6 header to that node. The 6LBR will zero the flow label upon entry in order to aid compression [RFC8138].

The Figure 29 summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR _i	RAL dst
Added headers	--	IPv6-in-IPv6 (RH3, RPI)	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3, RPI)	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3, RPI)
Untouched headers	--	--	--	--

Figure 29: Non-SM: Summary of the use of headers from Internet to RAL

8.2.3. Non-SM: Example of Flow from RUL to Internet

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR₁ --> 6LR_i --> root (6LBR) --> Internet
dst

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

6LR_i represents the intermediate routers from source to destination,
1 ≤ i ≤ n, where n is the total number of routers (6LRs) that the
packet goes through from the source (RUL) to the 6LBR, e.g., 6LR₁
(i=1).

In this case the flow label is recommended to be zero in the RUL. As
the RUL parent adds RPL headers in the RUL packet, the first 6LR
(6LR₁) will add an RPI inside a new IPv6-in-IPv6 header. The IPv6-
in-IPv6 header will be addressed to the root. This case is identical
to the storing-mode case (see Section 7.2.3).

The Figure 30 shows the table that summarizes what headers are needed
for this use case.

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

Figure 30: Non-SM: Summary of the use of headers from RUL to Internet

8.2.4. Non-SM: Example of Flow from Internet to RUL

In this case the flow comprises:

Internet src --> root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

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

6LR_i represents the intermediate routers from source to destination, $1 \leq i \leq n$, where n is the total number of routers (6LR) that the packet goes through from 6LBR to RUL.

The 6LBR must add an RH3 header inside an IPv6-in-IPv6 header. The 6LBR will know the path, and will recognize that the final node is not a RPL capable node as it will have received the connectivity DAO from the nearest 6LR. The 6LBR can therefore make the IPv6-in-IPv6 header destination be the last 6LR. The 6LBR will set to zero the flow label upon entry in order to aid compression [RFC8138].

The Figure 31 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	6LR_n	RUL dst
Added headers	--	IP6-IP6 (RH3,RPI)	--	--	--
Modified headers	--	--	IP6-IP6 (RH3,RPI)	--	--
Removed headers	--	--	--	IP6-IP6 (RH3,RPI)	--
Untouched headers	--	--	--	--	--

Figure 31: Non-SM: Summary of the use of headers from Internet to RUL.

8.3. Non-SM: Interaction between leaves

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

8.3.1. Non-SM: Example of Flow from RAL to RAL

In this case the flow comprises:

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

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

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of routers (6LR) that the packet goes through from RAL to the root.

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

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

The originating node may put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root, so that the 6LBR can remove that header. If it does not, then the RPI1 is forwarded down from the root in the inner header to no avail.

The 6LBR will need to insert an RH3 header, which requires that it add an IPv6-in-IPv6 header. It removes the RPI(RPI1), as it was contained in an IPv6-in-IPv6 header addressed to it. Otherwise, there may be an RPI buried inside the inner IP header, which should get ignored. The root inserts an RPI (RPI2) alongside the RH3.

Networks that use the RPL P2P extension [RFC6997] are essentially non-storing DODAGs and fall into this scenario or scenario Section 8.1.2, with the originating node acting as 6LBR.

The Figure 32 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place.

The Figure 33 shows the table that summarizes what headers are needed for this use case when there is no encapsulation to the root. Note that in the Modified headers row, going up in each 6LR_id only the RPI1 is changed. Going down, in each 6LR_id the IPv6 header is swapped with the RH3 so both are changed alongside with the RPI2.

Header	RAL src	6LR_ia	6LBR	6LR_id	RAL dst
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3-> RAL, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3,RPI2)	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--

Figure 32: Non-SM: Summary of the Use of Headers from RAL to RAL with encapsulation to the root.

Header	RAL	6LR_ia	6LBR	6LR_id	RAL
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	RPI1	RPI1	RPI1 (Ignored)

Figure 33: Non-SM: Summary of the Use of Headers from RAL to RAL without encapsulation to the root.

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

In this case the flow comprises:

RAL --> 6LR_ia --> root (6LBR) --> 6LR_id --> RUL (IPv6 dst node)

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

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LRs).

As in the previous case, the RAL (6LN) may insert an RPI (RPI1) header which must be in an IPv6-in-IPv6 header addressed to the root so that the 6LBR can remove this RPI. The 6LBR will then insert an RH3 inside a new IPv6-in-IPv6 header addressed to the last 6LR_id ($6LR_id = m$) alongside the insertion of RPI2.

If the originating node does not put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root. Then, the RPI1 is forwarded down from the root in the inner header to no avail.

The Figure 34 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place. The Figure 35 shows the table that summarizes what headers are needed for this use case when no encapsulation to the root takes place.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--

Figure 34: Non-SM: Summary of the use of headers from RAL to RUL with encapsulation to the root.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst node
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 35: Non-SM: Summary of the use of headers from RAL to RUL without encapsulation to the root.

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

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id
--> RAL dst (6LN)

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

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

In this scenario the RPI (RPI1) is added by the first 6LR (6LR_1) inside an IPv6-in-IPv6 header addressed to the root. The 6LBR will remove this RPI, and add its own IPv6-in-IPv6 header containing an RH3 header and an RPI (RPI2).

The Figure 36 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--	--

Figure 36: Non-SM: Summary of the use of headers from RUL to RAL.

8.3.4. Non-SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id
--> RUL (IPv6 dst node)

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

6LR_ia represents the intermediate routers from source to the root, $1 \leq ia \leq n$, where n is the total number of intermediate routers (6LR)

6LR_id represents the intermediate routers from the root to the destination, $1 \leq id \leq m$, where m is the total number of the intermediate routers (6LR).

This scenario is the combination of the previous two cases.

The Figure 37 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 37: Non-SM: Summary of the use of headers from RUL to RUL

9. Operational Considerations of supporting RUL-leaves

Roughly half of the situations described in this document involve leaf ("host") nodes that do not speak RPL. These nodes fall into two further categories: ones that drop a packet that have RPI or RH3 headers, and ones that continue to process a packet that has RPI and/or RH3 headers.

[RFC8200] provides for new rules that suggest that nodes that have not been configured (explicitly) to examine Hop-by-Hop headers, should ignore those headers, and continue processing the packet. Despite this, and despite the switch from 0x63 to 0x23, there may be nodes that are pre-RFC8200, or simply intolerant. Those nodes will drop packets that continue to have RPL artifacts in them. In general, such nodes can not be easily supported in RPL LLNs.

There are some specific cases where it is possible to remove the RPL artifacts prior to forwarding the packet to the leaf host. The critical thing is that the artifacts have been inserted by the RPL root inside an IPv6-in-IPv6 header, and that the header has been addressed to the 6LR immediately prior to the leaf node. In that case, in the process of removing the IPv6-in-IPv6 header, the artifacts can also be removed.

The above case occurs whenever traffic originates from the outside the LLN (the "Internet" cases above), and non-storing mode is used. In non-storing mode, the RPL root knows the exact topology (as it must create the RH3 header) and therefore knows which 6LR is prior to the leaf. For example, in Figure 6, Node E is the 6LR prior to leaf Node G, or Node C is the 6LR prior to leaf Node J.

Traffic originating from the RPL root (such as when the data collection system is co-located on the RPL root), does not require an IPv6-in-IPv6 header (in storing or non-storing mode), as the packet is originating at the root, and the root can insert the RPI and RH3 headers directly into the packet, as it is formed. Such a packet is slightly smaller, but only can be sent to nodes (whether RPL aware or not), that will tolerate the RPL artifacts.

An operator that finds itself with a high amount of traffic from the RPL root to RPL-not-aware-leaves, will have to do IPv6-in-IPv6 encapsulation if the leaf is not tolerant of the RPL artifacts. Such an operator could otherwise omit this unnecessary header if it was certain of the properties of the leaf.

As storing mode can not know the final path of the traffic, intolerant (that drop packets with RPL artifacts) leaf nodes can not be supported.

10. Operational considerations of introducing 0x23

This section describes the operational considerations of introducing the new RPI Option Type of 0x23.

During bootstrapping the node gets the DIO with the information of RPI Option Type, indicating the new RPI in the DODAG Configuration option Flag. The DODAG root is in charge to configure the current network to the new value, through DIO messages and when all the nodes are set with the new value. The DODAG should change to a new DODAG version. In case of rebooting, the node does not remember the RPI Option Type. Thus, the DIO is sent with a flag indicating the new RPI Option Type.

The DODAG Configuration option is contained in a RPL DIO message, which contains a unique DTSN counter. The leaf nodes respond to this message with DAO messages containing the same DTSN. This is a normal part of RPL routing; the RPL root therefore knows when the updated DODAG Configuration option has been seen by all nodes.

Before the migration happens, all the RPL-aware nodes should support both values . The migration procedure is triggered when the DIO is sent with the flag indicating the new RPI Option Type. Namely, it remains at 0x63 until it is sure that the network is capable of 0x23, then it abruptly changes to 0x23. The 0x23 RPI Option allows to send packets to not-RPL nodes. The not-RPL nodes should ignore the option and continue processing the packets.

As mentioned previously, indicating the new RPI in the DODAG Configuration option flag is a way to avoid the flag day (abrupt changeover) in a network using 0x63 as the RPI Option Type value. It is suggested that RPL implementations accept both 0x63 and 0x23 RPI Option type values when processing the header to enable interoperability.

11. IANA Considerations

11.1. Option Type in RPL Option

This document updates the registration made in [RFC6553] Destination Options and Hop-by-Hop Options registry from 0x63 to 0x23 as shown in Figure 38.

Hex Value	Binary Value			Description	Reference
	act	chg	rest		
0x23	00	1	00011	RPL Option	[RFCXXXX] (*)
0x63	01	1	00011	RPL Option (DEPRECATED)	[RFC6553] [RFCXXXX] (*)

Figure 38: Option Type in RPL Option. (*) represents this document
DODAG Configuration option is updated as follows (Figure 39):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 39: DODAG Configuration option Flag to indicate the RPI-flag-day.

11.2. Change to the DODAG Configuration Options Flags registry

This document requests IANA to change the name of the "DODAG Configuration Option Flags" registry to "DODAG Configuration Option Flags for MOP 0..6".

This document requests to be mentioned as a reference for this change.

11.3. Change MOP value 7 to Reserved

This document requests the changing the registration status of value 7 in the Mode of Operation registry from Unassigned to Reserved. This change is in support of future work.

This document requests to be mentioned as a reference for this entry in the registry.

12. Security Considerations

The security considerations covered in [RFC6553] and [RFC6554] apply when the packets are in the RPL Domain.

The IPv6-in-IPv6 mechanism described in this document is much more limited than the general mechanism described in [RFC2473]. The willingness of each node in the LLN to decapsulate packets and forward them could be exploited by nodes to disguise the origin of an attack.

While a typical LLN may be a very poor origin for attack traffic (as the networks tend to be very slow, and the nodes often have very low duty cycles), given enough nodes, LLNs could still have a significant impact, particularly if the attack is targeting another LLN. Additionally, some uses of RPL involve large backbone ISP scale equipment [I-D.ietf-anima-autonomic-control-plane], which may be equipped with multiple 100Gb/s interfaces.

Blocking or careful filtering of IPv6-in-IPv6 traffic entering the LLN as described above will make sure that any attack that is mounted must originate from compromised nodes within the LLN. The use of BCP38 [BCP38] filtering at the RPL root on egress traffic will both alert the operator to the existence of the attack, as well as drop the attack traffic. As the RPL network is typically numbered from a single prefix, which is itself assigned by RPL, BCP38 filtering involves a single prefix comparison and should be trivial to automatically configure.

There are some scenarios where IPv6-in-IPv6 traffic should be allowed to pass through the RPL root, such as the IPv6-in-IPv6 mediated communications between a new Pledge and the Join Registrar/Coordinator (JRC) when using [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-6tisch-dtsecurity-zerotouch-join]. This is the case for the RPL root to do careful filtering: it occurs only when the Join Coordinator is not co-located inside the RPL root.

With the above precautions, an attack using IPv6-in-IPv6 tunnels can only be by a node within the LLN on another node within the LLN. Such an attack could, of course, be done directly. An attack of this kind is meaningful only if the source addresses are either fake or if the point is to amplify return traffic. Such an attack, could also be done without the use of IPv6-in-IPv6 headers using forged source addresses. If the attack requires bi-directional communication, then IPv6-in-IPv6 provides no advantages.

Whenever IPv6-in-IPv6 headers are being proposed, there is a concern about creating security issues. In the Security Considerations

section of [RFC2473], it was suggested that tunnel entry and exit points can be secured by securing the IPv6 path between them. This recommendation is not practical for RPL networks. [RFC5406] goes into some detail on what additional details would be needed in order to "Use IPsec". Use of ESP would prevent [RFC8138] compression (compression must occur before encryption), and [RFC8138] compression is lossy in a way that prevents use of AH. These are minor issues. The major issue is how to establish trust enough such that IKEv2 could be used. This would require a system of certificates to be present in every single node, including any Internet nodes that might need to communicate with the LLN. Thus, using IPsec requires a global PKI in the general case.

More significantly, the use of IPsec tunnels to protect the IPv6-in-IPv6 headers would in the general case scale with the square of the number of nodes. This is a lot of resource for a constrained nodes on a constrained network. In the end, the IPsec tunnels would be providing only BCP38-like origin authentication! That is, IPsec provides a transitive guarantee to the tunnel exit point that the tunnel entry point did BCP38 on traffic going in. Just doing origin filtering per BCP 38 at the entry and exit of the LLN provides a similar level of security without all the scaling and trust problems related to IPv6 tunnels as discussed in RFC 2473. IPsec is not recommended.

An LLN with hostile nodes within it would not be protected against impersonation with the LLN by entry/exit filtering.

The RH3 header usage described here can be abused in equivalent ways. An external attacker may form a packet with an RH3 that is not fully consumed and encapsulate it to hide the RH3 from intermediate nodes and disguise the origin of traffic. As such, the attacker's RH3 header will not be seen by the network until it reaches the destination, which will decapsulate it. As indicated in section 4.2 of [RFC6554], RPL routers are responsible for ensuring that an SRH is only used between RPL routers. As such, if there is an RH3 that is not fully consumed in the encapsulated packet, the node that decapsulates it MUST ensure that the outer packet was originated in the RPL domain and drop the packet otherwise.

Also, as indicated by section 2 of [RFC6554], RPL Border Routers "do not allow datagrams carrying an SRH header to enter or exit a RPL routing domain". This sentence must be understood as concerning non-fully-consumed packets. A consumed (inert) RH3 header could be present in a packet that flows from one LLN, crosses the Internet, and enters another LLN. As per the discussion in this document, such headers do not need to be removed. However, there is no case described in this document where an RH3 is inserted in a non-storing

network on traffic that is leaving the LLN, but this document should not preclude such a future innovation.

In short, a packet that crosses the border of the RPL domain MAY carry and RH3, and if so, that RH3 MUST be fully consumed.

The RPI, if permitted to enter the LLN, could be used by an attacker to change the priority of a packet by selecting a different RPLInstanceID, perhaps one with a higher energy cost, for instance. It could also be that not all nodes are reachable in an LLN using the default RPLInstanceID, but a change of RPLInstanceID would permit an attacker to bypass such filtering. Like the RH3, an RPI is to be inserted by the RPL root on traffic entering the LLN by first inserting an IPv6-in-IPv6 header. The attacker's RPI therefore will not be seen by the network. Upon reaching the destination node the RPI has no further meaning and is just skipped; the presence of a second RPI will have no meaning to the end node as the packet has already been identified as being at it's final destination.

For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags. This is done in order to avoid: 1) The leaf is an external router that passes a packet that it did not generate and that carries an unrelated RPI and 2) The leaf is an attacker or presents misconfiguration and tries to inject traffic in a protected instance. Also, this applies in the case where the leaf is aware of the RPL instance and passes a correct RPI; the 6LR needs a configuration that allows that leaf to inject in that instance.

The RH3 and RPIs could be abused by an attacker inside of the network to route packets on non-obvious ways, perhaps eluding observation. This usage appears consistent with a normal operation of [RFC6997] and can not be restricted at all. This is a feature, not a bug.

[RFC7416] deals with many other threats to LLNs not directly related to the use of IPv6-in-IPv6 headers, and this document does not change that analysis.

Nodes within the LLN can use the IPv6-in-IPv6 mechanism to mount an attack on another part of the LLN, while disguising the origin of the attack. The mechanism can even be abused to make it appear that the attack is coming from outside the LLN, and unless countered, this could be used to mount a Distributed Denial Of Service attack upon nodes elsewhere in the Internet. See [DDOS-KREBS] for an example of such attacks already seen in the real world.

If an attack comes from inside of LLN, it can be alleviated with SAVI (Source Address Validation Improvement) using [RFC8505] with

[I-D.ietf-6lo-ap-nd]. The attacker will not be able to source traffic with an address that is not registered, and the registration process checks for topological correctness. Notice that there is an L2 authentication in most of the cases. If an attack comes from outside LLN IPv6-in- IPv6 can be used to hide inner routing headers, but by construction, the RH3 can typically only address nodes within the LLN. That is, an RH3 with a CmprI less than 8 , should be considered an attack (see RFC6554, section 3).

Nodes outside of the LLN will need to pass IPv6-in-IPv6 traffic through the RPL root to perform this attack. To counter, the RPL root SHOULD either restrict ingress of IPv6-in-IPv6 packets (the simpler solution), or it SHOULD walk the IP header extension chain until it can inspect the upper-layer-payload as described in [RFC7045]. In particular, the RPL root SHOULD do [BCP38] processing on the source addresses of all IP headers that it examines in both directions.

Note: there are some situations where a prefix will spread across multiple LLNs via mechanisms such as the one described in [I-D.ietf-6lo-backbone-router]. In this case the BCP38 filtering needs to take this into account, either by exchanging detailed routing information on each LLN, or by moving the BCP38 filtering further towards the Internet, so that the details of the multiple LLNs do not matter.

13. Acknowledgments

This work is done thanks to the grant given by the StandICT.eu project.

A special BIG thanks to C. M. Heard for the help with the Section 4. Much of the redaction in that section is based on his comments.

Additionally, the authors would like to acknowledge the review, feedback, and comments of (alphabetical order): Dominique Barthel, Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, Rahul Jadhav, Benjamin Kaduk, Matthias Kovatsch, Gustavo Mercado, Subramanian Moonesamy, Marcela Orbiscay, Charlie Perkins, Cristian Perez, Alvaro Retana, Peter van der Stok, Xavier Vilajosana, Eric Vyncke and Thomas Watteyne.

14. References

14.1. Normative References

- [BCP38] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/bcp38>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<https://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/info/rfc7045>>.

- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<https://www.rfc-editor.org/info/rfc8025>>.
- [RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <<https://www.rfc-editor.org/info/rfc8138>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

14.2. Informative References

- [DDOS-KREBS]
Goodin, D., "Record-breaking DDoS reportedly delivered by >145k hacked cameras", September 2016, <<http://arstechnica.com/security/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/>>.
- [I-D.ietf-6lo-ap-nd]
Thubert, P., Sarikaya, B., Sethi, M., and R. Struik, "Address Protected Neighbor Discovery for Low-power and Lossy Networks", draft-ietf-6lo-ap-nd-23 (work in progress), April 2020.
- [I-D.ietf-6lo-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-20 (work in progress), March 2020.
- [I-D.ietf-6tisch-dtsecurity-zerotouch-join]
Richardson, M., "6tisch Zero-Touch Secure Join protocol", draft-ietf-6tisch-dtsecurity-zerotouch-join-04 (work in progress), July 2019.
- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-30 (work in progress), October 2020.

- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M.,
and K. Watsen, "Bootstrapping Remote Secure Key
Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
keyinfra-45 (work in progress), November 2020.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet
Architecture", draft-ietf-intarea-tunnels-10 (work in
progress), September 2019.
- [I-D.ietf-roll-unaware-leaves]
Thubert, P. and M. Richardson, "Routing for RPL Leaves",
draft-ietf-roll-unaware-leaves-29 (work in progress),
January 2021.
- [RFC0801] Postel, J., "NCP/TCP transition plan", RFC 801,
DOI 10.17487/RFC0801, November 1981,
<<https://www.rfc-editor.org/info/rfc801>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
(IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in
IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,
December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet
Control Message Protocol (ICMPv6) for the Internet
Protocol Version 6 (IPv6) Specification", STD 89,
RFC 4443, DOI 10.17487/RFC4443, March 2006,
<<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5406] Bellovin, S., "Guidelines for Specifying the Use of IPsec
Version 2", BCP 146, RFC 5406, DOI 10.17487/RFC5406,
February 2009, <<https://www.rfc-editor.org/info/rfc5406>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
"IPv6 Flow Label Specification", RFC 6437,
DOI 10.17487/RFC6437, November 2011,
<<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C.
Bormann, "Neighbor Discovery Optimization for IPv6 over
Low-Power Wireless Personal Area Networks (6LoWPANs)",
RFC 6775, DOI 10.17487/RFC6775, November 2012,
<<https://www.rfc-editor.org/info/rfc6775>>.

- [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks", RFC 6997, DOI 10.17487/RFC6997, August 2013, <<https://www.rfc-editor.org/info/rfc6997>>.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <<https://www.rfc-editor.org/info/rfc7102>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.
- [RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <<https://www.rfc-editor.org/info/rfc8504>>.
- [RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <<https://www.rfc-editor.org/info/rfc8505>>.

Authors' Addresses

Maria Ines Robles
Universidad Tecno. Nac.(UTN)-FRM, Argentina/ Aalto University Finland

Email: mariainesrobles@gmail.com

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: <http://www.sandelman.ca/mcr/>

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

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

ROLL
Internet-Draft
Updates: 6550 (if approved)
Intended status: Standards Track
Expires: 28 September 2020

P. Thubert, Ed.
Cisco Systems
R.A. Jadhav
Huawei Tech
L. Zhao
Cisco Systems
D. Barthel
Orange Labs
27 March 2020

Eliding and Querying RPL Information
draft-thubert-roll-eliding-dio-information-04

Abstract

This document presents a method to safely elide a group of RPL options in a DIO message by synchronizing the state associated with each of these options between parent and child using a new sequence counter in DIO messages. A child that missed a DIO message with an update of any of those protected options detects it by the change of sequence counter and queries the update with a DIS Message. The draft also provides a method to fully elide the options in a DAO message.

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 September 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Terminology	4
2.1. BCP 14	4
2.2. References	4
2.3. Glossary	5
3. Updating RFC 6550	5
4. Message Formats	6
4.1. Updated DIO Base Object	6
4.2. Updated DIS Base Object	6
4.3. Updated DAO Base Object	7
4.4. New Abbreviated Option Option	8
5. RCSS Operation	8
5.1. Updating the RCSS	9
5.2. RCSS Freshness and Parent selection	10
5.3. RCSS of an Option	10
6. Synchronizing Options	12
7. Abbreviating the DAO Message	12
8. Security Considerations	13
9. IANA Considerations	13
9.1. New DODAG Information Solicitation Flags	14
9.2. New DODAG Advertisement Object Flag	14
9.3. New RPL Control Message Option	14
10. Acknowledgments	15
11. Normative References	15
12. Informative References	16
Authors' Addresses	16

1. Introduction

Classical Link State protocols synchronize their Link State Database (LSDB) by sequencing every change. Each interested node maintains the last sequence of the LSDB it is synchronizing with. If the last known sequence number is older than the current, the node needs to learn one by one all the state changes between the last known and the current state.

"RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" (LLNs) [RPL] does not operate that way. With RPL, the routing information is repeated over and over in DODAG Information Object (DIO) and Destination Advertisement Object (DAO) messages. There is no concept of synchronization. Still there is a concept of sequence to ensure that the most recent information is recognized and overrides a previous one. A stale state may exist in dead branches of the network and eventually time out.

The RPL way was designed to enable routing from most nodes to most nodes most of the time in a Low-Power Lossy Network (LLN) where the quality of the links and the cost of communications does not permit to maintain a permanent synchronization. This principle was applied to both the routing and non-routing information such as configuration settings, prefix information, and node capabilities.

This non-routing state is carried in RPL Messages as options. Some of the DIO options may be needed to decide whether a node can join a network as a leaf or as a router, and may affect the parent selection or the address selection. It is thus critical that each node maintains its state to the freshest and selects parents that are also synchronized to the freshest.

[RPL] allows a parent to elide options in the DIO messages that it sends repeatedly, to conserve battery and save bandwidth. When it does so, a newcomer child that missed DIOs that contained the configuration option may operate on default or partial information. If it is pessimistic, it may query all possible information even when it is not needed. Likewise, a node that slept may have missed a DIO with a change in some critical information and may not be even aware of it, so it may fail to query for the update and operate on deprecated parameters.

This document uses a new sequence counter called the RPL Configuration State Sequence (RCSS) to synchronize the state in a child node with that of its parent, and recursively with that of the whole network, to the latest setting from the Root.

The protected options are:

- * The Route Information Option (RIO) defined in section 6.7.5 of [RPL]
- * The DODAG Configuration Option (DCO) defined in section 6.7.6 of [RPL]
- * The Prefix Information Option (PIO) defined in section 6.7.10 of [RPL]
- * The Extended MOP Option (MOPex) defined in [MOPEX]
- * The Capability Option and TLVs defined in [CAPABILITIES]

Any change in those options causes an increment of the RCSS and enables a network-wide synchronization to the new state. If the change impacts the routing substantially, the Root should decide to increment the Version Number at the same time to fully rebuild the DODAG with the new settings of the options. It must be noted that rebuilding the DODAG does not guarantee that the non-routing state is fully synchronized unless all the options were present in all the DIO messages since the new Version is used.

2. Terminology

2.1. BCP 14

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

The Terminology used in this document is consistent with and incorporates that described in Terms Used in Routing for Low-Power and Lossy Networks [RFC7102].

Other terms in use in LLNs are found in Terminology for Constrained-Node Networks [RFC7228].

A glossary of classical RPL acronyms is given in Section 2.3.

The term "byte" is used in its now customary sense as a synonym for "octet".

"RPL", "RPL Packet Information" (RPI) and "RPL Instance", DIO, DAO and DIS messages are defined in the "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RPL] specification.

This document uses the terms RPL Leaf, RPL Aware Leaf (RAL), RPL-Aware Node (RAN) and RPL-Unaware Leaf (RUL) as defined in section 2 of [USE_OF_RPL_INFO].

A RPL-Unaware Leaf (RUL) thus refers to a host that does not understand RPL but uses a RPL router (without necessarily knowing it) as default gateway and depends on that router to obtain reachability for its addresses inside the RPL domain. Conversely, the term RPL-Aware Node (RAN) is used to refer to a node that participates to RPL and advertises its addresses or prefixes by itself.

2.3. Glossary

This document often uses the following acronyms:

DODAG: Destination-Oriented Directed Acyclic Graph

LLN: Low-Power and Lossy Network

RPI: RPL Packet Information (an Option in the Hop-By-Hop Header)

RAL: RPL-Aware Leaf

RAN: RPL-Aware Node

RS: Router Solicitation

RCSS: RPL Configuration State Sequence

RPL: IPv6 Routing Protocol for LLNs (pronounced ripple)

RUL: RPL-Unaware Leaf

3. Updating RFC 6550

This document adds a new field called RCSS to the DIO message. The RCSS is a sequence counter that is set by the Root and incremented as specified in Section 7 of [RPL], more in Section 5.

This document also introduces a new RPL Control Message Option called the Abbreviated Option Option (AOO). The AOO is the compressed replacement of a protected option that indicates the RCSS of the last change of that option, but elides its content, more in Section 4.4.

This document modifies the DIS Base Object to enable the individual query of the protected options by a node that missed a change, more in Section 4.2.

This document also enables to abbreviate a full DAO message when all the options are unchanged from the most recent DAO message that was positively acknowledged. In that case the DAO is resent with the same DAOSequence and all the options are elided. A new flag in the DAO Base Object indicates that this is an abbreviated DAO message, more in Section 7.

The abbreviated DAO renews the lifetime of a DAO state but does not change any information therein.

4. Message Formats

4.1. Updated DIO Base Object

The format of the DIO Base Object is defined in section 6.3.1 of [RPL]. This specification uses the 8th octet, which was reserved in [RPL], to transport the RCSS.

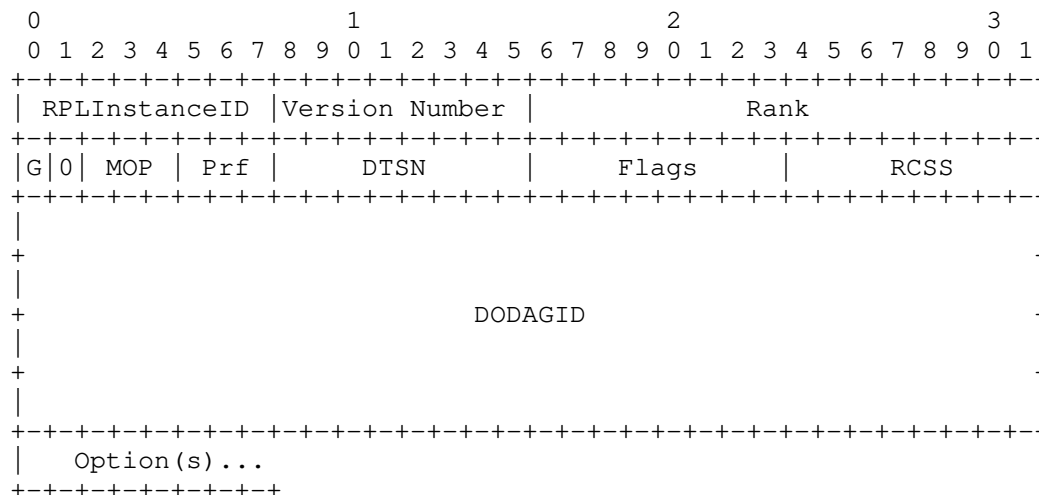


Figure 1: Updated DIO Base Object

Updated fields:

RCSS:

One Byte, the RPL Configuration State Sequence

4.2. Updated DIS Base Object

The DIS Base Object is use by a child to query from a parent the most recent changes in protected options. This specification adds flags to indicate which options are requested and the freshest RCSS to which the querying node was synchronized.

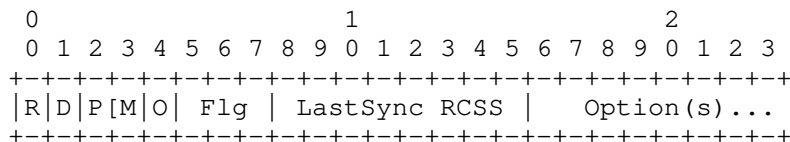


Figure 2: Updated DIS Base Object

Updated fields:

R:

One Bit, indicates that the RIO is requested

D:

One Bit, indicates that the DCO is requested

P:

One Bit, indicates that the PIO(s) is(are) requested

M:

One Bit, indicates that the MOPex is requested

O:

One Bit, indicates that the GCO is requested

Last Synchronized RCSS:

One Byte, the freshest RCSS to which the sender was synchronized

4.3. Updated DAO Base Object

The format of the DAO Base Object is defined in section 6.4.1 of [RPL]. This specification adds the 'A' flag to indicate that the DAO options are elided.

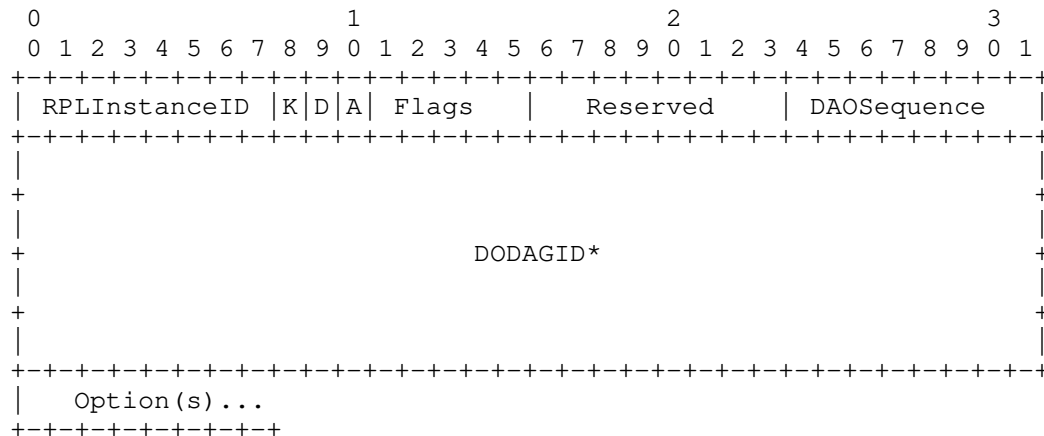


Figure 3: Updated DAO Base Object

Updated fields:

A:

One Bit, indicates DAO in abbreviated version

4.4. New Abbreviated Option Option

The Abbreviated Option Option (AOO) is a generic replacement for an option that only indicates the sender's value of the RCSS for that option. The format of the AOO is represented in Figure 4:

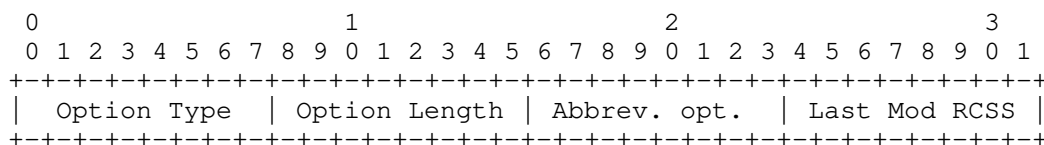


Figure 4: Abbreviated Option Option Format

Option fields:

Option Type:

One byte indicating "Abbreviated Option", see Table 3 in Section 9.3

Option Length:

MUST be set to 2 indicating Option data of 2 bytes

Abbreviated Option:

The Option Type of the option being abbreviated

Last Modification RCSS:

The RCSS at which the option was last modified

5. RCSS Operation

Settings and updates to network-wide parameters are initiated by the Root and propagated down the DODAG in RPL Control Message Options in DIO messages. The DIO messages arrive asynchronously via different parents and may confuse a child when they are not ordered.

The RCSS is a sequence number that is operated as specified in section 7.2 of [RPL]. The RCSS sequences the atomic state that is transported in the protected options in one or a burst of DIO Messages. The same value of the RCSS is used in the initial burst and in the subsequent DIO messages that are sent with no change in the protected options.

The Root of the DODAG is authoritative to set and update the RCSS and the options that it protects. The scope of an RCSS is one DODAG within one RPL Instance.

The RCSS and the sequenced state in the protected options are propagated together down the DODAG without a change, more in Section 5.1.

The RCSS allows a child to remain synchronized to a most recent settings of the network-wide parameters that are propagated in the protected options. The child recognizes stale DIO message(s) and only uses parents with a consistent state, more in Section 5.2.

By extension, the RCSS is also defined for each protected option. A child associates an option with the values of the RCSS indicated in DIO Messages in which the option was advertised and uses it to assess the relative freshness of different versions of an option, more in Section 5.3.

Unchanged options may be sent in full, elided, or in the abbreviated form specified in Section 4.4. Which form to use depends on the RCSS, more in Section 5.3

If the link MTU does not permit to send a single DIO message with all the options packaged then the options may be spread over multiple consecutive DIO messages with the same RCSS that are sent in a rapid sequence.

5.1. Updating the RCSS

The RCSS is incremented by the Root using a lollipop technique as specified in section 7.2 of [RPL]. RCSS values are comparable if they are within a window of comparison of SEQUENCE_WINDOW increments or one indicates a reboot. A reboot of the Root is detected when the RCSS moves from the circular to the straight part of the lollipop.

During the straight part of the lollipop, a second reboot of the Root might not be recognized and a same value of the RCSS may reappear with different settings in the protected options. For that reason the protected options MUST be provided in full with each increment on the RCSS during the straight part of the lollipop.

The straight part should be kept short with a RECOMMENDED initial value of 252 or above. The Root SHOULD jump rapidly away from the straight part once the network has sufficiently settled by resetting the RCSS to 0, which places the RCSS in the circular region of the lollipop, where the protected options MAY be elided or abbreviated.

When a field is modified in one of the protected options, the Root MUST send a DIO with the RCSS incremented and the modified protected option(s) in full. The Root MAY also update the Version Number to form a new DODAG altogether.

5.2. RCSS Freshness and Parent selection

A child node maintains the freshest RCSS received from its parents in each of the RPL Instances that it participates to, and uses that RCSS for its own DIO messages once it has synchronized all the protected options to that RCSS.

A child and a candidate parent are out-of-sync when the RCSS values that they maintain for a RPL Instance are not comparable. A child **MUST NOT** use a parent that is out-of-sync unless no other parent is available, in which case it **MAY** align its RCSS and synchronize to that parent.

When a child receives from a candidate parent a DIO with an RCSS that is fresher than the one it is using, the child **MUST** synchronize the state relative to the protected options with that parent. The child node **MUST** refrain from using that parent and the new state including the RCSS, until it has synchronized all of the protected options to that RCSS. When it is fully synchronized, the child may then use that parent and the new RCSS.

Using a back-level parent may cause packets to be dropped, misunderstood or misrouted. The child **SHOULD** refrain from using a parent that exposes an older RCSS if the change causes an incompatibility issue.

5.3. RCSS of an Option

By extension, the RCSS of an option is maintained by all nodes and is defined for all but the Root as the freshest RCSS indicated by a DIO message from a candidate parent in which the option was present, in the abbreviated form or in full. A child maintains a state for the RCSS of each of the protected options and synchronizes its state for the options by comparing that RCSS with the one found in new DIO messages for the option.

Protected options may be sent in full, elided, or in the abbreviated form. Which form to use depends on the RCSS of the option that a parent maintains:

- * A parent **MAY** use either form when the RCSS is not changed from a previous DIO; eliding options is **PREFERRED** in stable conditions to save resources.
- * When a protected option is updated, the RCSS is mechanically incremented, and the new option **MUST** be sent in full on the first DIO that advertises that new RCSS and the corresponding AOO **SHOULD NOT** be added.

- * When the RCSS is updated but a protected option is unchanged, the parent SHOULD NOT fully elide the option as it may cause multiple children to synchronize it to no avail. The use of AOO is RECOMMENDED unless it may cause a desynchronization for that option, in which case the option SHOULD be placed in full, more in Section 5.3.

When a child receives a DIO from a candidate parent, for each option:

If the Option is advertised in the abbreviated form, then the RCSS that the DIO advertises for the option is the Last Modification RCSS of the AOO, else

If the Option is advertised in full, then the RCSS that the DIO advertises for the option is the RCSS of the DIO, else

If the Option is elided, then the RCSS is unspecified but it is at most as fresh as the RCSS of the DIO, and the RCSS of the DIO is assumed for the comparison

This means that if an Option is advertised in both the abbreviated form and in full in a same DIO message then the RCSS in the AOO has precedence.

To keep the RCSS comparable for each option, the RCSS of an option must lazily progress along with the global RCSS even if there was no change in the options. Each parent including the Root MUST advertise a new RCSS for each of the protected options at least once within a sliding window of SEQUENCE_WINDOW increments.

When an option was not changed for a new RCSS, one parent may advertise it in the abbreviated form while another sends the option in full only, e.g., in response to a DIS message. A fresher RCSS indicates that the option is either the same or carries a more recent update than the one with an older RCSS.

The RCSS of an option may be obtained from a DIO message that carries the option in full even if the RCSS of the DIO is not the freshest across parents, as long as the RCSS of the DIO is fresher than the current one for that option.

If the current value of the maintained RCSS for a given option is not fresher or as fresh as that advertised in a DIO message, then the child MUST update its state for that option as specified in Section 6.

6. Synchronizing Options

As the value of the RCSS progresses, a child MUST NOT attempt to synchronize its state with a parent that advertises a value of RCSS that is out-of-sync with self, or that is already back level vs. the most recent known RCSS for each protected option, unless it lost reachability to all the candidate parents that advertise a fresher and not out-of-sync value of RCSS.

A child can synchronize any of the protected options to the latest RCSS by sending a DIS Message to a candidate parent that advertises that RCSS in DIO messages. The child MUST set the desired combination of 'R', 'D', 'P', 'M' and 'O' flags to indicate the option(s) that it needs updated. The child MUST signal in the Last Synchronized RCSS field of the DIS the freshest value of RCSS for which it was fully synchronized, or a conventional value of OUT-OF-SYNC-RCSS of 129 if it was never synchronized or is out-of-sync with the parent.

The DIO message that is sent in response MUST contain in full all the options that are requested and that were updated since the Last Synchronized RCSS in the DIS Message. This means all of the protected options if the child was never synchronized or is out-of-sync with the parent. The other options MUST be added in the abbreviated form.

The options MAY be spread over more than one DIO message sent in a quick sequence. It is possible that the DIS is not received by the parent or that a DIO that carries all or subset of the requested options is lost in return. In that case the child MUST resend a DIS with the bits associated to the options that are still missing after a reasonable technology-dependent time before it retries the request. The child MAY use any parent that advertises the RCSS to get any of the options up to that level.

7. Abbreviating the DAO Message

When a node receives a positive DAO-ACK upon a DAO message for a given DAOSequence, The DAO-ACK indicates that the DAO was fully processed by its destination (parent or Root).

Until there is a change in one of the DAO options since that DAOSequence, the next DAO messages merely refresh the lifetime of the routes. In that case, increasing the DAOSequence creates undesirable churn up the DODAG for no added value. This specification enables a node to refresh the state in a destination that is associated to one or more DAO message(s) that were acknowledged by that destination without resending the DAO message(s) in full.

Instead, the node MAY use a single abbreviated DAO message that is sent to the same destination and with the same DAOSequence as the DAO message(s) that it refreshes, and with the 'A' flag set (see Section 4.3) to signal it is an abbreviated DAO.

This can be more than one message if the node could not package all its state in a single message, e.g., due to MTU restrictions. In that case the DAO state that is refreshed is the aggregation of the DAO messages that were acknowledged for the provided DAOSequence by that destination.

Upon the abbreviated DAO, the destination refreshes the state associated to the original DAO message(s) received with that DAOSequence, typically by extending the lifetimes of the routes that were advertised with the same duration.

A node MAY also unset 'K' flag in the abbreviated DAP message and not expect a DAO-ACK, if the node can assume the risk that the DAO is lost, e.g., if the routes will be refreshed again before the lifetime expires.

Only the DAO message(s) with the last (freshest) DAOSequence can be a abbreviated. A node MUST NOT use an abbreviated DAO with a DAOSequence that is not the freshest and it MUST NOT use the abbreviated form of the DAO until the destination has acknowledged all state associated with that DAOSequence. If a destination receives an abbreviated DAO with a DAOSequence that is not the freshest from that node, or the destination does not have a state for that node, then MUST send a DAO-ACK with a DAO Status indicating an error. The destination MAY use a new Status of 'Out-of-Sync' in which case the node MUST resent the DAO Message(s) in full with its freshest DAOSequence and the destination synchronizes to that level.

It is RECOMMENDED to use an abbreviated DAO messages whenever possible, because a smaller DAO message consumes less energy and bandwidth and has better chances of delivery. In Non-Storing Mode the benefits increases with the number of hops to the Root, and in Storing Mode with the amount of state that is implicitly refreshed.

8. Security Considerations

TBD

9. IANA Considerations

9.1. New DODAG Information Solicitation Flags

5 new bits are allocated in the Registry for the DODAG Information Solicitation (DIS) Flags defined for [RPL].

Bit Number	Capability description	Reference
0	'R' bit "RIO requested"	THIS RFC
1	'D' bit "DCO requested"	THIS RFC
2	'P' bit "PIO(s) requested"	THIS RFC
3	'M' bit "MOPex requested"	THIS RFC
4	'O' bit "GCO iredrequested"	THIS RFC

Table 1: New DIS Flags

9.2. New DODAG Advertisement Object Flag

1 new bit is allocated in the Registry for the Destination Advertisement Object (DAO) Flags defined for [RPL].

Bit Number	Capability description	Reference
2	'A' bit "DAO abbreviated"	THIS RFC

Table 2: New DAO Flag

9.3. New RPL Control Message Option

A new entry is required for the new option of type "Abbreviated Option", from the "RPL Control Message Options" space defined for [RPL].

Code	Description	Reference
TBD IANA	Abbreviated Option	THIS RFC

Table 3: New Option Type

10. Acknowledgments

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [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>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [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>>.
- [USE_OF_RPL_INFO] Robles, 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", Work in Progress, Internet-Draft, draft-ietf-roll-useofrplinfo-38, 23 March 2020, <<https://tools.ietf.org/html/draft-ietf-roll-useofrplinfo-38>>.
- [CAPABILITIES] Jadhav, R., Thubert, P., Richardson, M., and R. Sahoo, "RPL Capabilities", Work in Progress, Internet-Draft, draft-ietf-roll-capabilities-02, 11 March 2020, <<https://tools.ietf.org/html/draft-ietf-roll-capabilities-02>>.
- [MOPEX] Jadhav, R., Thubert, P., and M. Richardson, "Mode of Operation extension", Work in Progress, Internet-Draft,

draft-jadhav-roll-mopex-02, 6 March 2020,
<<https://tools.ietf.org/html/draft-jadhav-roll-mopex-02>>.

12. Informative References

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

Li Zhao
Cisco Systems, Inc
Xinsi Building
No. 926 Yi Shan Rd
SHANGHAI
200233
China

Email: liz3@cisco.com

Dominique Barthel
Orange Labs
28 chemin du Vieux Chêne
38243 Meylan
France

Email: dominique.barthel@orange.com

ROLL
Internet-Draft
Updates: 6550,8138 (if approved)
Intended status: Standards Track
Expires: January 9, 2020

P. Thubert, Ed.
L. Zhao
Cisco Systems
July 8, 2019

Configuration option for RFC 8138
draft-thubert-roll-turnon-rfc8138-03

Abstract

This document complements RFC 8138 and dedicates a bit in the RPL configuration option defined in RFC 6550 to indicate whether RFC 8138 compression is used within the RPL instance.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. BCP 14	2
3. Updating RFC 6550	2
4. Updating RFC 8138	3
5. Transition Scenarios	3
5.1. Inconsistent State While Migrating	4
5.2. Single Instance Scenario	5
5.3. Double Instance Scenario	5
5.4. Rolling Back	6
6. IANA Considerations	6
7. Security Considerations	6
8. Acknowledgments	6
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Authors' Addresses	7

1. Introduction

The transition to [RFC8138] in a network can only be done when all nodes support the specification. In a mixed case with both RFC8138-capable and non-capable nodes, the compression should be turned off.

This document complements RFC 8138 and dedicates a bit in the RPL configuration option to indicate whether RFC 8138 compression should be used within the RPL instance. When the bit is not set, source nodes that support RFC 8138 should refrain from using the compression unless the information is superseded by configuration.

2. BCP 14

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.

3. Updating RFC 6550

RPL defines a configuration option that is registered to IANA in section 20.14. of [RFC6550]. This specification defines a new flag "Enable RFC8138 Compression" (T) that is encoded in one of the reserved control bits in the option. The new flag is set to turn on the use of the compression of RPL artifacts with RFC 8138.

4. Updating RFC 8138

This document specifies controls that enable and disable the use of the [RFC8138] compression in a RPL Instance. Arguably, this could have been done in [RFC8138] itself.

A node that supports this specification SHOULD source packets in the compressed form using [RFC8138] if the new "T" flag is set in the RPL configuration option from its parents. Failure to do so will result in larger packets, yields higher risks of loss and may cause a fragmentation.

A node that supports this specification SHOULD refrain from sourcing packets in the compressed form using [RFC8138] if the "T" flag is reset. This behavior can be overridden by a configuration of the node in order to cope with intermediate implementations of the root that support [RFC8138] but not this specification and cannot set the "T" flag.

The decision of using RFC 8138 to compress a packet is made at the source depending on its capabilities and its knowledge of the state of the "T" flag. A router MUST forward the packet in the form that the source used, either compressed or uncompressed. A router that encapsulates a packet is the source of the resulting packet and the rules above apply to it in that case.

5. Transition Scenarios

It is RECOMMENDED to only deploy nodes that support [RFC8138] in a network where the compression is turned on. A node that does not support [RFC8138] MUST only be used as a leaf in that network.

[RFC6550] states that "Nodes other than the DODAG root MUST NOT modify this information when propagating the DODAG Configuration option". In other words, the configuration option is a way for the root to configure the LLN nodes but it cannot be used by a parent to advertise its capabilities down the DODAG. It results whether a parent supports RFC 8138 is not known by the child with the current level of specifications, and a child cannot favor a parent based on a particular support.

Sections 8.5 and 9.2 of [RFC6550] also suggests that a RPL-aware node may attach to a DODAG as a leaf node only, e.g., when a node does not support the Mode of Operation of a RPL Instance, the Objective Function (OF) as indicated by the Objective Code Point (OCP) or some other parameters in the configuration option. But the node is also free to refrain from joining an Instance when a parameter is not suitable. This means that changing the OCP in a DODAG can be used to

force nodes that do not support a particular feature to join as leaf only. This specification reiterates that a node that is configured to operate in an Instance but does not support a value for a known parameter that is mandatory for routing MUST NOT operate as a router but MAY still joins as a leaf. Note that a legacy node will not recognize when a reserved field is now used and will not turn to a leaf when that happens.

A node that supports [RFC8138] but not this specification can only be used in an homogeneous network and an upgrade requires a "flag day" where all nodes are updated and then the network is rebooted with implicitly RFC 8138 compression turned on with the "T" flag set on.

A node that supports this specification can work in a network with RFC 8138 compression turned on or off with the "T" flag set accordingly and in a network in transition from off to on or on to off (see Section 5.1).

A node that does not support [RFC8138] can interoperate with a node that supports this specification in a network with RFC 8138 compression turned off. But it cannot forward compressed packets and therefore it cannot act as a router in a network with RFC 8138 compression turned on. It may remain connected to that network as a leaf and generate uncompressed packets as long as incoming packets are decapsulated by the parent and delivered in uncompressed form.

The intent for this specification is to perform a migration once and for all without the need for a flag day. In particular it is not the intention to undo the setting of the "T" flag, and though it is possible to roll back (see Section 5.4), adding nodes that do not support [RFC8138] after a roll back may be problematic if the roll back is not fully complete (see caveats in Section 5.2).

5.1. Inconsistent State While Migrating

When the "T" flag is turned on in the configuration option by the root, the information slowly percolates through the DODAG as the DIO gets propagated. Some nodes will see the flag and start sourcing packets in the compressed form while other nodes in the same instance are still not aware of it. Conversely, in non-storing mode, the root will start using RFC 8138 with a SRH-6LoRH that routes all the way to the last router or possibly to the leaf, if the leaf supports RFC 8138.

This is why it is required that all the routers in the Instance support [RFC8138] at the time of the switch, and all nodes that do not support [RFC8138] only operate as leaves.

Setting the "T" flag is ultimately the responsibility of the network administrator. In a case of upgrading a network to turn the compression on, the network SHOULD be operated with the "T" flag reset until all targeted nodes are upgraded to support this specification. Section 5.2 and Section 5.3 provide possible transition scenarios where this can be enforced.

5.2. Single Instance Scenario

In a single instance scenario, nodes that support RFC 8138 are configured with a new OCP, that may use the same OF operation or a variation of it. when it finally sets the "T" flag, the root also migrates to the new OCP. As a result, nodes that do not support RFC 8138 join as leaves and do not forward packets anymore. The leaves generate packets without compression. The parents - which supports RFC 8138 - may encapsulate the packets using RFC 8138 if needed. The other way around, the root encapsulates packets to the leaves all the way to the parent, which decapsulates and distribute the uncompresses inner packet to the leaf, as illustrated in Section 4.3 of [I-D.ietf-roll-useofrplinfo]

This scenario presents a number of caveats:

- o The method consumes an extra OCP. It also requires a means to signal the capabilities of the leaf, e.g., using "RPL Mode of Operation extension" [I-D.rahul-roll-mop-ext].
- o If an implementation does not move to a leaf mode when the OCP is changed to an unknown one, then the node may be stalled.
- o If the only possible parents of a node are nodes that do not support RFC 8138, then that node will loose all its parent at the time of the migration and it will be stalled until a parent is deployed with the new capability.
- o Nodes that only support RFC8138 for forwarding may not parse the RPI in native form. If such nodes are present, the parent needs to encapsulate with RFC8138.

5.3. Double Instance Scenario

An alternate to the Single Instance Scenario is to deploy an additional Instance for the nodes that support [RFC8138]. The two instances operate as ships-in-the-night as specified in [RFC6550]. The preexisting Instance that does not use [RFC8138], whereas the new Instance does. This is signaled by the "T" flag which is only set in the configuration option in DIO messages in the new Instance.

The legacy nodes would not be configured to participate to the second instance, and islands that are only connected via legacy nodes would not be reachable over the second instance.

Nodes that support RFC 8138 participate to both Instances but favor the new Instance for the traffic that they source. On the other hand, nodes that only support the uncompressed format would either not be configured for the new instance, or would be configured to join it as leaves only.

This method eliminates the risks of nodes being stalled that are described in Section 5.2 but requires implementations to support at least two RPL Instances and demands management capabilities to introduce new Instances and deprecate old ones.

5.4. Rolling Back

After downgrading a network to turn the [RFC8138] compression off, the administrator SHOULD make sure that all nodes have converged to the "T" flag reset before allowing nodes that do not support the compression in the network (see caveats in Section 5.2). This also requires a means to signal the current state of the setting of the logic that controls the compression in the node, also using [I-D.rahul-roll-mop-ext].

6. IANA Considerations

This specification updates the "Registry for the DODAG Configuration Option Flags" that was created for [RFC6550] as follows:

Bit Number	Meaning	Defining Spec
2 (suggested)	Turn on RFC8138 Compression (T)	This

Table 1: New DODAG Configuration Option Flag

7. Security Considerations

No specific threat was identified with this specification.

8. Acknowledgments

9. References

9.1. Normative References

- [I-D.ietf-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "Using RPL Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane", draft-ietf-roll-useofrplinfo-31 (work in progress), July 2019.
- [I-D.rahul-roll-mop-ext]
Jadhav, R. and P. Thubert, "RPL Mode of Operation extension", draft-rahul-roll-mop-ext-01 (work in progress), June 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [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>>.

Authors' Addresses

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

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

Li Zhao
Cisco Systems, Inc
Xinsi Building
No. 926 Yi Shan Rd
SHANGHAI 200233
CHINA

Email: liz3@cisco.com