Supporting Asymmetric Links in Low Power Networks: AODV-RPL
draft-ietf-roll-aodv-rpl-15

Abstract

   Route discovery for symmetric and asymmetric Peer-to-Peer (P2P)
   traffic flows is a desirable feature in Low power and Lossy Networks
   (LLNs). For that purpose, this document specifies a reactive P2P
   route discovery mechanism for both hop-by-hop routes and source
   routing: Ad Hoc On-demand Distance Vector Routing (AODV) based RPL
   protocol (AODV-RPL). Paired Instances are used to construct
   directional paths, for cases where there are asymmetric links between
   source and target nodes.

Status of This Memo

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

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

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

   This Internet-Draft will expire on 3 April 2023.

Copyright Notice

Table of Contents

1. Introduction

   Routing Protocol for Low-Power and Lossy Networks (RPL) [RFC6550] is
   an IPv6 distance vector routing protocol designed to support multiple
   traffic flows through a root-based Destination-Oriented Directed
   Acyclic Graph (DODAG). Typically, a router does not have routing
   information for most other routers. Consequently, for traffic
   between routers within the DODAG (i.e., Peer-to-Peer (P2P) traffic)
   data packets either have to traverse the root in non-storing mode, or
   traverse a common ancestor in storing mode. Such P2P traffic is
   thereby likely to traverse longer routes and may suffer severe
   congestion near the root (for more information see [RFC6687],
   [RFC6997], [RFC6998], [RFC9010]). The network environment that is
   considered in this document is assumed to be the same as described in
   Section 1 of [RFC6550]. Each radio interface/link and the associated
   address should be treated as an independent intermediate router.
   Such routers have different links and the rules for the link symmetry
   apply independently for each of these.

   The route discovery process in AODV-RPL is modeled on the analogous
   peer-to-peer procedure specified in AODV [RFC3561]. The on-demand
   nature of AODV route discovery is natural for the needs of routing in
   RPL-based LLNs when routes are needed but aren't yet established.
   Peer-to-peer routing is desirable to discover shorter routes, and
   especially when it is desired to avoid directing additional traffic
   through a root or gateway node of the network. It may happen that
   some routes need to be established proactively when known beforehand
   and when AODV-RPL's route discovery process introduces unwanted delay
   at the time when the application is launched.

AODV terminology has been adapted for use with AODV-RPL messages,
namely RREQ for Route Request, and RREP for Route Reply. AODV-RPL
currently omits some features compared to AODV -- in particular,
flagging Route Errors, "blacklisting" unidirectional links
([RFC3561]), multihoming, and handling unnumbered interfaces.

AODV-RPL reuses and extends the core RPL functionality to support
routes with bidirectional asymmetric links. It retains RPL's DODAG
formation, RPL Instance and the associated Objective Function
(defined in [RFC6551]), trickle timers, and support for storing and
non-storing modes. AODV-RPL adds basic messages RREQ and RREP as
part of RPL DODAG Information Object (DIO) control message, which go
in separate (paired) RPL instances. AODV-RPL does not utilize the
Destination Advertisement Object (DAO) control message of RPL. AODV-
RPL uses the "P2P Route Discovery Mode of Operation" (MOP == 4) with
three new Options for the DIO message, dedicated to discover P2P
routes. These P2P routes may differ from routes discoverable by
native RPL. Since AODV-RPL uses newly defined Options and a newly
allocated multicast group (see Section 9), there is no conflict with
P2P-RPL [RFC6997], a previous document using the same MOP. AODV-RPL
can be operated whether or not P2P-RPL or native RPL is running
otherwise. For many networks AODV-RPL could be a replacement for
RPL, since it can find better routes at very moderate cost.
Consequently, it seems unlikely that RPL would be needed in a network
that is running AODV-RPL, even though it would be possible to run
both protocols at the same time.

2. Terminology

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

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

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

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

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

Bi-directional Asymmetric Link
   A link that can be used in both directions but with different link
   characteristics.

DIO
   DODAG Information Object (as defined in [RFC6550])

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

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

Downward Direction
   The direction from the OrigNode to the TargNode.

Downward Route
   A route in the downward direction.

hop-by-hop route
   A route for which each router along the routing path stores
   routing information about the next hop. A hop-by-hop route is
   created using RPL's "storing mode".

on-demand routing
   Routing in which a route is established only when needed.

OrigNode
   The IPv6 router (Originating Node) initiating the AODV-RPL route
   discovery to obtain a route to TargNode.

Paired DODAGs
   Two DODAGs for a single route discovery process between OrigNode
   and TargNode.

    P2P
       Peer-to-Peer -- in other words, not constrained a priori to
       traverse a common ancestor.

    reactive routing
       Same as "on-demand" routing.

    REJOIN_REENABLE
       The duration during which a node is prohibited from joining a
       DODAG with a particular RREQ-InstanceID, after it has left a DODAG
       with the same RREQ-InstanceID. The default value of
       REJOIN_REENQBLE is 15 minutes.

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

    RREQ-InstanceID
       The RPLInstanceID for the RREQ-Instance. The RREQ-InstanceID is
       formed as the ordered pair (Orig_RPLInstanceID, OrigNode-IPaddr),
       where Orig_RPLInstanceID is the local RPLInstanceID allocated by
       OrigNode, and OrigNode-IPaddr is an IP address of OrigNode. The
       RREQ-InstanceID uniquely identifies the RREQ-Instance.

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

    RREP-InstanceID
       The RPLInstanceID for the RREP-Instance. The RREP-InstanceID is
       formed as the ordered pair (Targ_RPLInstanceID, TargNode-IPaddr),
       where Targ_RPLInstanceID is the local RPLInstanceID allocated by
       TargNode, and TargNode-IPaddr is an IP address of TargNode. The
       RREP-InstanceID uniquely identifies the RREP-Instance. The
       RPLInstanceID in the RREP message along with the Delta value
       indicates the associated RREQ-InstanceID.

    Source routing
       A mechanism by which the source supplies a vector of addresses
       towards the destination node along with each data packet
       [RFC6550].

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

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

Upward Direction
    The direction from the TargNode to the OrigNode.

Upward Route
    A route in the upward direction.

3. Overview of AODV-RPL

   With AODV-RPL, routes from OrigNode to TargNode within the LLN
   network are established "on-demand". In other words, the route
   discovery mechanism in AODV-RPL is invoked reactively when OrigNode
   has data for delivery to the TargNode but existing routes do not
   satisfy the application's requirements. AODV-RPL works without
   requiring the use of RPL or any other routing protocol.

   The routes discovered by AODV-RPL are not constrained to traverse a
   common ancestor. AODV-RPL can enable asymmetric communication paths
   in networks with bidirectional asymmetric links. For this purpose,
   AODV-RPL enables discovery of two routes: namely, one from OrigNode
   to TargNode, and another from TargNode to OrigNode. AODV-RPL also
   enables discovery of symmetric routes along Paired DODAGs, when
   symmetric routes are possible (see Section 5).

   In AODV-RPL, routes are discovered by first forming a temporary DAG
   rooted at the OrigNode. Paired DODAGs (Instances) are constructed
   during route formation between the OrigNode and TargNode. The RREQ-
   Instance is formed by route control messages from OrigNode to
   TargNode whereas the RREP-Instance is formed by route control
   messages from TargNode to OrigNode. The route discovered in the
   RREQ-Instance is used for transmitting data from TargNode to
   OrigNode, and the route discovered in RREP-Instance is used for
   transmitting data from OrigNode to TargNode.

   Intermediate routers join the DODAGs based on the Rank [RFC6550] as
   calculated from the DIO message. Henceforth in this document, "RREQ-
   DIO message" means the DIO message from OrigNode toward TargNode,
   containing the RREQ option as specified in Section 4.1. The RREQ-
   InstanceID is formed as the ordered pair (Orig_RPLInstanceID,
   OrigNode-IPaddr), where Orig_RPLInstanceID is the local RPLInstanceID
   allocated by OrigNode, and OrigNode-IPaddr is the IP address of
   OrigNode. A node receiving the RREQ-DIO can use the RREQ-InstanceID
   to identify the proper OF whenever that node receives a data packet
   with Source Address == OrigNode-IPaddr and IPv6 RPL Option having the
   RPLInstanceID == Orig_RPLInstanceID along with 'D' == 0.

Similarly, "RREP-DIO message" means the DIO message from TargNode
toward OrigNode, containing the RREP option as specified in
Section 4.2. The RREP-InstanceID is formed as the ordered pair
(Targ_RPLInstanceID, TargNode-IPaddr), where Targ_RPLInstanceID is
the local RPLInstanceID allocated by TargNode, and TargNode-IPaddr is
the IP address of TargNode. A node receiving the RREP-DIO can use
the RREP-InstanceID to identify the proper OF whenever that node
receives a data packet with Source Address == TargNode-IPaddr and
IPv6 RPL Option having the RPLInstanceID == Targ_RPLInstanceID along
with 'D' == 0.

4. AODV-RPL DIO Options

4.1. AODV-RPL RREQ Option

OrigNode selects one of its IPv6 addresses and sets it in the DODAGID
field of the RREQ-DIO message. The address scope of the selected
address must encompass the domain where the route is built (e.g, not
link-local). Exactly one RREQ option MUST be present in a RREQ-DIO
message, otherwise the message MUST be dropped.

```
 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  | Option Length |S|H|X| Compr | L |  RankLimit  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Orig SeqNo   |                                               |
+-+-+-+-+-+-+-+-+                                               |
|                                                               |
|                                                               |
|           Address Vector (Optional, Variable Length)          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Format for AODV-RPL RREQ Option

OrigNode supplies the following information in the RREQ option:

Option Type
   TBD2

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

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

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

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

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

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

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

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

     Orig SeqNo
     Sequence Number of OrigNode. See Section 6.1.

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

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

4.2. AODV-RPL RREP Option

   TargNode sets one of its IPv6 addresses in the DODAGID field of the
   RREP-DIO message. The address scope of the selected address must
   encompass the domain where the route is built (e.g, not link-local).
   Exactly one RREP option MUST be present in a RREP-DIO message,
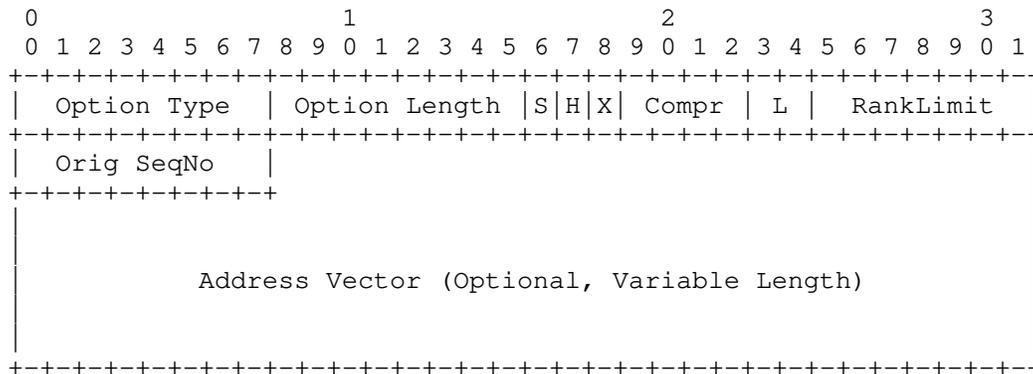   otherwise the message MUST be dropped. TargNode supplies the
   following information in the RREP option:

```
 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  | Option Length |G|H|X| Compr | L |  RankLimit  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Delta   |X X|                                               |
+-+-+-+-+-+-+-+                                                 |
|                                                               |
|                                                               |
|          Address Vector (Optional, Variable Length)           |
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                Figure 2: Format for AODV-RPL RREP option

Option Type
    TBD3

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

G
    Gratuitous RREP (see Section 7).

H
    The H bit in the RREP option MUST be set to be the same as the H
    bit in RREQ option. It requests either source routing (H=0) or
    hop-by-hop (H=1) for the downstream route.

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

Compr
    4-bit unsigned integer. Same definition as in RREQ option.

L
    2-bit unsigned integer defined as in RREQ option. The lifetime of
    the RREP-Instance MUST be no greater than the lifetime of the
    RREQ-Instance to which it is paired.

RankLimit
    Similarly to RankLimit in the RREQ message, this field indicates
    the upper limit on the integer portion of the Rank. A value of 0
    in this field indicates the limit is infinity.

Delta
    6-bit unsigned integer. This field is used to recover the RREQ-
    InstanceID (see Section 6.3.3); 0 indicates that the RREQ-
    InstanceID has the same value as the RPLInstanceID of the RREP
    message.

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

Address Vector
    Only present when the H bit is set to 0. For an asymmetric route,
    the Address Vector represents the IPv6 addresses of the path
    through the network the RREP-DIO has passed. For a symmetric
    route, it is the Address Vector when the RREQ-DIO arrives at the
    TargNode, unchanged during the transmission to the OrigNode.

4.3. AODV-RPL Target Option

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

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

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

```
 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  | Option Length | Dest SeqNo   |X|Prefix Length|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                              |
+                                                              |
|            Target Prefix / Address (Variable Length)         |
.                                                              .
.                                                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 3: ART Option format for AODV-RPL

Option Type
   TBD4

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

Dest SeqNo

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

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

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

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

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

5. Symmetric and Asymmetric Routes

   Links are considered symmetric until indication to the contrary is
   received. In Figure 4 and Figure 5, BR is the Border Router, O is
   the OrigNode, each R is an intermediate router, and T is the
   TargNode. In this example, the use of BR is only for illustrative
   purposes; AODV does not depend on the use of border routers for its
   operation. If the RREQ-DIO arrives over an interface that is known
   to be symmetric, and the S bit is set to 1, then it remains as 1, as
   illustrated in Figure 4. If an intermediate router sends out RREQ-
   DIO with the S bit set to 1, then each link en route from the
   OrigNode O to this router has met the requirements of route
   discovery, and the route can be used symmetrically.

```
                              BR
                          /----+----\
                        /      |      \
                      /        |        \
                    R          R          R
                  _/ \         |        / \
                  /    \       |       /   \
                 /      \      |      /     \
                R -------- R --- R ----- R -------- R
               / \   <--S=1-->  / \   <--S=1-->   / \
          <--S=1-->   \        /   \        /   <--S=1-->
             /          \     /     \      /         \
            O ---------- R ------ R------ R ----- R ----------- T
          / \                  / \        / \          / \
         /   \                /   \      /   \        /   \
        /     \              /     \    /     \      /     \
       R ----- R ----------- R ----- R ----- R ----- R ---- R----- R

        >---- RREQ-Instance (Control: O-->T;  Data: T-->O) ------->
        <---- RREP-Instance (Control: T-->O;  Data: O-->T) -------<
```

               Figure 4: AODV-RPL with Symmetric Instances

   Upon receiving a RREQ-DIO with the S bit set to 1, a node determines
   whether this link can be used symmetrically, i.e., both directions
   meet the requirements of data transmission. If the RREQ-DIO arrives

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

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

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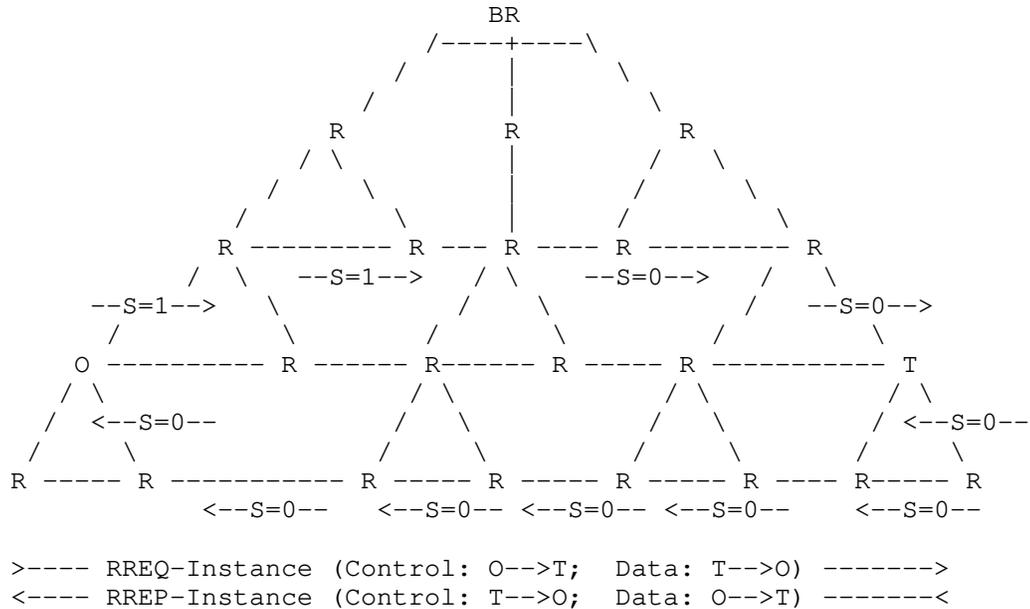of link quality using an averaging technique.

```
                               BR
                         /----+----\
                        /    |     \
                       /     |      \
                      R      R       R
                    /  \     |      /  \
                   /    \    |     /    \
                  /      \   |    /      \
                 R --------- R --- R ---- R --------- R
                / \    --S=1-->  / \    --S=0-->   /   \
          --S=1-->     \        /   \           /    --S=0-->
             /          \      /     \         /          \
            O --------- R ------ R------ R ----- R ----------- T
           / \          / \          / \          / \
          /  <--S=0--  /    \       /   \        / <--S=0--
         /       \    /      \     /     \      /       \
        R ----- R ----------- R ----- R ----- R ----- R ---- R----- R
              <--S=0--    <--S=0-- <--S=0-- <--S=0--    <--S=0--

         >---- RREQ-Instance (Control: O-->T;  Data: T-->O) ------->
         <---- RREP-Instance (Control: T-->O;  Data: O-->T) -------<
```

              Figure 5: AODV-RPL with Asymmetric Paired Instances

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

6. AODV-RPL Operation

6.1. Route Request Generation

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

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

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

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

The transmission of RREQ-DIO obeys the Trickle timer [RFC6206]. If
the duration specified by the L field has elapsed, the OrigNode MUST
leave the DODAG and stop sending RREQ-DIOs in the related
RPLInstance. OrigNode needs to set L field such that the DODAG will
not prematurely timeout during data transfer with the TargNode. For
setting this value, it has to consider factors such as Trickle timer,
TargNode hop distance, network size, link behavior, expected data
usage time, and so on.

6.2. Receiving and Forwarding RREQ messages

6.2.1. Step 1: RREQ reception and evaluation

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

When H=0 in the incoming RREQ, the router MUST drop the RREQ-DIO if
one of its addresses is present in the Address Vector. When H=1 in
the incoming RREQ, the router MUST drop the RREQ message if Orig
SeqNo field of the RREQ is older than the SeqNo value that X has
stored for a route to OrigNode. Otherwise, the router determines
whether to propagate the RREQ-DIO. It does this by determining
whether or not a route to OrigNode using the upstream direction of
the incoming link satisfies the Objective Function (OF). In order to
evaluate the OF, the router first determines the maximum useful rank

(MaxUsefulRank). If the router has previously joined the RREQ-
Instance associated with the RREQ-DIO, then MaxUsefulRank is set to
be the Rank value that was stored when the router processed the best
previous RREQ for the DODAG with the given RREQ-Instance. Otherwise,
MaxUsefulRank is set to be RankLimit. If OF cannot be satisfied
(i.e., the Rank evaluates to a value greater than MaxUsefulRank) the
RREQ-DIO MUST be dropped, and the following steps are not processed.
Otherwise, the router MUST join the RREQ-Instance and prepare to
propagate the RREQ-DIO, as follows. The upstream neighbor router
that transmitted the received RREQ-DIO is selected as the preferred
parent.

6.2.2. Step 2: TargNode and Intermediate Router determination

   After determining that a received RREQ provides a usable route to
   OrigNode, a router determines whether it is a TargNode, or a possible
   intermediate router between OrigNode and a TargNode, or both. The
   router is a TargNode if it finds one of its own addresses in a Target
   Option in the RREQ. After possibly propagating the RREQ according to
   the procedures in Steps 3, 4, and 5, the TargNode generates a RREP as
   specified in Section 6.3. If S=0, the determination of TargNode
   status and determination of a usable route to OrigNode is the same.

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

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

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

6.2.3. Step 3: Intermediate Router RREQ processing

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

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

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

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

   Suppose a router has joined the RREQ-Instance, and H=0, and the S-bit
   of the RREQ-Instance is set to 1. In this case, the router MAY
   optionally associate to the RREQ-Instance, the Address Vector of the
   symmetric route back to OrigNode. This is useful if the router later
   receives an RREP-DIO that is paired with the RREQ-Instance. If the
   router does NOT associate the Address Vector, then it has to rely on
   multicast for the RREP. This can impose a substantial performance
   penalty.

6.2.5. Step 5: RREQ propagation at an Intermediate Router

   If the router is an intermediate router, then it transmits the RREQ-
   DIO to the multicast group all-AODV-RPL-nodes; if the H bit is set to
   0, the intermediate router MUST append the address of its interface
   receiving the RREQ-DIO into the address vector. If, in addition, the
   address of the router's transmitting the RREQ-DIO is not the same as
   the address of the interface receiving the RREQ-DIO, the router MUST
   also append the transmitting interface address into the address
   vector.

6.2.6. Step 6: RREQ reception at TargNode

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

6.3. Generating Route Reply (RREP) at TargNode

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

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

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

6.3.1. RREP-DIO for Symmetric route

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

scope.

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

6.3.2. RREP-DIO for Asymmetric Route

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

6.3.3. RPLInstanceID Pairing

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

When preparing the RREP-DIO, a TargNode could find the RPLInstanceID
candidate for the RREP-Instance is already occupied by another RPL
Instance from an earlier route discovery operation which is still
active. This unlikely case might happen if two distinct OrigNodes
need routes to the same TargNode, and they happen to use the same
RPLInstanceID for RREQ-Instance. In such cases, the RPLInstanceID of
an already active RREP-Instance MUST NOT be used again for assigning
RPLInstanceID for the later RREP-Instance. If the same RPLInstanceID
were re-used for two distinct DODAGs originated with the same DODAGID
(TargNode address), intermediate routers could not distinguish
between these DODAGs (and their associated Objective Functions).
Instead, the RPLInstanceID MUST be replaced by another value so that
the two RREP-instances can be distinguished. In the RREP-DIO option,
the Delta field of the RREP-DIO message (Figure 2) indicates the
increment to be applied to the pre-existing RPLInstanceID to obtain
the value of the RPLInstanceID that is used in the RREP-DIO message.
When the new RPLInstanceID after incrementation exceeds 255, it rolls
over starting at 0. For example, if the RREQ-InstanceID is 252, and

incremented by 6, the new RPLInstanceID will be 2. Related
operations can be found in Section 6.4. RPLInstanceID collisions do
not occur across RREQ-DIOs; the DODAGID equals the OrigNode address
and is sufficient to disambiguate between DODAGs.

6.4. Receiving and Forwarding Route Reply

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

6.4.1. Step 1: Receiving and Evaluation

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

If the S bit of the associated RREQ-Instance is set to 1, the router
MUST proceed to Section 6.4.2.

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

6.4.2. Step 2: OrigNode or Intermediate Router

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

6.4.3. Step 3: Build Route to TargNode

If the H bit is set to 1, then the router (OrigNode or intermediate)
MUST build a downward route entry towards TargNode which includes at
least the following items: OrigNode Address, RPLInstanceID, TargNode
Address as destination, Next Hop, Lifetime and Sequence Number. For
a symmetric route, the Next Hop in the route entry is the router from
which the RREP-DIO is received. For an asymmetric route, the Next
Hop is the preferred parent in the DODAG of RREP-Instance. The

RPLInstanceID in the route entry MUST be the RREQ-InstanceID (i.e., after subtracting the Delta field value from the value of the RPLInstanceID). The source address is learned from the ART Option, and the destination address is learned from the DODAGID. The lifetime is set according to DODAG configuration (i.e., not the L field) and can be extended when the route is actually used. The sequence number represents the freshness of the route entry, and is copied from the Dest SeqNo field of the ART option of the RREP-DIO. A route entry with same source and destination address, same RPLInstanceID, but stale sequence number (i.e., incoming sequence number is less than the currently stored sequence number of the route entry), MUST be deleted.

6.4.4. Step 4: RREP Propagation

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

7. Gratuitous RREP

In some cases, an Intermediate router that receives a RREQ-DIO message MAY unicast a "Gratuitous" RREP-DIO message back to OrigNode before continuing the transmission of the RREQ-DIO towards TargNode. The Gratuitous RREP allows the OrigNode to start transmitting data to TargNode sooner. The G bit of the RREP option is provided to distinguish the Gratuitous RREP-DIO (G=1) sent by the Intermediate router from the RREP-DIO sent by TargNode (G=0).

The gratuitous RREP-DIO MAY be sent out when the Intermediate router receives a RREQ-DIO for a TargNode, and the router has a pair of downward and upward routes to the TargNode which also satisfy the Objective Function and for which the destination sequence number is at least as large as the sequence number in the RREQ-DIO message. After unicasting the Gratuitous RREP to the OrigNode, the Intermediate router then unicasts the RREQ towards TargNode, so that TargNode will have the advertised route towards OrigNode along with the RREQ-InstanceID for the RREQ-Instance.

In case of source routing, the intermediate router MUST include the address vector between the OrigNode and itself in the Gratuitous RREP. It also includes the address vector in the unicast RREQ-DIO towards TargNode. Upon reception of the unicast RREQ-DIO, the TargNode will have a route address vector from itself to the OrigNode. Then the router MUST include the address vector from the TargNode to the router itself in the gratuitous RREP-DIO to be transmitted.

For establishing hop-by-hop routes, the intermediate router MUST unicast the received RREQ-DIO to the Next Hop on the route. The Next Hop router along the route MUST build new route entries with the related RPLInstanceID and DODAGID in the downward direction. This process repeats at each node until the RREQ-DIO arrives at the TargNode. Then the TargNode and each router along the path towards OrigNode MUST unicast the RREP-DIO hop-by-hop towards OrigNode as specified in Section 6.3.

8. Operation of Trickle Timer

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

9. IANA Considerations

   Note to RFC editor:

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

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

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

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

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

Figure 6: AODV-RPL Options

IANA is requested to allocate a new permanent multicast address with
link-local scope called all-AODV-RPL-nodes for nodes implementing
this specification.

10. Security Considerations

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

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

If a rogue router knows the key for the security configuration in
use, it can join the secure AODV-RPL route discovery and cause
various types of damage. Such a rogue router could advertise false
information in its DIOs in order to include itself in the discovered
route(s). It could generate bogus RREQ-DIO, and RREP-DIO messages
carrying bad routes or maliciously modify genuine RREP-DIO messages

it receives. A rogue router acting as the OrigNode could launch denial-of-service attacks against the LLN deployment by initiating fake AODV-RPL route discoveries. When rogue routers might be present, RPL's preinstalled mode of operation, where the key to use for route discovery is preinstalled, SHOULD be used.

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

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

11. Acknowledgements

12. References

12.1. Normative References

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

   [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko,
              "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206,
              March 2011, <https://www.rfc-editor.org/info/rfc6206>.

   [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J.,
              Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur,
              JP., and R. Alexander, "RPL: IPv6 Routing Protocol for
              Low-Power and Lossy Networks", RFC 6550,
              DOI 10.17487/RFC6550, March 2012,
              <https://www.rfc-editor.org/info/rfc6550>.

   [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N.,
              and D. Barthel, "Routing Metrics Used for Path Calculation
              in Low-Power and Lossy Networks", RFC 6551,
              DOI 10.17487/RFC6551, March 2012,
              <https://www.rfc-editor.org/info/rfc6551>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

   [co-ioam] Rashmi Ballamajalu, Anand, S.V.R., and Malati Hegde, "Co-
              iOAM: In-situ Telemetry Metadata Transport for Resource
              Constrained Networks within IETF Standards Framework",
              2018 10th International Conference on Communication
              Systems & Networks (COMSNETS) pp.573-576, January 2018.

   [contiki] Contiki contributors, "The Contiki Open Source OS for the
              Internet of Things (Contiki Version 2.7)", November 2013,
              <https://github.com/contiki-os/contiki>.

   [Contiki-ng]
              Contiki-NG contributors, "Contiki-NG: The OS for Next
              Generation IoT Devices (Contiki-NG Version 4.6)", December
              2020, <https://github.com/contiki-ng/contiki-ng>.

   [cooja]   Contiki/Cooja contributors, "Cooja Simulator for Wireless
              Sensor Networks (Contiki/Cooja Version 2.7)", November
              2013, <https://github.com/contiki-
              os/contiki/tree/master/tools/cooja>.

   [empirical-study]
              Prasant Misra, Nadeem Ahmed, and Sanjay Jha, "An empirical
              study of asymmetry in low-power wireless links", IEEE
              Communications Magazine (Volume: 50, Issue: 7), July 2012.

   [Link_Asymmetry]
              Lifeng Sang, Anish Arora, and Hongwei Zhang, "On Link
              Asymmetry and One-way Estimation in Wireless Sensor
              Networks", ACM Transactions on Sensor Networks, Volume 6
              Issue 2 pp.1-25, February 2010,
              <https://doi.org/10.1145/1689239.1689242>.

   [low-power-wireless]
              Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and
              Philip Levis, "An empirical study of low-power wireless",
              ACM Transactions on Sensor Networks (Volume 6 Issue 2
              pp.1-49), February 2010,
              <https://doi.org/10.1145/1689239.1689246>.

   [RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-
              Demand Distance Vector (AODV) Routing", RFC 3561,
              DOI 10.17487/RFC3561, July 2003,
              <https://www.rfc-editor.org/info/rfc3561>.

   [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>.

   [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and
              J. Martocci, "Reactive Discovery of Point-to-Point Routes
              in Low-Power and Lossy Networks", RFC 6997,
              DOI 10.17487/RFC6997, August 2013,
              <https://www.rfc-editor.org/info/rfc6997>.

   [RFC6998] Goyal, M., Ed., Baccelli, E., Brandt, A., and J. Martocci,
              "A Mechanism to Measure the Routing Metrics along a Point-
              to-Point Route in a Low-Power and Lossy Network",
              RFC 6998, DOI 10.17487/RFC6998, August 2013,
              <https://www.rfc-editor.org/info/rfc6998>.

   [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y.
              Weingarten, "An Overview of Operations, Administration,
              and Maintenance (OAM) Tools", RFC 7276,
              DOI 10.17487/RFC7276, June 2014,
              <https://www.rfc-editor.org/info/rfc7276>.

   [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A.,
              and M. Richardson, Ed., "A Security Threat Analysis for
              the Routing Protocol for Low-Power and Lossy Networks
              (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015,
              <https://www.rfc-editor.org/info/rfc7416>.

   [RFC7548] Ersue, M., Ed., Romascanu, D., Schoenwaelder, J., and A.
              Sehgal, "Management of Networks with Constrained Devices:
              Use Cases", RFC 7548, DOI 10.17487/RFC7548, May 2015,
              <https://www.rfc-editor.org/info/rfc7548>.

   [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>.

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

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

   Method 1: In the first method, we constructed a table measuring RSSI
      vs ETX using the Cooja simulation [cooja] setup in the Contiki OS
      environment[contiki]. We used Contiki-2.7 running 6LoWPAN/RPL
      protocol stack for the simulations. For approximating the number
      of packet drops based on the RSSI values, we implemented simple
      logic that drops transmitted packets with certain pre-defined
      ratios before handing over the packets to the receiver. The
      packet drop ratio is implemented as a table lookup of RSSI ranges
      mapping to different packet drop ratios with lower RSSI ranges
      resulting in higher values. While this table has been defined for
      the purpose of capturing the overall link behavior, it is highly
      recommended to conduct physical radio measurement experiments, in
      general. By keeping the receiving node at different distances, we
      let the packets experience different packet drops as per the
      described method. The ETX value computation is done by another
      module which is part of RPL Objective Function implementation.
      Since ETX value is reflective of the extent of packet drops, it
      allowed us to prepare a useful ETX vs RSSI table. ETX versus RSSI
      values obtained in this way may be used as explained below:

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

          Figure 7: Communication link from Source to Destination

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

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

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

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

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

Appendix B. Changelog

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

B.1.   Changes from version 13 to version 14

   *   Provided more details about scenarios naturally supporting the
       choice of AODV-RPL as a routing protocol

   *   Added new informative references [RFC6687], [RFC9010]) that
       describe the value provided by peer-to-peer routing.

   *   Requested IANA to allocate a new multicast group to enable clean
       separation of AODV-RPL operation from previous routing protocols
       in the RPL family.

   *   Cited [RFC6550] as the origination of the definition of DIO

   *   Defined "hop-by-hop route" as a route created using RPL's storing
       mode.

   *   Defined new configuration variable REJOIN_REENABLE.

   *   Improved definition for RREQ-InstanceID. Created analogous
       definition for RREP-InstanceID=(RPLInstanceID, TargNode_IPaddr)

   *   Improved definition of source routing

   *   Clarified that the Border Router (BR) in Figure 4 does not imply
       that AODV does not a require a BR as a protocol entity.

   *   Provided more guidelines about factors to be considered by
       OrigNode when selecting a value for the 'L' field.

   *   Described the disadvantage of not keeping track of the Address
       Vector in the RREQ-Instance.

   *   Specified that in non-storing mode an intermediate node has to
       record the IP addresses of both incoming and outgoing interfaces
       into the Address Vector, when those interfaces have different IP
       addresses.

   *   Added three informative references to describe relevant details
       about evaluating link assymetry.

   *   Clarified details about Gratuitous RREP.

B.2.   Changes from version 12 to version 13

   *   Changed name of "Shift" field to be the "Delta" field.

   * Specified that if a node does not have resources, it MUST drop the
      RREQ.

   * Changed name of MaxUseRank to MaxUsefulRank.

   * Revised a sentence that was not clear about when a TargNode can
      delay transmission of the RREP in response to a RREQ.

   * Provided advice about running AODV-RPL at same time as P2P-RPL or
      native RPL.

   * Small reorganization and enlargement of the description of Trickle
      time operation in Section 8.

   * Added definition for "RREQ-InstanceID" to Terminology section.

   * Specified that once a node leaves an RREQ-Instance, it MUST NOT
      rejoin the same RREQ-Instance.

B.3. Changes from version 11 to version 12

   * Defined RREP_WAIT_TIME for asymmetric as well as symmetric
      handling of RREP-DIO.

   * Clarifed link-local multicast transmission to use link-local
      multicast group all-RPL nodes.

   * Identified some security threats more explicitly.

   * Specified that the pairing between RREQ-DIO and RREP-DIO happens
      at OrigNode and TargNode. Intermediate routers do not necessarily
      maintain the pairing.

   * When RREQ-DIO is received with H=0 and S=1, specified that
      intermediate routers MAY store symmetric Address Vector
      information for possible use when a matchine RREP-DIO is received.

   * Specified that AODV-RPL uses the "P2P Route Discovery Mode of
      Operation" (MOP == 4), instead of requesting the allocation of a
      new MOP. Clarified that there is no conflict with [RFC6997].

   * Fixed several important typos and improved language in numerous
      places.

   * Reorganized the steps in the specification for handling RREQ and
      RREP at an intermediate router, to more closely follow the order
      of processing actions to be taken by the router.

B.4. Changes from version 10 to version 11

   * Numerous editorial improvements.

   * Replace Floor((7+(Prefix Length))/8) by Ceil(Prefix Length/8) for
      simplicity and ease of understanding.

   * Use "L field" instead of "L bit" since L is a two-bit field.

   * Improved the procedures in section 6.2.1.

   * Define the S bit of the data structure a router uses to represent
      whether or not the RREQ instance is for a symmetric or an
      asymmetric route. This replaces text in the document that was a
      holdover from earlier versions in which the RREP had an S bit for
      that purpose.

   * Quote terminology from AODV that has been identified as possibly
      originating in language reflecting various kinds of bias against
      certain cultures.

   * Clarified the relationship of AODV-RPL to RPL.

   * Eliminated the "Point-to-Point" terminology to avoid suggesting
      only a single link.

   * Modified certain passages to better reflect the possibility that a
      router might have multiple IP addresses.

   * "Rsv" replaced by "X X" for reserved field.

   * Added mandates for reserved fields, and replaces some ambiguous
      language phraseology by mandates.

   * Replaced "retransmit" terminology by more correct "propagate"
      terminology.

   * Added text about determining link symmetry near Figure 5.

   * Mandated checking the Address Vector to avoid routing loops.

   * Improved specification for use of the Delta value in
      Section 6.3.3.

   * Corrected the wrong use of RREQ-Instance to be RREP-Instance.

   * Referred to Subregistry values instead of Registry values in
      Section 9.

   * Sharpened language in Section 10, eliminated misleading use of
      capitalization in the words "Security Configuration".

   * Added acknowledgements and contributors.

B.5. Changes from version 09 to version 10

   * Changed the title for brevity and to remove acronyms.

   * Added "Note to the RFC Editor" in Section 9.

   * Expanded DAO and P2MP in Section 1.

   * Reclassified [RFC6998] and [RFC7416] as Informational.

   * SHOULD changed to MUST in Section 4.1 and Section 4.2.

   * Several editorial improvements and clarifications.

B.6. Changes from version 08 to version 09

   * Removed section "Link State Determination" and put some of the
      relevant material into Section 5.

   * Cited security section of [RFC6550] as part of the RREP-DIO
      message description in Section 2.

   * SHOULD has been changed to MUST in Section 4.2.

   * Expanded the terms ETX and RSSI in Section 5.

   * Section 6.4 has been expanded to provide a more precise
      explanation of the handling of route reply.

   * Added [RFC7416] in the Security Considerations (Section 10) for
      RPL security threats. Cited [RFC6550] for authenticated mode of
      operation.

   * Appendix A has been mostly re-written to describe methods to
      determine whether or not the S bit should be set to 1.

   * For consistency, adjusted several mandates from SHOULD to MUST and
      from SHOULD NOT to MUST NOT.

   * Numerous editorial improvements and clarifications.

B.7. Changes from version 07 to version 08

   * Instead of describing the need for routes to "fulfill the
      requirements", specify that routes need to "satisfy the Objective
      Function".

   * Removed all normative dependencies on [RFC6997]

   * Rewrote Section 10 to avoid duplication of language in cited
      specifications.

   * Added a new section "Link State Determination" with text and
      citations to more fully describe how implementations determine
      whether links are symmetric.

   * Modified text comparing AODV-RPL to other protocols to emphasize
      the need for AODV-RPL instead of the problems with the other
      protocols.

   * Clarified that AODV-RPL uses some of the base RPL specification
      but does not require an instance of RPL to run.

   * Improved capitalization, quotation, and spelling variations.

   * Specified behavior upon reception of a RREQ-DIO or RREP-DIO
      message for an already existing DODAGID (e.g, Section 6.4).

   * Fixed numerous language issues in IANA Considerations Section 9.

   * For consistency, adjusted several mandates from SHOULD to MUST and
      from SHOULD NOT to MUST NOT.

   * Numerous editorial improvements and clarifications.

B.8. Changes from version 06 to version 07

   * Added definitions for all fields of the ART option (see
      Section 4.3). Modified definition of Prefix Length to prohibit
      Prefix Length values greater than 127.

   * Modified the language from [RFC6550] Target Option definition so
      that the trailing zero bits of the Prefix Length are no longer
      described as "reserved".

   * Reclassified [RFC3561] and [RFC6998] as Informative.

   * Added citation for [RFC8174] to Terminology section.

B.9. Changes from version 05 to version 06

   * Added Security Considerations based on the security mechanisms
      defined in [RFC6550].

   * Clarified the nature of improvements due to P2P route discovery
      versus bidirectional asymmetric route discovery.

   * Editorial improvements and corrections.

B.10. Changes from version 04 to version 05

   * Add description for sequence number operations.

   * Extend the residence duration L in section 4.1.

   * Change AODV-RPL Target option to ART option.

B.11. Changes from version 03 to version 04

   * Updated RREP option format. Remove the T bit in RREP option.

   * Using the same RPLInstanceID for RREQ and RREP, no need to update
      [RFC6550].

   * Explanation of Delta field in RREP.

   * Multiple target options handling during transmission.

B.12. Changes from version 02 to version 03

   * Include the support for source routing.

   * Import some features from [RFC6997], e.g., choice between hop-by-
      hop and source routing, the L field which determines the duration
      of residence in the DAG, RankLimit, etc.

   * Define new target option for AODV-RPL, including the Destination
      Sequence Number in it. Move the TargNode address in RREQ option
      and the OrigNode address in RREP option into ADOV-RPL Target
      Option.

   * Support route discovery for multiple targets in one RREQ-DIO.

   * New RPLInstanceID pairing mechanism.

Appendix C. Contributors

   Abdur Rashid Sangi

Huaiyin Institute of Technology

No.89 North Beijing Road, Qinghe District

Huaian 223001

P.R. China

Email: sangi_bahrian@yahoo.com

Malati Hegde

Indian Institute of Science

Bangalore 560012

India

Email: malati@iisc.ac.in

Mingui Zhang

Huawei Technologies

No. 156 Beiqing Rd. Haidian District

Beijing 100095

P.R. China

Email: zhangmingui@huawei.com

Authors' Addresses

   Charles E. Perkins
   Lupin Lodge
   Los Gatos, 95033
   United States
   Email: charliep@lupinlodge.com

   S.V.R Anand
   Indian Institute of Science
   Bangalore 560012
   India
   Email: anandsvr@iisc.ac.in

     Satish Anamalamudi
     SRM University-AP
     Amaravati Campus
     Amaravati, Andhra Pradesh 522 502
     India
     Email: satishnaidu80@gmail.com

     Bing Liu
     Huawei Technologies
     No. 156 Beiqing Rd. Haidian District
     Beijing
     100095
     China
     Email: remy.liubing@huawei.com

                   Root initiated routing state in RPL
                    draft-ietf-roll-dao-projection-01

Abstract

   This document proposes a protocol extension to RPL that enables to
   install a limited amount of centrally-computed routes in a RPL graph,
   enabling loose source routing down a non-storing mode DODAG, or
   transversal routes inside the DODAG. As opposed to the classical
   route injection in RPL that are injected by the end devices, this
   draft enables the root of the DODAG to projects the routes that are
   needed on the nodes where they should be installed.

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

Table of Contents

1. Introduction

   The Routing Protocol for Low Power and Lossy Networks (LLN)(RPL)
   [RFC6550] is a generic Distance Vector protocol that is well suited
   for application in a variety of low energy Internet of Things (IoT)
   networks. 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 Internet. The root is responsible to select
   the RPL Instance that is used to forward a packet coming from the
   Internet into the RPL domain and set the related RPL information in
   the packets.

   The 6TiSCH architecture [I-D.ietf-6tisch-architecture] leverages RPL
   for its routing operation and considers the Deterministic Networking
   Architecture [I-D.ietf-detnet-architecture] as one possible model
   whereby the device resources and capabilities are exposed to an
   external controller which installs routing states into the network
   based on some objective functions that reside in that external
   entity.

Based on heuristics of usage, path length, and knowledge of device
capacity and available resources such as battery levels and
reservable buffers, a Path Computation Element ([PCE]) with a global
visibility on the system could install additional P2P routes that are
more optimized for the current needs as expressed by the objective
function.

This draft enables a RPL root, with optionally the assistance of a
PCE, to install and maintain additional storing and non-storing mode
routes within the RPL domain, along a selected set of nodes and for a
selected duration, thus providing routes more suitable than those
obtained with the distributed operation of RPL. Those routes may be
installed in either storing and non-storing modes RPL instances,
resulting in potentially hybrid situations where the mode of the
projected routes is different from that of the other routes in the
instance.

2. Terminology

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

The Terminology used in this document is consistent with and
incorporates that described in 'Terminology in Low power And Lossy
Networks' [RFC7102] and [RFC6550].

3. New RPL Control Message Options

Section 6.7 of [RFC6550] specifies Control Message Options (CMO) to
be placed in RPL messages such as the Destination Advertisement
Object (DAO) message. The RPL Target Option and the Transit
Information Option (TIO) are such options; the former indicates a
node to be reached and the latter specifies a parent that can be used
to reach that node. Options may be factorized; one or more
contiguous TIOs apply to the one or more contiguous Target options
that immediately precede the TIOs in the RPL message.

This specification introduces a new Control Message Option, the Via
Information option (VIO). Like the TIO, the VIO MUST be preceded by
one or more RPL Target options to which it applies. Unlike the TIO,
the VIO are not factorized: multiple contiguous Via options indicate
an ordered sequence of routers to reach the target(s), presented in
the order of the packet stream, source to destination, and in which a
routing state must be installed.

The Via Information option MUST contain at least one Via Address.

3.1. Via Information Option

   The Via Information option MAY be present in DAO messages, and its
   format is as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 0x0A  | Option Length | Path Sequence | Path Lifetime |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   .                                                               .
   .                        Via Address 1                          .
   .                                                               .
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   .                            ....                               .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   .                                                               .
   .                        Via Address n                          .
   .                                                               .
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
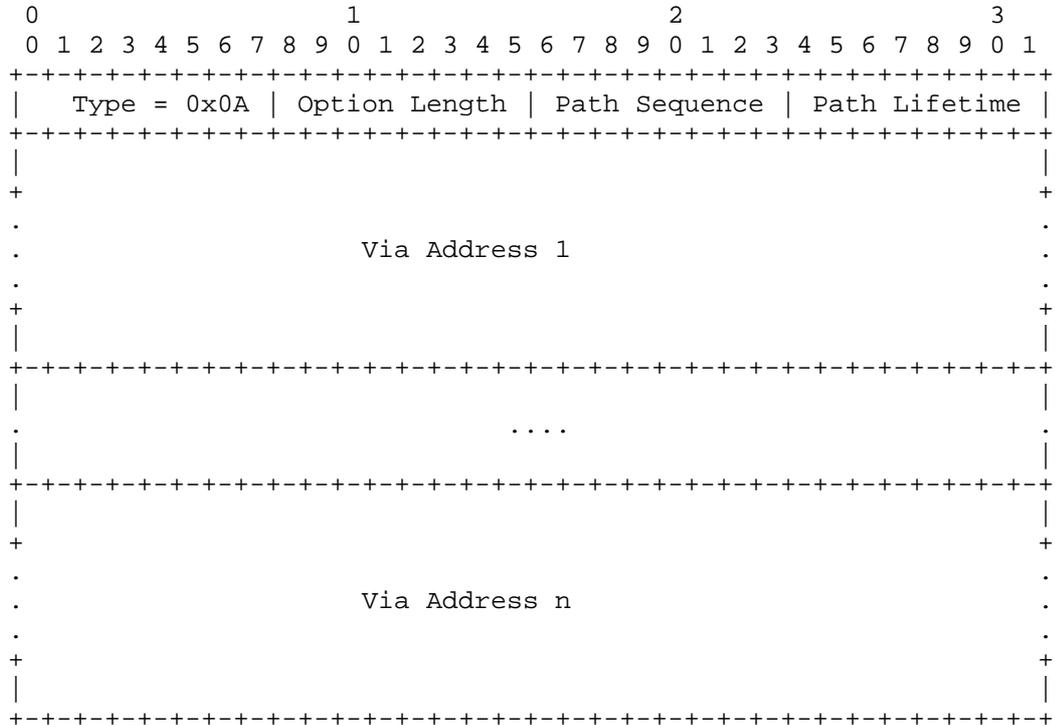
                 Figure 1: Via Information option format

   Option Type: 0x0A (to be confirmed by IANA)

   Option Length: In bytes; variable, depending on the number of Via
         Addresses.

   Path Sequence: 8-bit unsigned integer. When a RPL Target option is
         issued by the root of the DODAG (i.e. in a DAO message), that
         root sets the Path Sequence and increments the Path Sequence
         each time it issues a RPL Target option with updated
         information. The indicated sequence deprecates any state for a
         given Target that was learned from a previous sequence and adds
         to any state that was learned for that sequence.

Path Lifetime: 8-bit unsigned integer. The length of time in
     Lifetime Units (obtained from the Configuration option) that
     the prefix is valid for route determination. The period starts
     when a new Path Sequence is seen. A value of all one bits
     (0xFF) represents infinity. A value of all zero bits (0x00)
     indicates a loss of reachability. A DAO message that contains
     a Via Information option with a Path Lifetime of 0x00 for a
     Target is referred as a No-Path (for that Target) in this
     document.

Via Address: 16 bytes. IPv6 Address of the next hop towards the
     destination(s) indicated in the target option that immediately
     precede the VIO. TBD: See how the /64 prefix can be elided if
     it is the same as that of (all of) the target(s). In that
     case, the Next-Hop Address could be expressed as the 8-bytes
     suffix only, otherwise it is expressed as 16 bytes, at least in
     storing mode.

4. Projected DAO

This draft adds a capability to RPL whereby the root projects a route
through an extended DAO message called a Projected-DAO (P-DAO) to an
arbitrary router down the DODAG, indicating a next hop or a sequence
of routers via which a certain destination indicated in the Target
Information option may be reached.

A P-DAO message MUST contain at least a Target Information option and
at least one VIA Information option following it.

Like a classical DAO message, a P-DAO is processed only if it is
"new" per section 9.2.2. "Generation of DAO Messages" of the RPL
specification [RFC6550]; this is determined using the Path Sequence
information from the VIO as opposed to a TIO. Also, a Path Lifetime
of 0 in a VIO indicates that a route is to be removed.

There are two kinds of P-DAO, the storing mode and the non-storing
mode ones.

The non-storing mode P-DAO discussed in section Section 4.1 has a
single VIO with one or more Via Addresses in it, the list of Via
Addresses indicating the source-routed path to the target to be
installed in the router that receives the message, which replies
to the root directly with a DAO-ACK message.

The storing mode P-DAO discussed in section Section 4.2 has at
least two Via Information options with one Via Address each, for
the ingress and the egress of the path, and more if there are
intermediate routers. The Via Addresses indicate the routers in

which the routing state to the target have to be installed via the
next Via Address in the sequence of VIO. In normal operations,
the P-DAO is propagated along the chain of Via Routers from the
egress router of the path till the ingress one, which confirms the
installation to the root with a DAO-ACK message. Note that the
root may be the ingress and it may be the egress of the path, that
it can also be neither but it cannot be both.

The root is expected to use these mechanisms optimally and with
required parsimony to limit the state installed in the devices to fit
within their resources, but how the root figures the amount of
resources that is available in each device is out of scope for this
document.

In particular, the draft expects that the root has enough information
about the capability for each node to store a number of routes, which
can be discovered for instance using a Network Management System
(NMS) and/or the RPL routing extensions specified in Routing for Path
Calculation in LLNs [RFC6551].

A route that is installed by a P-DAO is not necessarily installed
along the DODAG, though how the root and the optional PCE obtain the
additional topological information to compute other routes is out of
scope for this document

4.1. Non-storing Mode Projected DAO

As illustrated in Figure 2, the non-storing mode P-DAO enables the
root to install a source-routed path towards a target in any
particular router; with this path information the router can add a
source routed header reflecting the path to any packet for which the
current destination either is the said target or can be reached via
the target, for instance a loose source routed packet for which the
next loose hop is the target, or a packet for which the router has a
routing state to the final destination via the target.

```
              ------+---------
                    |              Internet
                    |
          +-----+
          |     | Border Router
          |     | (RPL Root)
          +-----+                    |  P   ^              |
                    |                | DAO  | ACK          | Loose
              o    o    o    o    router  V   |           | Source
            o o   o  o   o  o  o      o  o     | P-DAO     . Route
           o  o o  o o   o   o   o   o  o      | Source    . Path
           o    o    o  o    o  o    o  o  o   | Route     . From
         o  o   o  o    o           o    o o   | Path      . Root
           o   o   o   o                 o     target  V   . To
            o        o                 o    o             | Desti-
          o          o              o     o               | nation
                                destination               V

                    LLN
```

Figure 2: Projecting a non-storing route

A router that receives a non-storing P-DAO installs a source routed
path towards each of the consecutive targets via a source route path
indicated in the following VIO.

When forwarding a packet to a destination for which the router
determines that routing happens via the target, the router inserts
the source routing header in the packet to reach the target.

In order to do so, the router encapsulates the packet with an IP in
IP header and a non-storing mode source routing header (SRH)
[RFC6554].

In the uncompressed form the source of the packet would be self, the
destination would be the first Via Address in the VIO, and the SRH
would contain the list of the remaining Via Addresses and then the
target.

In practice, the router will normally use the IPv6 over Low-Power
Wireless Personal Area Network (6LoWPAN) Paging Dispatch [RFC8025] to
compress the RPL artifacts as indicated in the 6LoWPAN Routing Header
[I-D.ietf-roll-routing-dispatch] specification. In that case, the
router indicates self as encapsulator in an IP-in-IP 6LoRH Header,
and places the list of Via Addresses in the order of the VIO and then
the target in the SRH 6LoRH Header.

4.2. Storing-Mode Projected DAO

   As illustrated in Figure 3, the storing mode P-DAO enables the root
   to install a routing state towards a target in the routers along a
   segment between an ingress and an egress router; this enables the
   routers to forward along that segment any packet for which the next
   loose hop is the said target, for instance a loose source routed
   packet for which the next loose hop is the target, or a packet for
   which the router has a routing state to the final destination via the
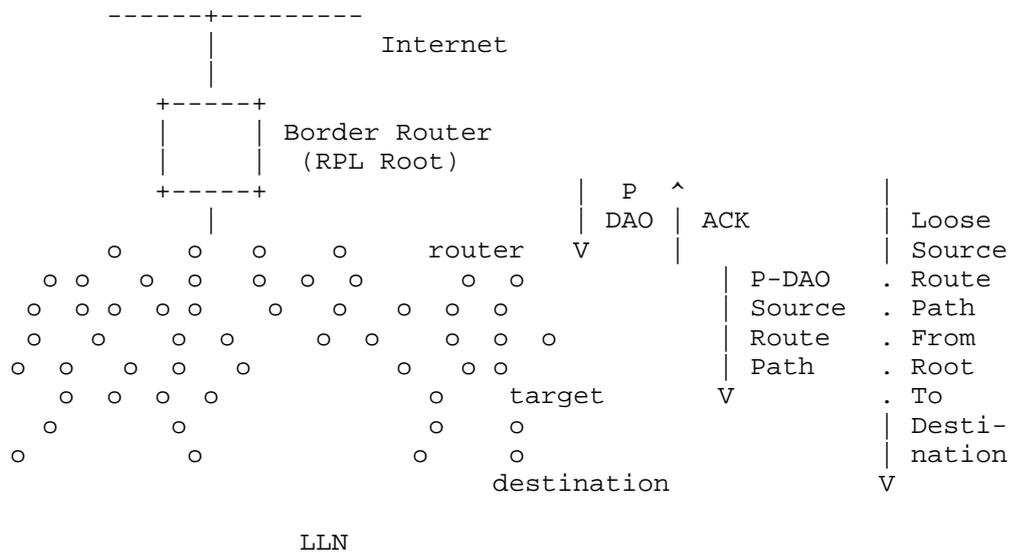   target.

```
           ------+---------
              |           Internet
              |
         +-----+
         |     | Border Router
         |     | (RPL Root)
         +-----+                           |     ^       |
              |                            | DAO | ACK   |
         o    o    o     o                 |     |       | Loose
          o o    o  o   o  o  o o     o     |  ^          | Source
         o  o o  o o    o   o   o   o       |  | DAO      | Route
         o   o     o  o     o  o    o  o  o |  ^          |
        o  o   o  o    o          o   o o   v | DAO        v
        o          o                 o    o

                        LLN
```
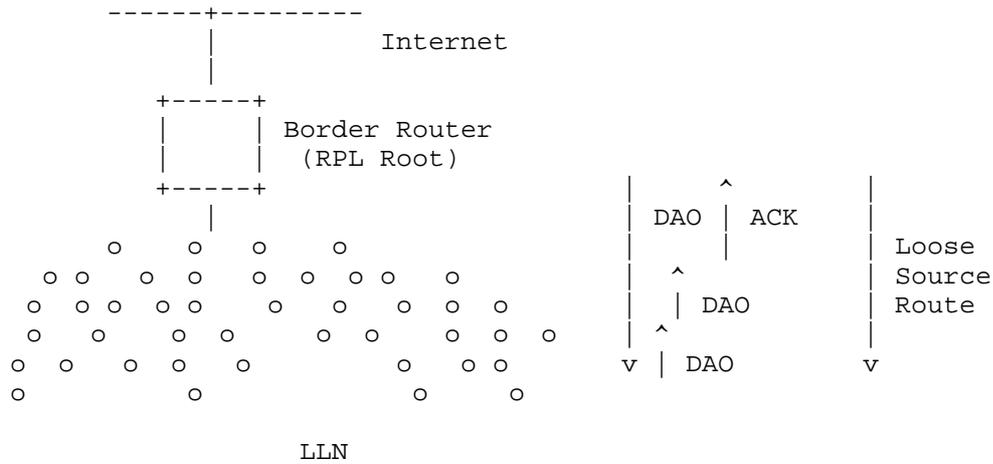
                    Figure 3: Projecting a route

   Based on available topological, usage and capabilities node
   information, the root or an associated PCE computes which segment
   should be optimized and which relevant state should be installed in
   which nodes. The algorithm is out of scope but it is envisaged that
   the root could compute the ratio between the optimal path (existing
   path not traversing the root, and the current path), the application
   service level agreement (SLA) for specific flows that could benefit
   from shorter paths, the energy wasted in the network, local
   congestion on various links that would benefit from having flows
   routed along alternate paths.

   In order to install the relevant routing state along the segment
   between an ingress and an egress routers, the root sends a unicast
   P-DAO message to the egress router of the routing segment that must
   be installed. The P-DAO message contains the ordered list of hops
   along the segment as a direct sequence of Via Information options
   that are preceded by one or more RPL Target options to which they

relate. Each Via Information option contains a Path Lifetime for
which the state is to be maintained.

The root sends the P-DAO directly to the egress node of the segment,
which In that P-DAO, the destination IP address matches the Via
Address in the last VIO. This is how the egress recognizes its role.
In a similar fashion, the ingress node recognizes its role as it
matches Via Address in the first 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) on its own. It may either be
the target, or may have some existing information to reach the
target(s), such as a connected route or an already installed
projected route. If one of the targets cannot be located, the node
MUST answer to the root with a negative DAO-ACK listing the target(s)
that could not be located (suggested status 10 to be confirmed by
IANA).

If the egress node can reach all the targets, then it forwards the
P-DAO with unchanged content to its loose 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 Information option the
precedes the one that contain the address of the propagating node,
which is used as source of the packet.

Upon receiving a propagated DAO, an intermediate router as well as
the ingress router install a route towards the DAO target(s) via its
successor in the P-DAO; the router locates the VIO that contains its
address, and uses as next hop the address found in the Via Address
field in the following VIO. The router MAY install additional routes
towards the addresses that are located in VIOs that are after the
next one, if any, but in case of a conflict or a lack of resource, a
route to a target installed by the root has precedence.

The process recurses till the P-DAO is propagated to ingress router
of the segment, which answers with a DAO-ACK to the root.

Also, the path indicated in a P-DAO may be loose, in which case the
reachability to the next hop has to be asserted. Each router along
the path indicated in a P-DAO is expected to be able to reach its
successor, either with a connected route (direct neighbor), or by
routing, for instance following a route installed previously by a DAO

or a P-DAO message. If that route is not connected then a recursive
lookup may take place at packet forwarding time to find the next hop
to reach the target(s). If it does not and cannot reach the next
router in the P-DAO, the router MUST answer to the root with a
negative DAO-ACK indicating the successor that is unreachable
(suggested status 11 to be confirmed by IANA).

A Path Lifetime of 0 in a Via Information option is used to clean up
the state. The P-DAO is forwarded as described above, but the DAO 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.

5. Applications

5.1. Loose Source Routing in Non-storing Mode

A RPL implementation operating in a very constrained LLN typically
uses the non-storing mode of operation whereby a RPL node indicates a
parent-child relationship to the root, using a Destination
Advertisement Object (DAO) that is unicast from the node directly to
the root, and the root typically builds a source routed path to a
destination down the DODAG by recursively concatenating this
information.

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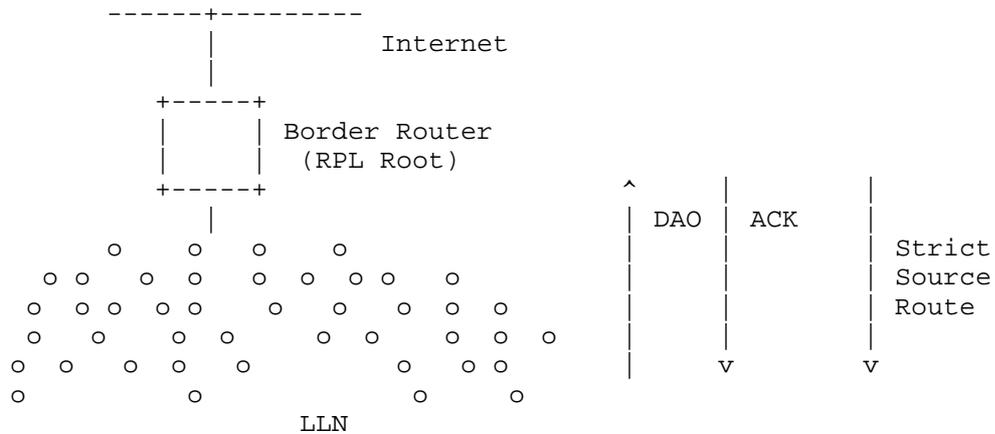```

                 Figure 4: RPL non-storing mode of operation

Based on the parent-children relationships expressed in the non-
storing DAO messages,the root possesses topological information about
the whole network, though this information is limited to the
structure of the DODAG for which it is the destination.  A packet
that is generated within the domain will always reach the root, which

can then apply a source routing information to reach the destination
if the destination is also in the DODAG.  Similarly, a packet coming
from the outside of the domain for a destination that is expected to
be in a RPL domain reaches the root.

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

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

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

5.2.  Transversal Routes in storing and non-storing modes

RPL is optimized for Point-to-Multipoint (P2MP), root to leaves and
Multipoint-to-Point (MP2P) leaves to root operations, whereby routes
are always installed along the RPL DODAG.  Transversal Peer to Peer
(P2P) routes in a RPL network will generally suffer from some stretch
since routing between 2 peers always happens via a common parent, as
illustrated in Figure 5:

o  in non-storing mode, all packets routed within the DODAG flow all
   the way up to the root of the DODAG.  If the destination is in the
   same DODAG, the root must encapsulate the packet to place a

Routing Header that has the strict source route information down the DODAG to the destination.  This will be the case even if the destination is relatively close to the source and the root is relatively far off.

o  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.

```
                     ------+---------
                     |               Internet
                     |
             +-----+
             |     | Border Router
             |     |  (RPL Root)
             +-----+
                X
         ^     v    o    o
       ^  o    o  v    o   o  o o    o
       ^   o o   o v     o    o    o   o
        ^    o    o  v      o   o    o  o   o
     S   o   o   o   D         o    o o
     o          o           o     o
                      LLN
```
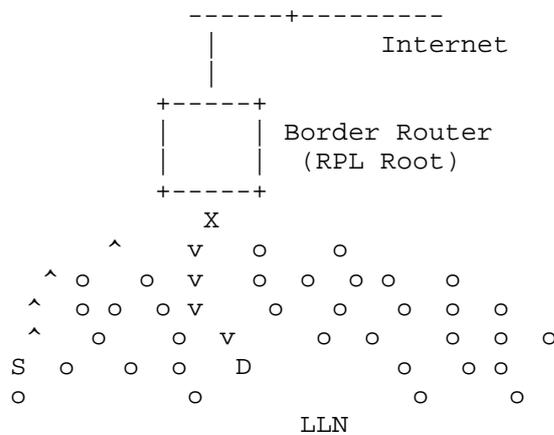
Figure 5: Routing Stretch between S and D via common parent X

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

```
          ------+---------
               |            Internet
               |
          +-----+
          |     |  Border Router
          |     |   (RPL Root)
          +-----+
               |
          o     o   o     o
        o o   o   o    o   o  o o    o
        o   o o   o o     o    o    o   o
        o    o    o  o      o  o    o  o   o
      S>>A>>>B>>C>>>D          o   o o
        o           o          o      o
                         LLN
```
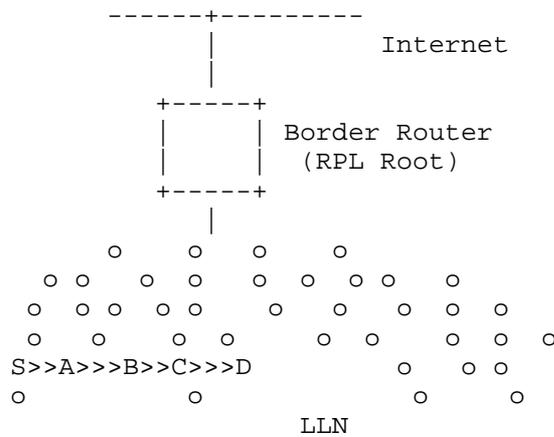
Figure 6: Projected Transversal Route

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

6. RPL Instances

It must be noted that RPL has a concept of instance but does not have
a concept of an administrative distance, which exists in certain
proprietary implementations to sort out conflicts between multiple
sources. This draft conforms the instance model as follows:

o if the PCE needs to influence a particular instance to add better
   routes in conformance with the routing objectives in that
   instance, it may do so. When the PCE modifies an existing
   instance then the added routes must not create a loop in that
   instance. This is achieved by always preferring a route obtained
   from the PCE over a route that is learned via RPL.

o If the PCE installs a more specific (Traffic Engineering) route
   between a particular pair of nodes then it should use a Local
   Instance from the ingress node of that path. Only packets
   associated with that instance will be routed along that path.

In all cases, the path is indicated by a new Via Information option, and the flow is similar to the flow used to obtain loose source routing.

7. Security Considerations

This draft uses messages that are already present in [RFC6550] with optional secured versions. The same secured versions may be used with this draft, and whatever security is deployed for a given network also applies to the flows in this draft.

8. IANA Considerations

This document updates the IANA registry for the Mode of Operation (MOP)

   4: Non-Storing with Projected routes [this]

This document updates IANA registry for the RPL Control Message Options

   0x0A: Via descriptor [this]

9. Acknowledgments

The authors wish to acknowledge JP Vasseur and Patrick Wetterwald for their contributions to the ideas developed here.

10. References

10.1. Normative References

[I-D.ietf-roll-routing-dispatch]
          Thubert, P., Bormann, C., Toutain, L., and R. Cragie,
          "6LoWPAN Routing Header", draft-ietf-roll-routing-
          dispatch-05 (work in progress), October 2016.

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

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J.,
          Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur,
          JP., and R. Alexander, "RPL: IPv6 Routing Protocol for
          Low-Power and Lossy Networks", RFC 6550,
          DOI 10.17487/RFC6550, March 2012,
          <http://www.rfc-editor.org/info/rfc6550>.

   [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N.,
              and D. Barthel, "Routing Metrics Used for Path Calculation
              in Low-Power and Lossy Networks", RFC 6551,
              DOI 10.17487/RFC6551, March 2012,
              <http://www.rfc-editor.org/info/rfc6551>.

   [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6
              Routing Header for Source Routes with the Routing Protocol
              for Low-Power and Lossy Networks (RPL)", RFC 6554,
              DOI 10.17487/RFC6554, March 2012,
              <http://www.rfc-editor.org/info/rfc6554>.

   [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,
              <http://www.rfc-editor.org/info/rfc8025>.

10.2. Informative References

   [I-D.ietf-6tisch-architecture]
              Thubert, P., "An Architecture for IPv6 over the TSCH mode
              of IEEE 802.15.4", draft-ietf-6tisch-architecture-11 (work
              in progress), January 2017.

   [I-D.ietf-detnet-architecture]
              Finn, N. and P. Thubert, "Deterministic Networking
              Architecture", draft-ietf-detnet-architecture-00 (work in
              progress), September 2016.

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

   [RFC6997] Goyal, M., Ed., Baccelli, E., Philipp, M., Brandt, A., and
              J. Martocci, "Reactive Discovery of Point-to-Point Routes
              in Low-Power and Lossy Networks", RFC 6997,
              DOI 10.17487/RFC6997, August 2013,
              <http://www.rfc-editor.org/info/rfc6997>.

   [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and
              Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January
              2014, <http://www.rfc-editor.org/info/rfc7102>.

Appendix A. Examples

A.1. Using storing mode P-DAO in non-storing mode MOP

   In non-storing mode, the DAG root maintains the knowledge of the
   whole DODAG topology, so when both the source and the destination of
   a packet are in the DODAG, the root can determine the common parent
   that would have been used in storing mode, and thus the list of nodes
   in the path between the common parent and the destination. For
   instance in the diagram shown in Figure 7, if the source is node 41
   and the destination is node 52, then the common parent is node 22.

```
                ------+---------
                     |            Internet
                     |
              +-----+
              |     |  Border Router
              |     |   (RPL Root)
              +-----+
               | \  \____      \
              /   \      \      \
             o 11   o 12    o  13
            /       |      /  \
           o 22      o 23   o 24  o 25
          /  \       | \      \
        o 31   o 32  o   o      o 35
       /       /     |    \     |   \
      o 41   o 42    o     o    o 45   o 46
      |       |      |      |    \      |
      o 51   o 52    o 53   o      o 55  o 56

                       LLN
```
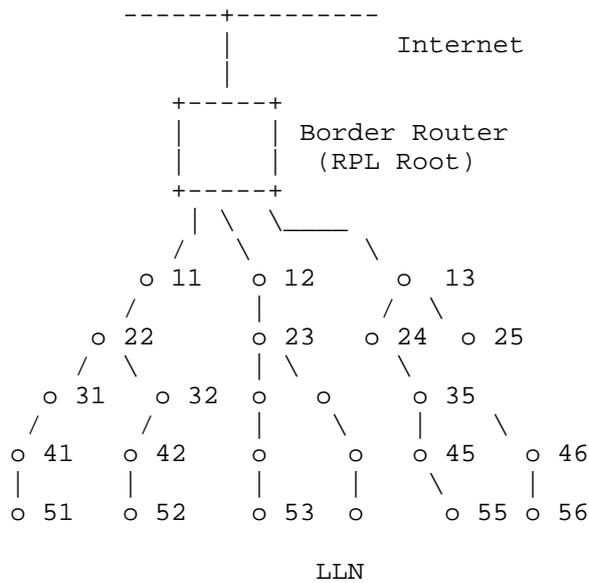
            Figure 7: Example DODAG forming a logical tree topology

   With this draft, the root can install a storing mode routing states
   along a segment that is either from itself to the destination, or
   from one or more common parents for a particular source/destination
   pair towards that destination (in this particular example, this would
   be the segment made of nodes 22, 32, 42).

   In the example below, say that there is a lot of traffic to nodes 55
   and 56 and the root decides to reduce the size of routing headers to
   those destinations. The root can first send a DAO to node 45
   indicating target 55 and a Via segment (35, 45), as well as another
   DAO to node 46 indicating target 56 and a Via segment (35, 46). This
   will save one entry in the routing header on both sides. The root
   may then send a DAO to node 35 indicating targets 55 and 56 a Via
   segment (13, 24, 35) to fully optimize that path.

Alternatively, the root may send a DAO to node 45 indicating target 55 and a Via segment (13, 24, 35, 45) and then a DAO to node 46 indicating target 56 and a Via segment (13, 24, 35, 46), indicating the same DAO Sequence.

A.2. Projecting a storing-mode transversal route

In this example, say that a PCE determines that a path must be installed between node S and node D via routers A, B and C, in order to serve the needs of a particular application.

The root sends a P-DAO with a target option indicating the destination D and a sequence Via Information option, one for S, which is the ingress router of the segment, one for A and then for B, which are an intermediate routers, and one for C, which is the egress router.

```
             ------+---------
                   |            Internet
                   |
           +-----+
           |     | Border Router
           |     |  (RPL Root)
           +-----+
                 | Projected DAO message to C
         o   |  o    o
       o o   o |   o  o o o   o
       o  o o  | o   o   o   o  o  o
       o   o  V  o     o  o    o  o  o
      S   A  B  C   D         o   o o
       o           o            o      o
                         LLN
```
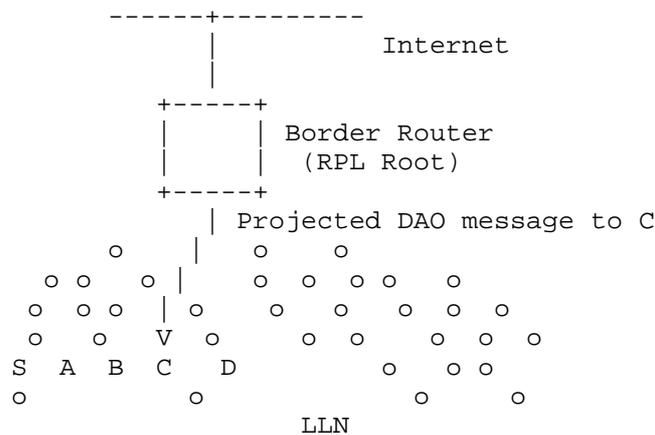
Figure 8: Projected DAO from root

Upon reception of the P-DAO, C validates that it can reach D, e.g. using IPv6 Neighbor Discovery, and if so, propagates the P-DAO unchanged to B.

B checks that it can reach C and of so, installs a route towards D via C. Then it propagates the P-DAO to A.

The process recurses till the P-DAO reaches S, the ingress of the segment, which installs a route to D via A and sends a DAO-ACK to the root.

```
              ------+---------
                    |               Internet
                    |
              +-----+
              |     | Border Router
              |     |  (RPL Root)
              +-----+
               ^ Projected DAO-ACK from S
             /    o    o    o
           /   o o    o  o  o o   o
          |  o o  o o    o   o   o o
          |   o    o  o     o  o    o  o  o
         S  A   B   C   D          o   o o
          o           o           o        o
                          LLN
```
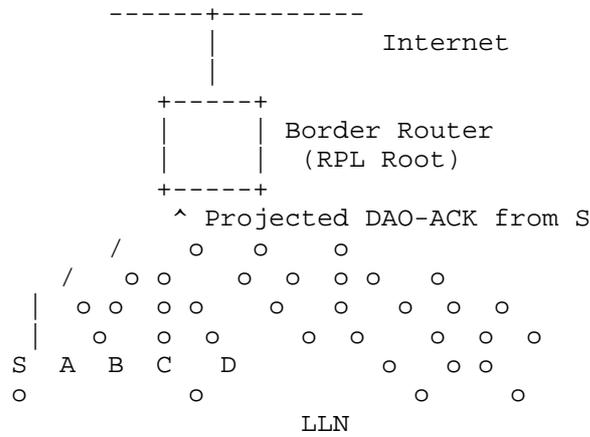
Figure 9: Projected DAO-ACK to root

As a result, a transversal route is installed that does not need to follow the DODAG structure.

```
              ------+---------
                    |               Internet
                    |
              +-----+
              |     | Border Router
              |     |  (RPL Root)
              +-----+
                    |
              o     o    o     o
           o o    o   o    o  o  o o    o
          o   o o  o o    o   o   o   o
          o    o    o  o      o  o    o  o  o
         S>>A>>>B>>C>>>D          o    o o
          o           o           o        o
                          LLN
```
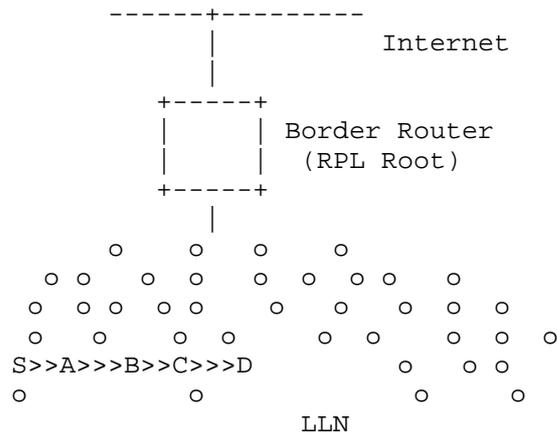
Figure 10: Projected Transversal Route

Authors' Addresses

   Pascal Thubert (editor)
   Cisco Systems
   Village d'Entreprises Green Side
   400, Avenue de Roumanille
   Batiment T3
   Biot - Sophia Antipolis   06410
   FRANCE

   Phone: +33 4 97 23 26 34
   Email: pthubert@cisco.com

   James Pylakutty
   Cisco Systems
   Cessna Business Park
   Kadubeesanahalli
   Marathalli ORR
   Bangalore, Karnataka   560087
   INDIA

   Phone: +91 80 4426 4140
   Email: mundenma@cisco.com

ROLL   P. Thubert, Ed.
Internet-Draft   Cisco Systems
Intended status: Standards Track   R.A. Jadhav
Expires: 10 June 2023   Huawei Tech
                                                        M. Richardson
                                                            Sandelman
                                                     7 December 2022

                       Root initiated routing state in RPL
                        draft-ietf-roll-dao-projection-30

Abstract

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

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

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

   RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) in
   which the Root often acts as the Border router to connect the RPL
   domain to the IP backbone. Routers inside the DODAG route along that
   graph up towards the Root for the default route and down towards
   destinations in the RPL domain for more specific routes. This
   specification expects as a pre-requisite a pre-existing RPL Instance
   with an associated DODAG and RPL Root, which are referred to as main
   Instance, main DODAG and main Root respectively. The main Instance
   is operated in RPL Non-Storing Mode of Operation (MOP).

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

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

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

2. Terminology

2.1. Requirements Language

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

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

2.2. References

   In this document, readers will encounter terms and concepts that are
   discussed in the "Routing Protocol for Low Power and Lossy Networks"
   [RPL], the "6TiSCH Architecture" [RFC9030], the "Deterministic
   Networking Architecture" [RFC8655], the "Reliable and Available
   Wireless (RAW) Architecture" [RAW-ARCHI], and "Terminology in Low
   power And Lossy Networks" [RFC7102]. Both architecture documents
   define the concept of Track in a compatible fashion. This documents
   only builds Tracks that are DODAGs, meaning that all links are
   oriented From Ingress to Egress. This specification also utilizes
   the terms Segment and Lane that are also defined in the RAW
   Architecture.

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

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

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

2.3. Glossary

This document often uses the following acronyms:

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
RIB: Routing Information Base, aka the routing table.
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 an SM-VIO or a NSM-VIO

2.4. Domain Terms

This specification uses the following terminology:

2.4.1. Projected Route

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

2.4.2. Projected DAO

A DAO message used to install a P-Route.

2.4.3. Path

   Quoting section 1.1.3 of [INT-ARCHI]:

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

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

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

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

2.4.4. Routing Stretch

   RPL is anisotropic, meaning that it is directional, or more exactly
   polar. RPL does not behave the same way "downwards" (root towards
   leaves) with _multicast_ DIO messages that form the DODAG and
   "upwards" (leaves towards root) with _unicast_ DAO messages that
   follow the DODAG. This is in contrast with traditional IGPs that
   operate the same way in all directions and are thus called isotropic.

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

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

2.4.5. Track

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

   Figure 1 illustrates the mapping of the DODAG with the generic
   concept of a Track, with the DODAG Root acting as Ingress for the
   Track, and the mapping of Lanes and Segments, and only East-West
   Segments, meaning that they are directional and progressing towards
   the destination.
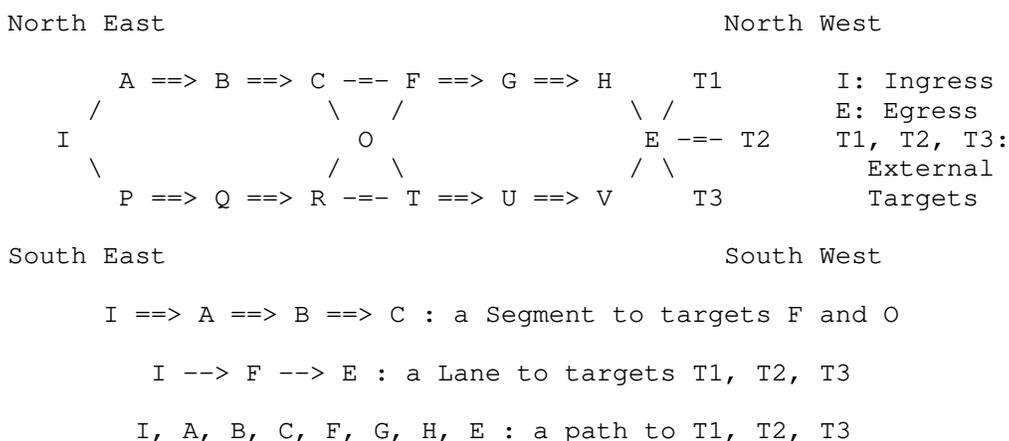
```
      North East                                   North West

         A ==> B ==> C -=- F ==> G ==> H     T1         I: Ingress
        /              \   /              \ /           E: Egress
      I                 O                  E -=- T2     T1, T2, T3:
        \              /   \              / \              External
         P ==> Q ==> R -=- T ==> U ==> V     T3         Targets

      South East                                   South West

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

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

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

                  Figure 1: A Track and its Components
```

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

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

2.4.5.1. TrackID

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

2.4.5.2. Namespace

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

2.4.5.3. Serial Track

   A Track that has only one path.

2.4.5.4. Complex Track

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

2.4.5.5. Stand-Alone

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

2.4.5.6. Stitching

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

2.4.5.7. Lane

   The concept of Lane is defined in the RAW Architecture" [RAW-ARCHI]
   as an end-to-end East-West serial path. With this specification, a
   Lane is installed by the Root of the main DODAG using a Non-Storing
   Mode P-DAO message.

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

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

2.4.5.8. 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 Lanes that form the structure of a
   complex Track. The same Segment may be leveraged by more than one
   Lane where the Lanes overlap.

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

2.4.5.8.1. Section of a Segment

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

2.4.5.8.2. Segment Routing and SRH

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

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

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

3. Context and Goal

3.1. RPL Applicability

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

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

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

   RPL instances operate as ships passing in the night, unbeknownst of
   one another. The RPL Root is responsible for selecting the RPL
   Instance that is used to forward a packet coming from the Backbone
   into the RPL domain and for setting the related RPL information in
   the packets. Each Instance creates its own routing table (RIB) in
   participating nodes, and the RIB associated to the instance must be
   used end to end in the RPL domain. To that effect, RPL tags the
   packets with the Instance ID in a Hop-by-Hop extension Header.
   6TiSCH leverages RPL for its distributed routing operations.

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

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

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

3.2. Multi-Topology Routing and Loop Avoidance

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

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

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

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

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

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

Routing in a multi-topology domain may cause loops unless strict
rules are applied. This specification defines two strict orders to
ensure loop avoidance when projected routes are used in a RPL domain,
one between forwarding methods and one between RPL Instances, seen as
routing topologies. The possible forwarding methods are to a direct
next hop, to an indirect neighbor via a common neighbor, along a
Segment, and along a Track. The methods are strictly ordered as
listed above. A forwarding method may leverage any of the lower
order ones, but never one with a higher order; for instance, when
forwarding a packet along a Segment, the router may use direct or
indirect neighbors but cannot use a Track. The lower order methods
have a strict precedence, so the router will always prefer a direct
neighbor over an indirect one, or a Segment within the current RPL
Instance vs. another Track. The second strict and partial order is
between RPL Instances. That order must be defined by the
administrator for his RPL domain and defines a DODAG of underlays
with the main Instance as Root. A RPL Instance may leverage another
as underlay if and only if that other Instance is higher in that
order, that is farther to the main Instance in that graph; as an
example, a packet that is being forwarded along the main Instance may
be encapsulated in any Track, but a packet that is forwarded along a
Track may never continue along the main Instance.

3.3. Requirements

3.3.1. Loose Source Routing

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

```
             +-----+
             |     | Border router
             |     |  (RPL Root)
             +-----+                     ^       |       |
                |                        | DAO   | ACK   |
           o     o    o     o            |       |       | Strict
         o o   o  o    o  o  o o    o     |       |       | Source
        o  o o  o o    o    o   o    o    |       |       | Route
        o   o    o  o     o   o    o  o   |       |       |
       o   o   o   o    o     o   o o     |       v       v
        o        o           o      o
                      LLN
```
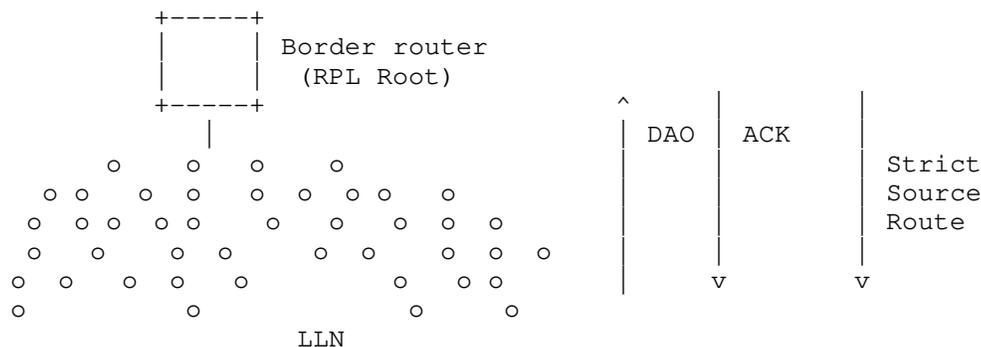
               Figure 2: RPL Non-Storing Mode of operation

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

   The RPL Root must add a source routing header to all downward
   packets. As a network grows, the size of the source routing header
   increases with the depth of the network. In some use cases, a RPL
   network forms long lines along physical structures such as streets
   for lighting. Limiting the packet size is 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 reduce the overhead of routing information in packets. Because
the capability to store routing state in every node is limited, the
decision of which route is installed where can only be optimized with
global knowledge of the system, knowledge that the Root or an
associated PCE may possess by means that are outside the scope of
this specification.

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

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

3.3.2. 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
experience elongated (stretched) paths versus direct (optimized)
paths, since routing between two nodes always happens via a common
parent, as illustrated in Figure 3:

```
            ------+---------
                 |             Internet
            +-----+
            |     | Border router
            |     |  (RPL Root)
            +-----+
              X
          ^     v   o      o
        ^ o   o  v   o  o  o o    o
        ^   o o  o v     o    o    o   o
        ^    o    o  v     o  o     o   o   o
      S  o    o  o    D          o   o o
      o          o           o     o
                        LLN
```
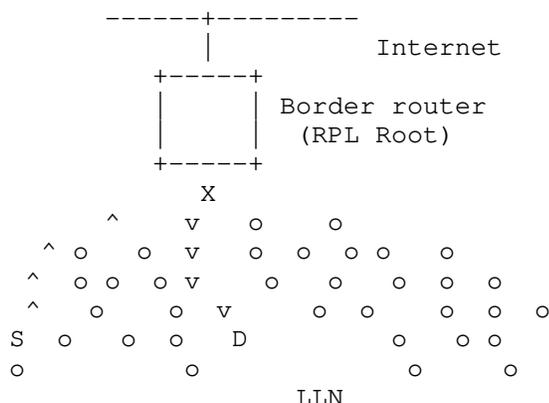
Figure 3: Routing Stretch between S and D via common parent X
along North-South Paths

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

*   in Non-Storing Mode, all packets routed within the DODAG flow all
    the way up to the Root of the DODAG. If the destination is in the
    same DODAG, the Root must encapsulate the packet to place an RH
    that has the strict source route information down the DODAG to the
    destination. This will be the case even if the destination is
    relatively close to the source and the Root is relatively far off.

*   In Storing Mode, unless the destination is a child of the source,
    the packets will follow the default route up the DODAG as well.
    If the destination is in the same DODAG, they will eventually
    reach a common parent that has a route to the destination; at
    worse, the common parent may also be the Root. From that common
    parent, the packet will follow a path down the DODAG that is
    optimized for the Objective Function that was used to build the
    DODAG.

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

```
                +-----+
                |     | Border router
                |     |   (RPL Root)
                +-----+
                   |
           o    o    o    o
         o o   o  o   o  o  o o    o
         o  o o  o o   o    o    o   o
         o   o    o  o     o  o   o  o  o
       S>>A>>>B>>C>>>D         o    o o
         o          o           o      o
                         LLN
```
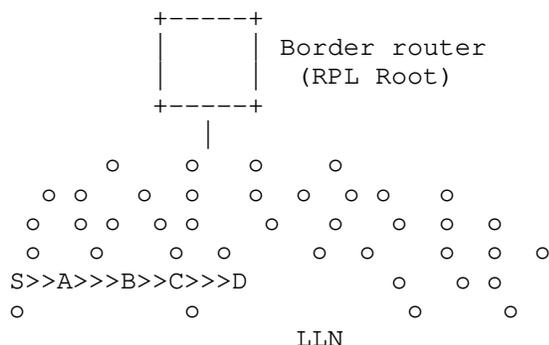
              Figure 4: More direct East-West Route between S and D

   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 Lane. The nodes that are directly connected,
   reachable via existing Tracks as illustrated in Section 3.5.2.3 or
   joined with strict Segments of other nodes as shown in
   Section 3.5.1.3. The Lanes are expressed in RPL Non-Storing Mode and
   require an encapsulation to add a Source Route Header, whereas the
   Segments are expressed in RPL Storing Mode.

A Serial Track provides only one path between Ingress and Egress. It
comprises at most one Lane. 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 Lanes that may be parallel or cross at certain points, or as a
more generic DODAG.

3.4.2. Tracks and RPL Instances

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

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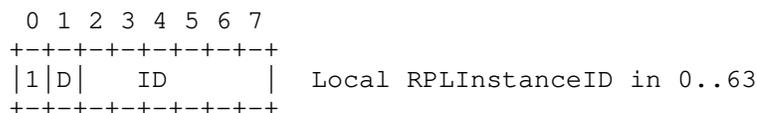
                   Figure 5: Local RPLInstanceID Encoding

A Track typically forms an underlay to the main Instance, and is
associated with a Local RPL Instance wherein the RPLInstanceID is
used as the TrackID; the encapsulating source IP address and RPI
Instance are set to the Track Ingress IP address and local
RPLInstanceID, respectively (more in Section 6.3).

A Track Lane may also be used as an alternative to a Segment in the
main DODAG, causing an extra encapsulation to signal a source-routed
path between loose hops in the encapsulated source routing header.
In that case, the TrackID in the outer header remains that of 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 the Track Lane is not a
Root of the instance, and the source address of the encapsulated
packet is not used to determine the Track.

3.5. Serial Track Signaling

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

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

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

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

   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:

```
                                        /===> F
                  A ===> B ===> C ===> D===> E <
                                        \===> 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 the Track Ingress and E is the Track Egress. C
   is a stitching point. F and G are "external" Targets for the Track,
   and become reachable from A via the Track A (Ingress) to E (Egress
   and implicit Target in Non-Storing Mode) leading to F and G (explicit
   Targets).

   In a general manner the desired outcome is as follows:

   * Targets are E, F, and G

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

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

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

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

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

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

3.5.1. Using Storing Mode Segments

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

3.5.1.1. Stitched Segments

   In this formulation:

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

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

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

```
+==================+=============+=============+
|      Field       | P-DAO 1 to E | P-DAO 2 to C |
+==================+=============+=============+
|      Mode        | Storing     | Storing     |
+------------------+-------------+-------------+
|  Track Ingress   | A           | A           |
+------------------+-------------+-------------+
| (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 originating at A going to F or G do not require encapsulation as the RPI can be placed in the native header chain. For packets that it routes, A must encapsulate to add the RPI that signals the trackID; the outer headers of the packets that are forwarded along the Track have the following settings:

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

Table 3: Packet Header Settings

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

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

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

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

3.5.1.2. External routes

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

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

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

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

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

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

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

Table 4: P-DAO Messages

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

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 if(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 and sends it down the
   Track signaled by P-DAO 3, with the outer header above. Now the
   packet destination is E.

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

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

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

3.5.1.3. Segment Routing

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

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

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

* P-DAO 3 signals F and G via the A-->C-->E-to-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 Lane. P-DAO 1 generates more signaling since E is the Segment
Egress when D could be, but has the benefit that it validates that
the connectivity between D and E still exists.

As a result the RIBs are set as follows:

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

Table 8: RIB setting

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

Header	IPv6 Source Addr.	IPv6 Dest. Addr.	TrackID in RPI
========	===================	===================	================
Outer	A	C until C then E	(A, 129)
Inner	X	E if(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 until D then E |   (C, 131)    |
+--------+------------------+------------------+---------------+
| Inner  |        X         |    E, F, or G    |      N/A      |
+--------+------------------+------------------+---------------+
```

Table 12: Packet Header Settings between C and E

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

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

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

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

3.5.2.2. External routes

In this formulation:

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

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

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

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

```
+==================+=============+=============+=============+
|                  | P-DAO 1 to C| P-DAO 2 to A| P-DAO 3 to A|
+==================+=============+=============+=============+
|      Mode        | Non-Storing | Non-Storing | Non-Storing |
+------------------+-------------+-------------+-------------+
|  Track Ingress   | C           | A           | A           |
+------------------+-------------+-------------+-------------+
| (DODAGID, TrackID)| (C, 131)   | (A, 129)    | (A, 141)    |
+------------------+-------------+-------------+-------------+
|    SegmentID     | 1           | 1           | 1           |
+------------------+-------------+-------------+-------------+
|      VIO         | D, E        | B, C        | E           |
+------------------+-------------+-------------+-------------+
|     Targets      | E           | 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 until D then E |    (C, 131)    |
+--------+------------------+------------------+----------------+
| Middle |        A         |        E         |    (A, 141)    |
+--------+------------------+------------------+----------------+
| Inner  |        X         |    E, F or G     |      N/A       |
+--------+------------------+------------------+----------------+
```

Table 15: Packet Header Settings

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

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

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

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

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

3.5.2.3. Segment Routing

In this formulation:

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

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

*   P-DAO 3 signals F and G via the A-->C-->E-to-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 until D then E |   (A, 129)     |
+--------+------------------+------------------+---------------+
| Middle |        A         |        C         |   (A, 141)     |
+--------+------------------+------------------+---------------+
| Inner  |        X         |    E, F or G     |     N/A        |
+--------+------------------+------------------+---------------+
```

Table 18: Packet Header Settings

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

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

Table 19: Packet Header Settings

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

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

Table 20: Packet Header Settings

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

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

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

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

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

3.6. Complex Tracks

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

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

```
        CPF             CPF           CPF                 CPF

               Southbound API

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

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

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

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

      <--- Segment 1 A,B ---> <- S5-> <---- Segment 4 J,E ---->
      <-      Lane 3            B,     H,           E ->
                                                          )
        (
                   (                                  )
```

                    Figure 7: Segments and Tracks

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

overlapping Lanes as illustrated in Figure 7, with Lane 3 that is
congruent with Lane 1 until node B and congruent with Lane 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 main DODAG is operated in RPL
Non-Storing Mode to sustain the exchanges with the Root. Based on
its comprehensive knowledge of the parent-child relationship, the
Root can form an abstracted view of the whole DODAG topology. This
document adds the capability for nodes to advertise additional
sibling information to complement the topological awareness of the
Root to be passed on to the PCE, and enable the PCE to build more /
better paths that traverse those siblings.

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

3.7.2. Positioning vs. Related IETF Standards

3.7.2.1. Extending 6TiSCH

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

3.7.2.2. Mapping to DetNet

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

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

3.7.2.3. Leveraging PCE

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

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

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

3.7.2.4. Providing for RAW

The RAW Architecture [RAW-ARCHI] extends the definition of Track, as
being composed of 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
Lanes, 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, in other words, 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
specifications. It is expected that extensions to existing
specifications do not cause existing code on legacy 6LRs to
malfunction, as the extensions will simply be ignored. New code is
required for an extension. Those 6LRs will be unable to participate
in the new mechanisms, but may also cause projected DAOs to be
impossible to install. Amendments to existing specifications are
situations where there are semantic changes required to existing
code, and which may require new unit tests to confirm that legacy
operations will continue unaffected.

4.1. Extending RFC 6550

This specification Extends RPL [RPL] to enable the Root to install
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 a Track Segment or a Track Lane,
respectively.

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

A P-DAO Request (PDR) message enables a Track Ingress to request the Track from the Root. The resulting Track is also a DODAG for which the Track Ingress is the Root, the owner the address that serves as DODAGID and authoritative for the associated namespace from which the TrackID is selected. In the context of this specification, the installed route appears as a more specific route to the Track Targets, and the Track Ingress 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 document Amends the specification of the DAO to create the P-DAO message. This Amended DAO is signaled with a new "Projected DAO" (P) flag, see Figure 8.

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

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

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TrackID     |K|D|P|  Flags  |     Reserved    | DAOSequence  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                    DODAGID field set to the                   |
+                 IPv6 Address of the Track Ingress             +
|                 used to source encapsulated packets           |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Option(s)...
+-+-+-+-+-+-+-+-+
```
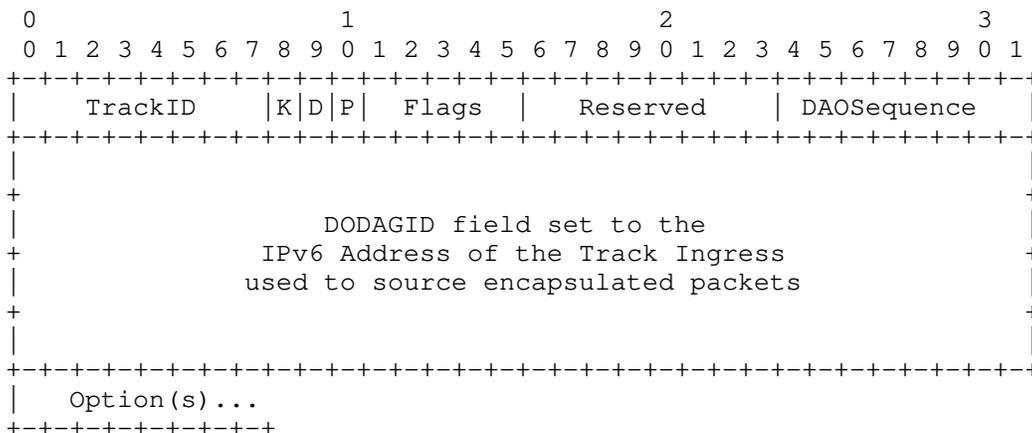
Figure 8: Projected DAO Base Object

New fields:

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

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

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

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

In RPL Non-Storing Mode, the TIO and RTO are combined in a DAO
message to inform the DODAG Root of all the edges in the DODAG, which
are formed by the directed parent-child relationships. The DAO

message signals to the Root that a given parent can be used to reach
a given child. The P-DAO message generalizes the DAO to signal to
the Track Ingress that a Track for which it is Root can be used to
reach children and siblings of the Track Egress. In both cases,
options may be factorized and multiple RTOs may be present to signal
a collection of children that can be reached through the parent or
the Track, respectively.

4.1.2. Projected DAO-ACK

This document also Amends the DAO-ACK message. The new P flag
signals the projected form.

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     TrackID    |D|P| Reserved  |   DAOSequence |     Status    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                     DODAGID field set to the                  |
+                  IPv6 Address of the Track Ingress            +
|                  used to source encapsulated packets          |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option(s)...
+-+-+-+-+-+-+-+-+
```
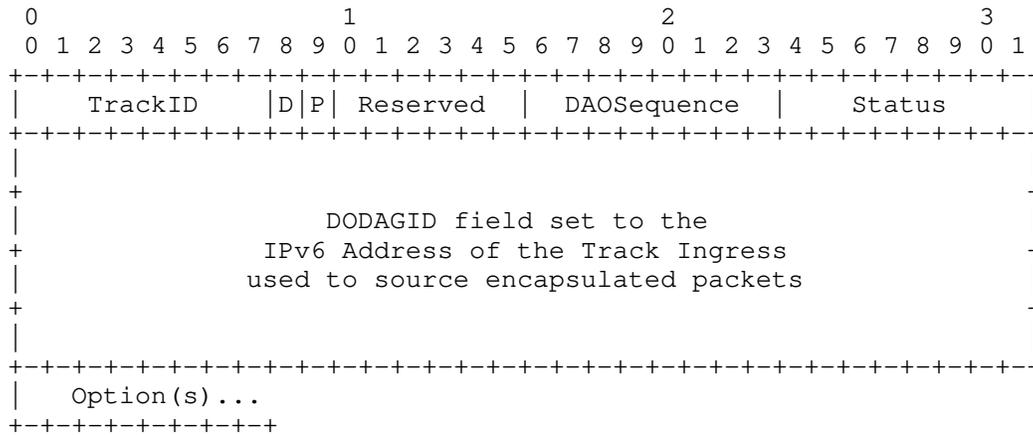
Figure 9: Projected DAO-ACK Base Object

New fields:

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

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

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

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

4.1.3. Via Information Option

This document Extends the 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 Lane at the Track Ingress, which uses that state to encapsulate a packet IPv6_in_IPv6 with a new Routing Header (RH) to the Track Egress (more in Section 6.7).

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

4.1.4. Sibling Information Option

This specification Extends the CMO to create the Sibling Information Option (SIO). The SIO is used by a RPL Aware Node (RAN) to advertise a selection of its candidate neighbors as siblings to the Root (more in Section 5.4). The SIO is placed in DAO messages that are sent directly to the main Root, including multicast DAO (see section 9.10 of [RPL]).

This draft AMENDS the multicast DAO operation as follows:

1. A multicast DAO message MUST be used only to advertise information about the node (using the Target Option), and direct Link Neighbors such as learned by Neighbor Discovery (using the Sibling Information Option).

2. The multicast DAO may be used to enable direct and indirect (via a common neighbor) P2P communication without needing the DODAG to relay the packets. The multicast DAO exposes the sender's addresses as Targets in RTOs and the sender's neighbors addresses as siblings in SIOs.

4.1.5. P-DAO Request

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

4.1.6. Amending the RPI

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

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

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

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

4.1.7. Additional Flag in the RPL DODAG Configuration Option

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

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Type = 0x04 |Opt Length = 14|D| | | |A|        ...         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                   +
                                       |4 bits |
```
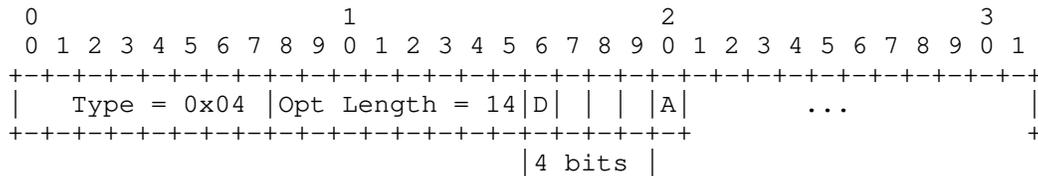
                Figure 10: DODAG Configuration Option (Partial View)

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

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

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

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

[RPL] states that:

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

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

4.2. Extending RFC 6553

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

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

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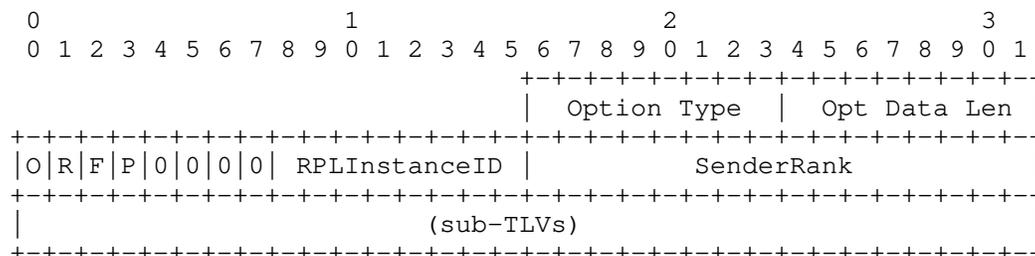follows:

```
      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:

```
            0                   1                   2
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |1|0|E| Length  |   6LoRH Type  | RPLInstanceID |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
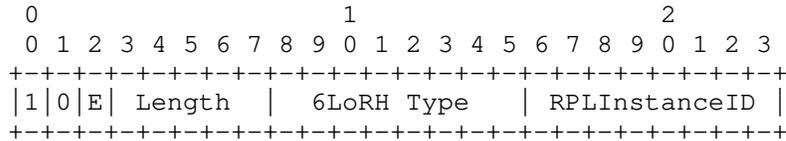
Figure 12: P-RPI-6LoRH Format

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

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

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

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

5. New RPL Control Messages and Options

5.1. New P-DAO Request Control Message

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

   The main Root MAY indicate to the Track Ingress that the Track was
   terminated before its time and to do so, it MUST use an asynchronous
   PDR-ACK with a negative status. A status of "Transient Failure" (see
   Section 11.10) is an indication that the PDR may be retried after a
   reasonable time that depends on the deployment. Other negative
   status values indicate a permanent error; the attempt must be
   abandoned until a corrective action is taken at the application layer
   or through network management.

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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    TrackID    |K|R|  Flags  |  ReqLifetime  |  PDRSequence   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Option(s)...
   +-+-+-+-+-+-+-+-+
```
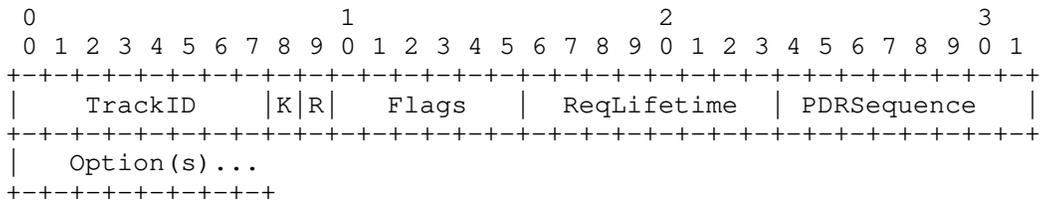
                   Figure 13: New P-DAO Request Format

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

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

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

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

ReqLifetime: 8-bit unsigned integer. The requested lifetime for the
   Track expressed in Lifetime Units (obtained from the DODAG
   Configuration option). The value of 255 (0xFF) represents
   infinity.

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

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

5.2. New PDR-ACK Control Message

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     TrackID    |     Flags     | Track Lifetime| PDRSequence   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PDR-ACK Status|                 Reserved                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option(s)...
+-+-+-+-+-+-+-+
```
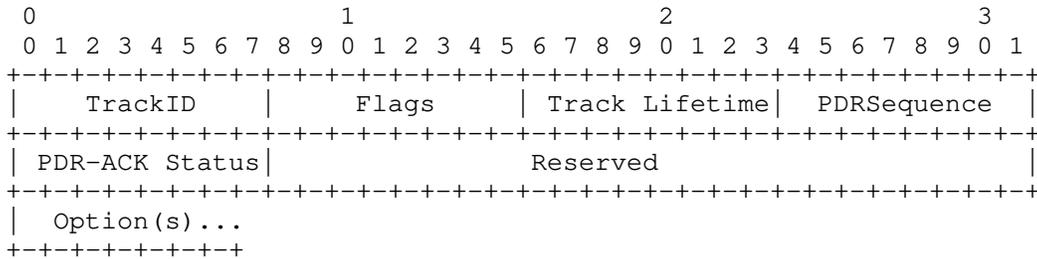
Figure 14: New PDR-ACK Control Message Format

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

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

Track Lifetime: Indicates the remaining Lifetime for the Track,
   expressed in Lifetime Units; The value of 255 (0xFF) represents
   infinity. The value of zero (0x00) indicates that the Track was
   destroyed or not created.

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

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

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

                  Figure 15: PDR-ACK status Format

   E: 1-bit flag. Set to indicate a rejection. When not set, the
      value of 0 indicates Success/Unqualified Acceptance and other
      values indicate "not an outright rejection".
   R: 1-bit flag. Reserved, MUST be set to 0 by the sender and
      ignored by the receiver.
   Status Value: 6-bit unsigned integer. Values depending on the
      setting of the 'E' flag, see Table 28 and Table 29.

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

5.3. Via Information Options

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

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

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

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

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

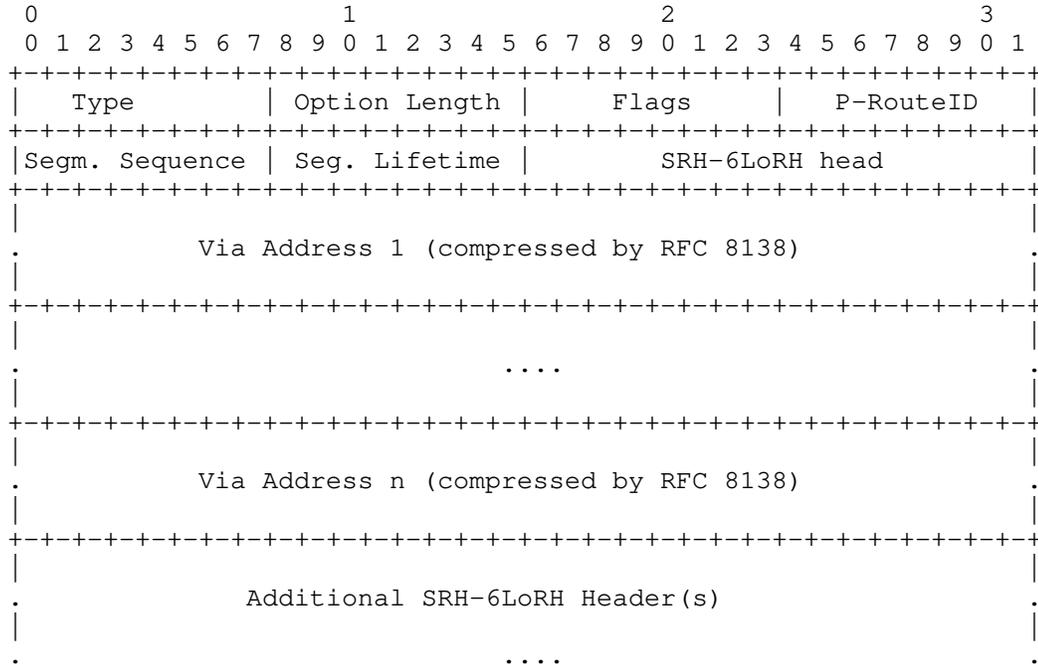The format of the Via Information Options is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type        | Option Length |     Flags     |   P-RouteID   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Segm. Sequence | Seg. Lifetime |          SRH-6LoRH head       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.              Via Address 1 (compressed by RFC 8138)           .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                             ....                              .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.              Via Address n (compressed by RFC 8138)           .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.               Additional SRH-6LoRH Header(s)                  .
|                                                               |
.                             ....                              .
```

Figure 16: VIO format

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

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

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/
Lane. In an SM-VIO, the P-RouteID indicates an actual Segment.
In an NSM-VIO, it indicates a Lane, that is a serial Track that is
added to the overall topology of the Track.

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

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

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

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

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

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

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

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

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

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

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

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

5.4. Sibling Information Option

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

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

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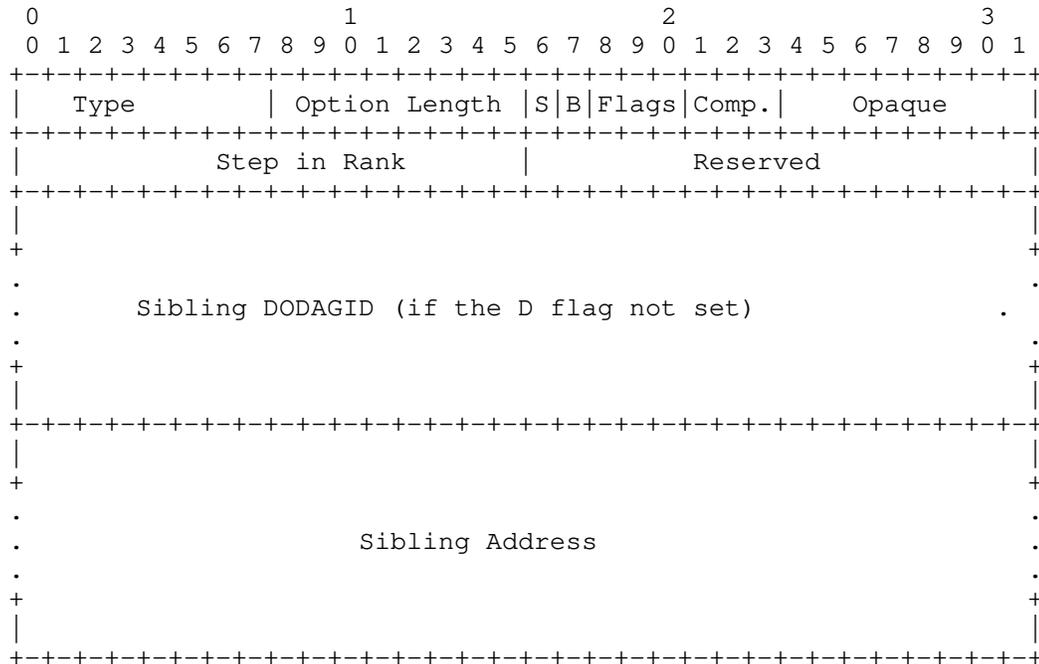
The format of the SIO is as follows:

```
            0                   1                   2                   3
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |     Type      |  Option Length |S|B|Flags|Comp.|    Opaque    |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |        Step in Rank           |           Reserved            |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                                                               |
           +                                                               +
           .                                                               .
           .        Sibling DODAGID (if the D flag not set)                .
           .                                                               .
           +                                                               +
           |                                                               |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |                                                               |
           +                                                               +
           .                                                               .
           .                     Sibling Address                          .
           .                                                               .
           +                                                               +
           |                                                               |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

               Figure 17: Sibling Information Option Format

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

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

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

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

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

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

Opaque: MAY be used to carry information that the node and the Root
    understand, e.g., a particular representation of the Link
    properties such as a proprietary Link Quality Information for
    packets received from the sibling. 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 present. The Compression reference is the Root of
    the main DODAG.

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

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

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

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

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

6. Root Initiated Routing State

6.1. RPL Network Setup

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

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

6.2. Requesting a Track

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

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

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

When the Track Lifetime that was returned in the PDR-ACK is close to
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 main Root MAY use P-DAO messages to add better routes in the
      main Instance in conformance with the routing objectives in that
      Instance.

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

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

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

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

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

The Track Ingress is the Root of the DODAG ID formed by the local RPL Instance. It owns the namespace of its TrackIDs, so it can pick any unused value to request a new Track with a PDR. In a particular deployment where PDRs are not used, a portion of the namespace can be administratively delegated to the main Root, meaning that the main Root is authoritative for assigning the TrackIDs for the Tracks it creates.

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

6.4. Installing a Track

A 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-Lane Track. A single-Lane Track can be installed as a loose Non-Storing Mode P-Route, in which case the next loose entry must recursively be reached over a 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 Lane of the same Track. In the latter case, the Targets of the P-Route must include the next hop of the Lane if there is one, to ensure forwarding continuity. In the case of a Complex Track, each Segment is maintained independently and asynchronously by the Root, with its own lifetime that may be shorter, the same, or longer than that of the Track.

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

*   Longest match MUST be the prime comparison for routing.

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

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

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

6.4.1. Signaling a Projected Route

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

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

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

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

In Non-Storing Mode, as discussed in Section 6.4.3, the Root sends the P-DAO to the Track Ingress where the source-routing state is applied, whereas in Storing Mode, the P-DAO is sent to the last node

on the installed path and forwarded in the reverse direction, installing a Storing Mode state at each hop, as discussed in Section 6.4.2. In both cases the Track Ingress is the owner of the Track, and it generates the P-DAO-ACK when the installation is successful.

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

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

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

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

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

As illustrated in Figure 18, a Storing Mode P-DAO installs a route along the Segment signaled by the SM-VIO towards the Targets indicated in the Target Options. The Segment is to be included in a DODAG indicated by the P-DAO Base Object, that may be the one formed by the main DODAG, or a Track associated with a local RPL Instance.
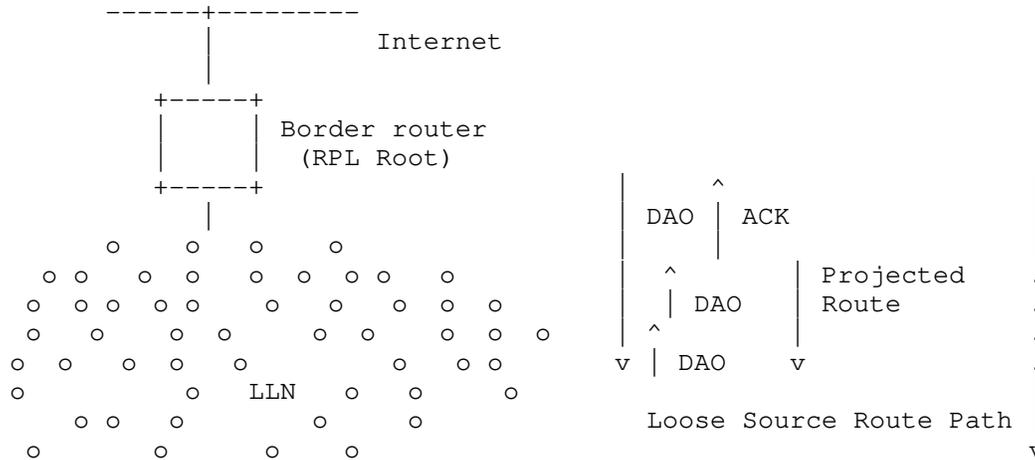
```
        ------+---------
              |               Internet
              |
      +-----+
      |     |  Border router
      |     |   (RPL Root)
      +-----+                       |        ^           |
         |                          | DAO    | ACK        |
       o     o   o     o            |        |            |
      o o   o o   o o  o o   o       |     ^         |  Projected    .
      o  o o  o o    o   o   o  o    |     | DAO     |  Route        .
      o   o    o o     o  o   o o  o |     ^         |               .
     o  o  o o   o       o   o o    v |  DAO    v    |               .
      o        o   LLN  o   o    o     |                             |
       o o   o      o      o      o    Loose Source Route Path |
     o      o     o    o                                        v
```

                    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
because it matches the 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 list, at the
position preceeding the one that contains the address of the
propagating node, which is used as source of the message.

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

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

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

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

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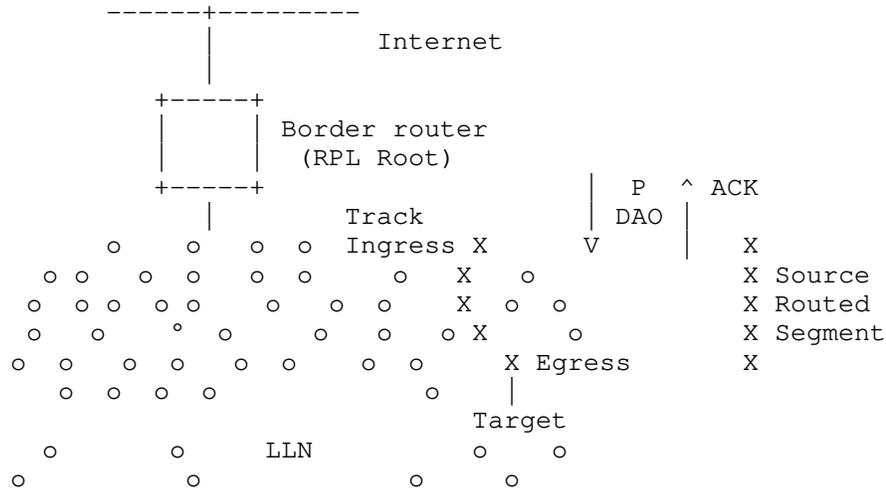forward it over the Track.

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

                Figure 19: Projecting a Non-Storing Route

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

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

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

6.5. Tearing Down a P-Route

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

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

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

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

6.6. Maintaining a Track

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

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

6.6.1. Maintaining a Track Segment

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

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

   After a reasonable time to enable the deprecated sections to drain
   their traffic, the Root tears down the remaining section(s) of the
   old Segments as described in Section 6.5.

6.6.2. Maintaining a Track Lane

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

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

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

6.7. Encapsulating and Forwarding Along a Track

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

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

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

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

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

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

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

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

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

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

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

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

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

  - So the Track Ingress must encapsulate the traffic that it did not originate, and 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 Segment must be strict and a packet that it placed in a Storing Mode Segment MUST follow that Segment till the Segment Egress.

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

* When forwarding along a Segment, if the next strict hop in the RIB for the destination of the packet is a direct Neighbor, the packet MUST be forwarded to that neighbor. Otherwise the packet MUST be dropped.

* When forwarding along a Track, if the next hop in the source route header is a direct Neighbor, the packet MUST be forwarded to that neighbor; else, the packet can be forwarded in the following case:

1. the previous next hop has a common Neighbor that can relay to
    the next loose hop (e.g., learned through a SIO in a multicast
    DAO message, see Section 4.1.4), in which case the packet MUST
    be forwarded to that neighbor,

2. else the previous next hop is the Ingress of a Segment of the
    same Track for which the loose next hop is a target, in which
    case the Segment MUST be used,

3. else the previous next hop is the Ingress of a (nested) Track
    to the loose next hop, in which case the (nested) Track MUST
    be used; another encapsulation takes place and the process may
    recurse.

Otherwise the packet MUST be dropped to avoid loops; as an
example, forwarding the packet along the main DODAG is disallowed
since it may cause such a loop.

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 to one or more nodes
being removed (See Section 6.6).

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

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

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

6.8. Compression of the RPL Artifacts

When using [RFC8138] in the main DODAG operated in Non-Storing Mode
in a 6LoWPAN LLN, a typical packet that circulates in the main DODAG
is formatted as shown in Figure 20, representing the case where an
IPv6-in-IPv6 encapsulation is needed (see Table 19 of [RFC9008]):

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

Figure 20: A Packet as Forwarded along the main DODAG

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

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

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

                    Figure 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 Lane.

```
    0                   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5  ..         ..          ..
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-    -+-    -+ ... +-    -+
   |1|0|0|  Size    |6LoRH Type 0..4| Hop1 | Hop2 |     | HopN |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-    -+-    -+ ... +-    -+
                                              Where N = Size + 1
```
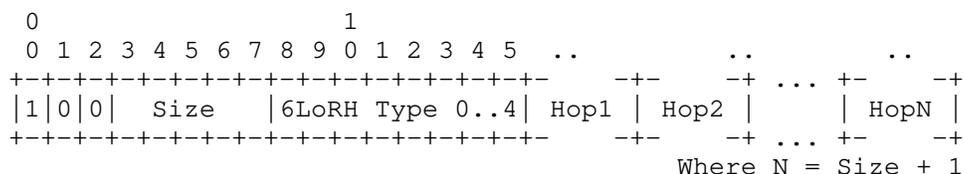
                    Figure 22: The SRH 6LoRH Header

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

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

    Signals : Loose Hops :   TrackID  : Track DODAGID :
```
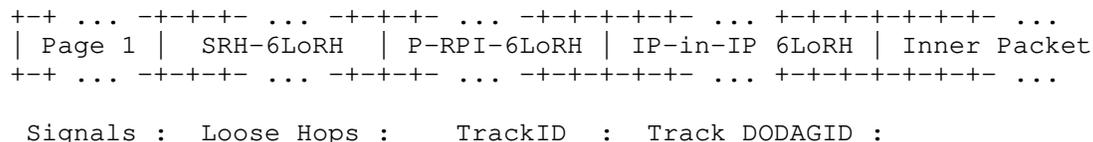
                  Figure 23: A Packet as Forwarded along a Track

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

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

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

On the other hand, since RPL is a Layer-3 routing protocol, its applicability extends beyond LLNs to a generic IP network. RPL requires less resources than alternative IGPs like OSPF, ISIS, EIGRP, BABEL or RIP at the expense of a route stretch vs. the shortest path routes to a destination that those protocols compute. P-Routes add the capability to install shortest and/or constrained routes to special destinations such as discussed in section A.9.4. of the ANIMA ACP [RFC8994].

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

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

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

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

The DAO message MUST be formed as follows:

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

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

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

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

7.2. A Track as a Full DODAG

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

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

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

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

   The RAW Architecture [RAW-ARCHI] actually expects the 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 needs the full richness of a
   DODAG to form any path that could meet the Service Level Objective
   (SLO).

   This section specifies the additions that are needed to turn the
   Track into a full DODAG and enable the main Root to provide the
   necessary topological information to the Track Ingress. The
   expectation is that the metrics that the 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 document provides a set of tools that may or may not be needed
   by an implementation depending on the type of application it serves.
   This sections described profiles that can be implemented separately
   and can be used to discriminate what an implementation can and cannot
   do. This section describes profiles that enable implementing only a
   portion of this specification to meet a particular use case.

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

Profiles 0 and 1 are REQUIRED by all implementations that may be used
in LLNs; Profile 1 leverages Storing Mode to reduce the size of the
Source Route Header in the most common LLN deployments. Profile 2 is
RECOMMENDED in high speed / wired environment to enable traffic
Engineering and network automation. All the other profile /
environment combinations are OPTIONAL.

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

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

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

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

Profile 4 Profile 4 extends Profile 2 with Strict Source-Routing
   Non-Storing Mode P-Routes to form 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 routed paths between the Root
   that is the Track Ingress, and the Track Egress that is the last
   node.

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

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

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

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

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

9. Backwards Compatibility

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

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

10. Security Considerations

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

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

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

With this specification, only the Root may generate P-DAO messages.
PDR messages may only be sent to the Root. This specification
expects that the communication with the Root is authenticated but
does not enforce which method is used.

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

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

11. IANA Considerations

11.1. RPL DODAG Configuration Option Flag

   IANA is requested to assign a flag from the "DODAG Configuration
   Option Flags for MOP 0..6" [RFC9010] registry under the heading
   "Routing Protocol for Low Power and Lossy Networks (RPL)" as follows:

```
+---------------+------------------------------+-----------+
| Bit Number    | Capability Description        | Reference |
+---------------+------------------------------+-----------+
| 0 (suggested) | Projected Routes Support (D)  | THIS RFC  |
+---------------+------------------------------+-----------+
```

           Table 21: New DODAG Configuration Option Flag

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

11.2. Elective 6LoWPAN Routing Header Type

   IANA is requested to update the "Elective 6LoWPAN Routing Header
   Type" registry that was created for [RFC8138] under the heading
   "Routing Protocol for Low Power and Lossy Networks (RPL)" and assign
   the following value:

```
+===============+=============+===========+
|     Value     | Description | Reference |
+===============+=============+===========+
| 8 (Suggested) | P-RPI-6LoRH | THIS RFC  |
+---------------+-------------+-----------+
```

                Table 22: New Elective 6LoWPAN Routing
                            Header Type

11.3. Critical 6LoWPAN Routing Header Type

   IANA is requested to update the "Critical 6LoWPAN Routing Header
   Type" registry that was created for [RFC8138] under the heading
   "Routing Protocol for Low Power and Lossy Networks (RPL)" and assign
   the following value:

```
+===============+==============+===========+
|     Value     | Description  | Reference |
+===============+==============+===========+
| 8 (Suggested) | P-RPI-6LoRH  | THIS RFC  |
+---------------+--------------+-----------+
```

Table 23: New Critical 6LoWPAN Routing
Header Type

11.4. Registry For The RPL Option Flags

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

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

*   Indication When Set

*   Reference

Registration procedure is "Standards Action" [RFC8126]. The initial
allocation is as indicated in Table 24:

```
+===============+=====================+===========+
|  Bit number   | Indication When Set | Reference |
+===============+=====================+===========+
|       0       | Down 'O'            | [RFC6553] |
+---------------+---------------------+-----------+
|       1       | Rank-Error (R)      | [RFC6553] |
+---------------+---------------------+-----------+
|       2       | Forwarding-Error (F)| [RFC6553] |
+---------------+---------------------+-----------+
| 3 (Suggested) | Projected-Route (P) | THIS RFC  |
+---------------+---------------------+-----------+
|     4..255    | Unassigned          |           |
+---------------+---------------------+-----------+
```

Table 24: Initial PDR Flags

11.5. RPL Control Codes

IANA is requested to update the "RPL Control Codes" registry under
the heading "Routing Protocol for Low Power and Lossy Networks (RPL)"
as indicated in Table 25:

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

               Table 25: New RPL Control Codes

11.6. RPL Control Message Options

   IANA is requested to update the "RPL Control Message Options"
   registry under the heading "Routing Protocol for Low Power and Lossy
   Networks (RPL)" as indicated in Table 26:

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

             Table 26: RPL Control Message Options

11.7. SubRegistry for the Projected DAO Request Flags

   IANA is requested to create a registry for the 8-bit "Projected DAO
   Request (PDR)" field under the heading "Routing Protocol for Low
   Power and Lossy Networks (RPL)". The bits are indexed from 0
   (leftmost) to 7. Each bit is tracked with the following qualities:

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

   * Capability description

   * Reference

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

```
+===========+=======================================+===========+
| Bit number | Capability description               | Reference |
+===========+=======================================+===========+
|     0     | PDR-ACK request (K)                  | THIS RFC  |
+-----------+---------------------------------------+-----------+
|     1     | Requested path should be redundant (R) | THIS RFC  |
+-----------+---------------------------------------+-----------+
|   2..255  | Unassigned                           |           |
+-----------+---------------------------------------+-----------+
```

                   Table 27: Initial PDR Flags

11.8. SubRegistry for the PDR-ACK Flags

   IANA is requested to create a registry for the 8-bit "PDR-ACK Flags"
   field under the heading "Routing Protocol for Low Power and Lossy
   Networks (RPL)". The bits are indexed from 0 (leftmost) to 7. Each
   bit is tracked with the following qualities:

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

   * Capability description

   * Reference

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

11.9. Registry for the PDR-ACK Acceptance Status Values

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

   * Value

   * Meaning

   * Reference

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

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

```
+-------+----------------------+----------+
| Value | Meaning              | Reference |
+-------+----------------------+----------+
| 0     | Unqualified Acceptance | THIS RFC |
+-------+----------------------+----------+
| 1..63 | Unassigned           |          |
+-------+----------------------+----------+
```

          Table 28: Acceptance values of the PDR-ACK
                           Status

11.10. Registry for the PDR-ACK Rejection Status Values

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

   * Value

   * Meaning

   * Reference

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

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

```
+-------+----------------------+----------+
| Value | Meaning              | Reference |
+-------+----------------------+----------+
| 0     | Unqualified Rejection | THIS RFC |
+-------+----------------------+----------+
| 1     | Transient Failure    | THIS RFC |
+-------+----------------------+----------+
| 2..63 | Unassigned           |          |
+-------+----------------------+----------+
```

          Table 29: Rejection values of the PDR-ACK
                           Status

11.11. SubRegistry for the Via Information Options Flags

   IANA is requested to create a registry for the 8-bit "Via Information
   Options (VIO) Flags" field under the heading "Routing Protocol for
   Low Power and Lossy Networks (RPL)". The bits are indexed from 0
   (leftmost) to 7. Each bit is tracked with the following qualities:

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

* Capability description

* Reference

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

11.12. SubRegistry for the Sibling Information Option Flags

IANA is requested to create a registry for the 5-bit "Sibling Information Option (SIO) Flags" field under the heading "Routing Protocol for Low Power and Lossy Networks (RPL)". The bits are indexed from 0 (leftmost) to 4. Each bit is tracked with the following qualities:

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

* Capability description

* Reference

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

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

Table 30: Initial SIO Flags

11.13. Destination Advertisement Object Flag

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

```
+--------------+------------------------+-----------+
| Bit Number   | Capability Description  | Reference |
+--------------+------------------------+-----------+
| 2 (Suggested)| Projected DAO (P)       | THIS RFC  |
+--------------+------------------------+-----------+
```

Table 31: New Destination Advertisement Object
(DAO) Flag

11.14. Destination Advertisement Object Acknowledgment Flag

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

```
+--------------+------------------------+-----------+
| 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

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

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

Table 33: Rejection values of the RPL Status

12. Acknowledgments

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

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>.

   [RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-
              Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)",
              RFC 9030, DOI 10.17487/RFC9030, May 2021,
              <https://www.rfc-editor.org/info/rfc9030>.

   [RAW-ARCHI]
              Thubert, P., "Reliable and Available Wireless
              Architecture", Work in Progress, Internet-Draft, draft-
              ietf-raw-architecture-10, 14 November 2022,
              <https://www.ietf.org/archive/id/draft-ietf-raw-
              architecture-10.txt>.

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>.

   [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>.

   [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-08, 22 October 2022,
              <https://www.ietf.org/archive/id/draft-ietf-raw-use-cases-
              08.txt>.

   [I-D.kuehlewind-update-tag]
              Kühlewind, M. and S. Krishnan, "Definition of new tags for
              relations between RFCs", Work in Progress, Internet-Draft,
              draft-kuehlewind-update-tag-04, 12 July 2021,
              <https://www.ietf.org/archive/id/draft-kuehlewind-update-
              tag-04.txt>.

   [I-D.irtf-panrg-path-properties]
              Enghardt, R. and C. Krähenbühl, "A Vocabulary of Path
              Properties", Work in Progress, Internet-Draft, draft-irtf-
              panrg-path-properties-06, 22 September 2022,
              <https://www.ietf.org/archive/id/draft-irtf-panrg-path-
              properties-06.txt>.

   [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/

        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

Copyright Notice

Table of Contents

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 RPI 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) formating 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.

```
+-------------+
|             |
| Upper Layers|
|             |
+-------------+
|             |
|    RPL      |
|             |
+-------------+
|             |
|   ICMPv6    |
|             |
+-------------+
|             |
|    IPv6     |
|             |
+-------------+
|             |
|   6LoWPAN   |
|             |
+-------------+
|             |
|   PHY-MAC   |
|             |
+-------------+
```
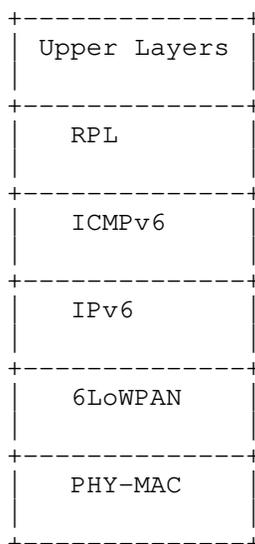
                       Figure 1: RPL Stack.

   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  |    Binary Value   |  Description  | Reference |
+ Value +-------------------+               +           +
|       | 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   |   Binary Value    | Description | Reference |
+ Value +-------------------+            +           +
|       | 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        |11110CPP| UDP | UDP
|Page 1  |Type1 S=0| 6LoRH  |6LoRH   |LOWPAN_IPHC| UDP    | hdr |Payld
+-+ ... -+-+ ... +-+- ... -+-+-.+-+-+-+ ... +-+-+ ... -+ ... +-...
         <-4bytes->                        <-      RFC 6282     ->
                                              No RPL artifact
```
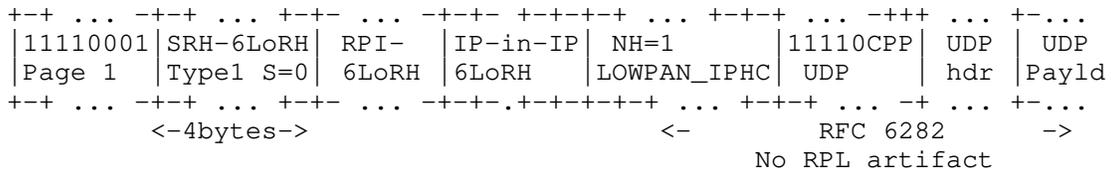
          Figure 5: RPI Inserted by the Root in Storing Mode

   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.

```
                  +-----------+
                  | INTERNET  ----------+
                  |           |         |
                  +-----------+         |
                                        |
                                        |
                                     A  |
                                +-------+
                                | 6LBR  |
                   +------------| (root)|-------+
                   |            +-------+       |
                   |                            |
                   |                            |
                   |                            |
                B  |                         C  |
             +--- ---+                   +--- ---+
             |  6LR  |                   |  6LR  |
         +---------- ---+  --+       +---     ---+  |
         |            +-------+       |    +-------+ |
         |                            |              |
         |                            |              |
         |                            |              |
      D  |                         E  |              |
     +- -----+                   +--- ---+           |
     |  6LR  |                   |  6LR  |           |
     |       |          +------- |       |           |
     +--- ---+          |        +--- ---+           |
         |              |            |               |
         |              |          +--+              |
         |              |          |                 |
         |              |          |                 |
      F  |           G  |       H  |      I  |    J  |
   +----+-+       +- -----+   +--- --+   +--- ---+  +--- ---+
   | RAL  |       | RUL   |   | RAL  |   | RAL   |  | RUL   |
   | 6LN  |       | 6LN   |   | 6LN  |   | 6LN   |  | 6LN   |
   +------+       +-------+   +------+   +-------+  +-------+
```
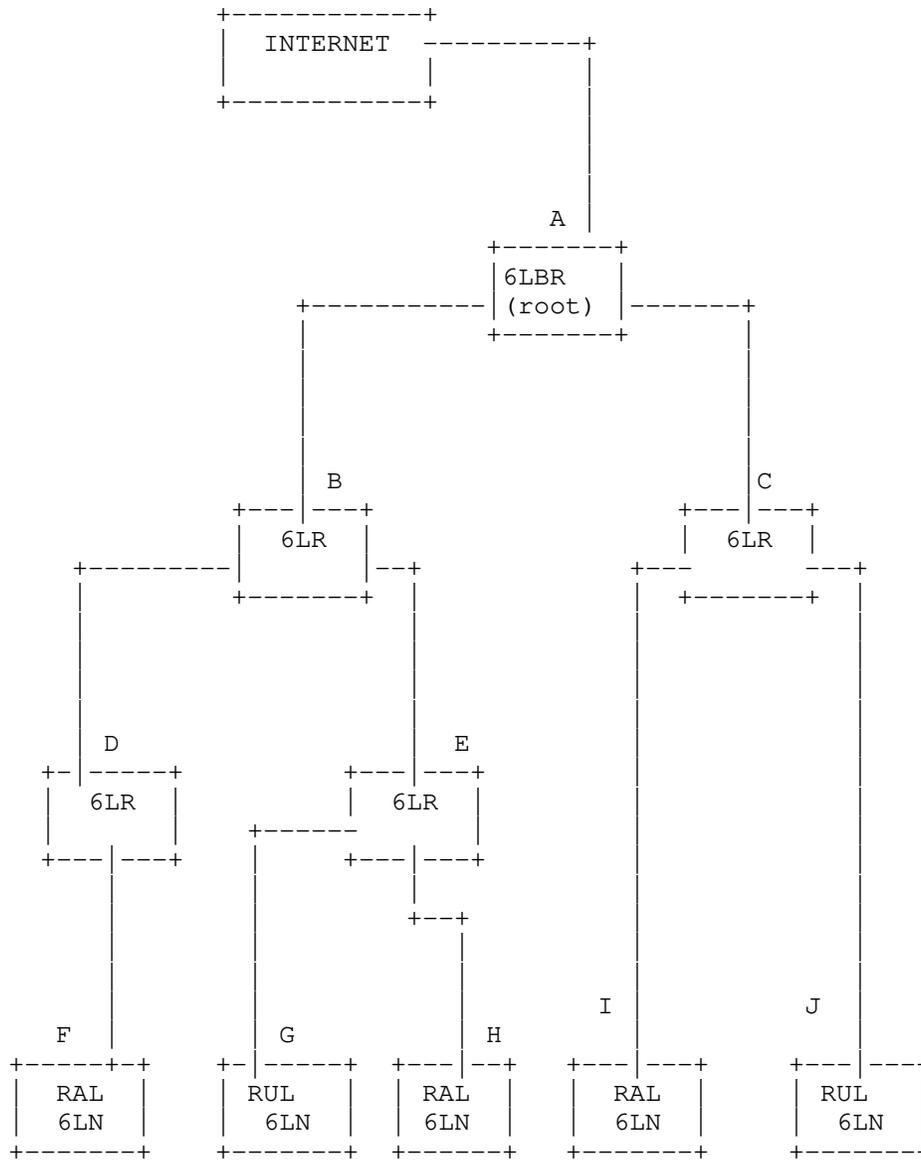
                    Figure 6: A reference RPL Topology.

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
	RAL to root	No	N/A
Leaf - Root	root to RAL	No	N/A
	root to RUL	must	6LR
	RUL to root	must	root
	RAL to Int	may	root
Leaf - Internet	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 | 6LR_i | RAL |
   | | src | | dst |
   +----------+------+-------+-----+
   | Added | RPI | -- | -- |
   | headers | | | |
   +----------+------+-------+-----+
   | Modified | -- | RPI | -- |
   | headers | | | |
   +----------+------+-------+-----+
   | Removed | -- | -- | RPI |
   | headers | | | |
   +----------+------+-------+-----+
   | 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 <= i <= 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=(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) --> 6LR_1 --> 6LR_i --> root (6LBR)

   For example, a communication flow could be: Node G (RUL) --> Node E
   (6LR_1)--> Node B (6LR_i)--> Node A root(6LBR)

   6LR_i represents the intermediate routers from the source (RUL) to
   the destination (6LBR), 1 <= i <= n, where n is the total number of
   routers (6LR) that the packet goes through from the RUL to the 6LBR.

   When the packet arrives from the RUL (Node G) to 6LR_1 (Node E), the
   6LR_1 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 <= i <= 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- |   6LBR    |    6LR_i    |    6LR_n  |  RUL  |
|         |  net  |           | [i=1,..,n-1]|           |  dst  |
|         |  src  |           |             |           |       |
|         |       |           |             |           |       |
+---------+-------+-----------+-------------+-----------+-------+
| Inserted|  --   |IP6-IP6(RPI)|     --     |     --    |  --   |
| headers |       |           |             |           |       |
+---------+-------+-----------+-------------+-----------+-------+
| Modified|  --   |    --     |     RPI     |     --    |  --   |
| headers |       |           |             |           |       |
+---------+-------+-----------+-------------+-----------+-------+
| Removed |  --   |    --     |     --      |IP6-IP6(RPI)|  --  |
| headers |       |           |             |           |       |
+---------+-------+-----------+-------------+-----------+-------+
|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 <= ia <= 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 <= id <= 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 <= ia <= 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 <= id <= 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 <= ia <= 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 <= id <= 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 <= ia <= 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
   <= id <= 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 |    6LR_1   | 6LR_ia |   6LBR  | 6LR_id |6LR_n  |RUL|
|         |src |            |        |         |        |       |dst|
|         |    |            |        |         |        |       |   |
+---------+----+------------+--------+---------+--------+-------+---+
|  Added  | -- |IP6-IP6(RPI1)|   --  | IP6-IP6 |   --   |  --   | --|
| Headers |    |            |        |  (RPI2) |        |       |   |
+---------+----+------------+--------+---------+--------+-------+---+
|Modified | -- |     --     |        |    --   |        |  --   | --|
|headers  |    |            |  RPI1  |         |  RPI2  |       |   |
+---------+----+------------+--------+---------+--------+-------+---+
| Removed | -- |     --     |   --   | IP6-IP6 |   --   |IP6-IP6| --|
| headers |    |            |        |  (RPI1) |        | (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 <= i <= 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 <= i <= 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   |  6LR_i  |   RAL   |
|          |   src   |         |   dst   |
+----------+---------+---------+---------+
|  Added   | RPI, RH3|   --    |   --    |
| headers  |         |         |         |
+----------+---------+---------+---------+
| Modified |   --    | RPI, RH3|   --    |
| headers  |         |         |         |
+----------+---------+---------+---------+
| Removed  |   --    |   --    | RPI, RH3|
| headers  |         |         |         |
+----------+---------+---------+---------+
|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 <= i <= 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=(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) --> 6LR_1 --> 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 <= i <= n, where n is the total number of routers (6LR) that the packet goes through from source (RUL) to destination (6LBR). For example, 6LR_1 (i=1) is the router that receives the packets from the RUL.

In this case, the RPI is added by the first 6LR (6LR_1) (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.

```
+--------+----+----------------+----------------+-----------------+
|        |RUL |                |                |                 |
| Header |src |     6LR_1      |     6LR_i      |    6LBR dst     |
|        |node|                |                |                 |
+--------+----+----------------+----------------+-----------------+
| Added  | -- |IPv6-in-IPv6(RPI)|      --       |       --        |
| headers|    |                |                |                 |
+--------+----+----------------+----------------+-----------------+
|Modified| -- |      --        |     RPI        |       --        |
| headers|    |                |                |                 |
+--------+----+----------------+----------------+-----------------+
|Removed | -- |      --        |      --        |IPv6-in-IPv6(RPI)|
| headers|    |                |                |                 |
+--------+----+----------------+----------------+-----------------+
|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 <= i <= 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 | 6LR_i | 6LBR |  Internet |
|           | src |       |      |    dst    |
+-----------+-----+-------+------+-----------+
|   Added   | RPI |  --   |  --  |    --     |
|  headers  |     |       |      |           |
+-----------+-----+-------+------+-----------+
| Modified  | --  |  RPI  | RPI  |    --     |
|  headers  |     |       |      |           |
+-----------+-----+-------+------+-----------+
|  Removed  | --  |  --   |  --  |    --     |
|  headers  |     |       |      |           |
+-----------+-----+-------+------+-----------+
| Untouched | --  |  --   |  --  |    RPI    |
|  headers  |     |       |      | (Ignored) |
+-----------+-----+-------+------+-----------+
```

Figure 27: Non-SM: Summary of the use of headers from RAL to Internet
                        with no encapsulation

```
+----------+-------------+-------------+-------------+----------+
|  Header  |    RAL      |    6LR_i    |    6LBR     | Internet |
|          |    src      |             |             |   dst    |
+----------+-------------+-------------+-------------+----------+
|  Added   | IPv6-in-IPv6|     --      |     --      |    --    |
| headers  |   (RPI)     |             |             |          |
+----------+-------------+-------------+-------------+----------+
| Modified |     --      |             |     --      |    --    |
| headers  |             |     RPI     |             |          |
+----------+-------------+-------------+-------------+----------+
| Removed  |     --      |     --      | IPv6-in-IPv6|    --    |
| headers  |             |             |   (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 <= i <= n, where n is the total number of routers (6LR) that the
   packet goes through from 6LBR to destination (RAL).

   The 6LBR must add an RH3 header. As the 6LBR will know the path and
   address of the target node, it can address the IPv6-in-IPv6 header to
   that node. The 6LBR will zero the flow label upon entry in order to
   aid compression [RFC8138].

   The Figure 29 summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	RAL dst
Added headers	--	IPv6-in-IPv6 (RH3, RPI)	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3, RPI)	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3, RPI)
Untouched headers	--	--	--	--

Figure 29: Non-SM: Summary of the use of headers from Internet to RAL

8.2.3. Non-SM: Example of Flow from RUL to Internet

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_i -->root (6LBR) --> Internet
dst

For example, a communication flow could be: Node G --> Node E -->
Node B --> Node A --> Internet

6LR_i represents the intermediate routers from source to destination,
1 <= 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., 6LR_1
(i=1).

In this case the flow label is recommended to be zero in the RUL. As
the RUL parent adds RPL headers in the RUL packet, the first 6LR
(6LR_1) will add an RPI inside a new IPv6-in-IPv6 header. The IPv6-
in-IPv6 header will be addressed to the root. This case is identical
to the storing-mode case (see Section 7.2.3).

The Figure 30 shows the table that summarizes what headers are needed
for this use case.

Header	RUL src node	6LR_1	6LR_i [i=2,..,n]	6LBR	Internet dst
Added headers	--	IP6-IP6(RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6(RPI)	--
Untouched headers	--	--	--	--	--

Figure 30: Non-SM: Summary of the use of headers from RUL to Internet

8.2.4. Non-SM: Example of Flow from Internet to RUL

In this case the flow comprises:

Internet src --> root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

For example, a communication flow could be: Internet --> Node A (root) --> Node B --> Node E --> Node G

6LR_i represents the intermediate routers from source to destination, 1 <= i <= 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
   <= ia <= 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 <= id <= 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_ia 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
   <= ia <= n, where n is the total number of intermediate routers (6LR)

   6LR_id represents the intermediate routers from the root to the
   destination, 1 <= id <= 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
   <= ia <= n, where n is the total number of intermediate routers (6LR)

   6LR_id represents the intermediate routers from the root to the
   destination, 1 <= id <= 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	6LR_1	6LR_ia	6LBR	6LR_id	RAL
	src					dst
	node					node
   +----------+------+--------+--------+--------+--------+--------+
Added	--	IP6-IP6	--	IP6-IP6	--	--
headers		(RPI1)		(RH3,		
				RPI2)		
   +----------+------+--------+--------+--------+--------+--------+
Modified	--			--	IP6-IP6	--
headers		--	RPI1		(RH3,	
					RPI2)	
   +----------+------+--------+--------+--------+--------+--------+
Removed	--		--	IP6-IP6	--	IP6-IP6
headers		--		(RPI1)		(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
   <= ia <= n, where n is the total number of intermediate routers (6LR)

   6LR_id represents the intermediate routers from the root to the
   destination, 1 <= id <= 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

                         No-Path DAO modifications
                     draft-jadhav-roll-efficient-npdao-00

Abstract

   This document describes the problems associated with the use of No-
   Path DAO messaging in RPL and a signaling improvement to improve
   route invalidation efficiency.

Status of This Memo

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

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

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

   This Internet-Draft will expire on August 18, 2017.

Copyright Notice

Table of Contents

1. Introduction

   RPL [RFC6550] specifies a proactive distance-vector based routing
   scheme. The specification has an optional messaging in the form of
   DAO messages using which the 6LBR can learn route towards any of the
   nodes. In storing mode, DAO messages would result in routing entries

been created on all intermediate hops from the node's parent all the way towards the 6LBR.

RPL also allows use of No-Path DAO (NPDAO) messaging to invalidate a routing path and thus releasing of any resources utilized on that path. A No-Path DAO is a DAO message with route lifetime of zero, signaling route invalidation for the given target. This document studies the problems associated with the current use of No-Path DAO messaging, which creates route inefficiency and inconsistence. This document also discusses the requirements for an optimized No-Path DAO messaging scheme.

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].

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

Common Ancestor node: 6LR node which is the first common node on the old and new path for the child node.

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

New parent: Parent 6LR node after switching to the new path.

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

Reverse NPDAO: A No-Path DAO message which traverses downstream in the network.

Regular DAO: A DAO message with non-zero lifetime.

This document also uses terminology described in [RFC6550] and [RFC6775].

1.2. Current No-Path DAO messaging

RPL introduced No-Path DAO messaging in the storing mode so that the node switching its current parent can inform its parents and ancestors to invalidate the existing route. Subsequently parents or ancestors would release any resources (such as the routing entry) it maintains on behalf of that child node. The No-Path DAO message

always traverses the RPL tree in upward direction, originating at the
target node itself.

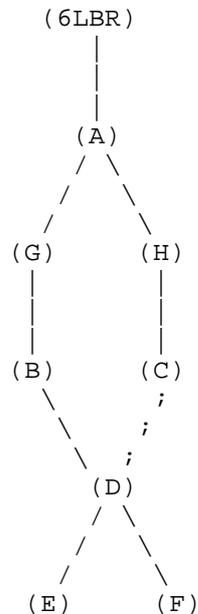For the rest of this document consider the following topology:

```
                      (6LBR)
                        |
                        |
                        |
                      (A)
                      / \
                     /   \
                    /     \
                 (G)       (H)
                  |         |
                  |         |
                  |         |
                 (B)       (C)
                   \        ;
                    \       ;
                     \    ;
                      (D)
                      / \
                     /   \
                    /     \
                 (E)       (F)
```

Figure 1: Sample topology

Node (D) is connected via preferred parent (B). (D) has an alternate
path via (C) towards the BR. Node (A) is the common ancestor for (D)
for paths through (B)-(G) and (C)-(H). When (D) switches from (B) to
(C), [RFC6550] suggests sending No-Path DAO to (B) and regular DAO to
(C).

1.3. Cases when No-Path DAO may be used

There are following cases in which a node switches its parent and may
employ No-Path DAO messaging:

Case I: Current parent becomes unavailable because of transient or
permanent link or parent node failure.

Case II: The node finds a better parent node i.e. the metrics of
another parent is better than its current parent.

Case III: The node switches to a new parent whom it "thinks" has a
better metric but does not in reality.

The usual steps of operation when the node switches the parent is
that the node sends a No-Path DAO message via its current parent to
invalidate its current route and subsequently it tries to establish a
new routing path by sending a new DAO via its new parent.

1.4. Why No-Path DAO is important?

Nodes in LLNs may be resource constrained. There is limited memory
available and routing entry records are the 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
achieve resource utilization in case of contention. Thus it becomes
necessary to have 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 invalidation
needs to be done for (D), (E) and (F). Thus without efficient route
invalidation, a 6LR may have to hold a lot of unwanted route entries.

2. Problems with current No-Path DAO messaging

There are following problems with the usage of current NP-DAO
messaging

2.1. Lost NP-DAO due to link break to the previous parent

When the node switches its parent, the NPDAO is to be sent via its
previous parent and a regular DAO via 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. RPL
assumes communication link with the previous parent for No-Path DAO
messaging.

RPL mentions use of route lifetime to remove unwanted routes in case
the routes could not be refreshed. But route lifetimes in case of
LLNs could be substantially high and thus the route entries would be
stuck for long.

2.2. Invalidate routes to dependent nodes of the switching node

No-path DAO is sent by the node who has switched the parent but it
does not work for the dependent child nodes below it. The
specification does not specify how route invalidation will work for
sub-childs, resulting in stale routing entries on behalf of the sub-
childs on the previous route. 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. Post switching, Node (D) transmits a DIO with
incremented DTSN so that child nodes, node (E) and (F), generate DAOs
to trigger route update on the new path for themselves. There is no
NPDAO generated by these child nodes through the previous path
resulting in stale entries on nodes (B) and (G) for nodes (E) and
(F).

2.3. 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 thus impacting downward traffic for the switching node. In
   the example topology, consider Node (D) switches from parent (B) to
   (C) because the metrics of the path via (C) are better. Note that
   the previous path via (B) may still be available (albeit at
   relatively bad metrics). An NPDAO sent from previous route may
   invalidate the existing route whereas there is no way to determine
   whether the new DAO has successfully updated the route entries on the
   new path.

   An implementation technique to avoid this problem is to further delay
   the route invalidation by a fixed time interval after receiving an
   NPDAO, considering the time taken for the new path to be established.
   Coming up with such a time interval is tricky since the new route may
   also not be available and it may subsequently require more parent
   switches to establish a new path.

3. Requirements for the No-Path DAO Optimization

   We identify the following requirements for the NP-DAO optimization.

3.1. Req#1: Tolerant to the link failures to the previous parents

   When the switching node send the NP-DAO message to the previous
   parent, it is normal that the link to the previous parent is prone to
   failure. Therefore, it is required that the NP-DAO message MUST be
   tolerant to the link failure during the switching.

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

   While switching the parent node and sending NP-DAO message, it is
   required that the routing entries to the dependent nodes of the
   switching node will be updated accordingly on the previous parents
   and other relevant upstream nodes.

3.3. Req#3: No impact on traffic while NP-DAO operation in progress

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

4. Existing Solution

4.1. NP-DAO can be generated by the parent node who detects link
      failure to the child

   RPL states mechanisms which could be utilized to clear DAO states in
   a sub-DODAG. [RFC6550] Section 11.2.2.3 states "With DAO
   inconsistency loop recovery, a packet can be used to recursively
   explore and clean up the obsolete DAO states along a sub-DODAG".

   Thus in the sample topology in Figure 1, when Node (B) detects link
   failure to (D), (B) has an option of generating an NP-DAO on behalf
   of Node (D) and its sub-childs, (E) and (F).

   This section explains why generation of an NP-DAO in such cases may
   not function as desired. Primarily the DAO state information in the
   form of Path Sequence plays a major role here. Every target is
   associated with a Path Sequence number which relates to the latest
   state of the target. [RFC6550] Section 7.1 explains the semantics of
   Path Sequence number. The target node increments the Path Sequence
   number every time it generates a new DAO. The router nodes en-route
   utilize this Path Sequence number to decide the freshness of target
   information. If a non-target node has to generate an NP-DAO then it
   could use following two possibilities with Path Sequence number:

   Let the Path Sequence number of old regular DAO that flowed through
   (B) be x. The subsequent regular DAO generated by Node (D) will have
   sequence number x+1.

   i. Node (B) uses the previous Path Sequence number from the regular
   DAO i.e. NP-DAO(pathseq=x)

   ii. Node (B) increments the Path Sequence number i.e. NP-
   DAO(pathseq=x+1)

   In case i, the NP-DAO(pathseq=x) will be dropped by all the
   intermediate nodes since the semantics of Path Sequence number
   dictates that any DAO with an older Path Sequence number be dropped.

In case ii, there is a risk that the NP-DAO(pathseq=x+1) traverses up the DODAG and invalidates all the routes till the root and then the regular DAO(pathseq=x+1) from the target traverses upwards. In this case the regular DAO(pathseq=x+1) will be dropped from common ancestor node to the root. This will result in route downtime.

Another problem with this scheme is its dependence on the upstream neighbor to detect that the downstream neighbor is unavailable. There are two possibilities by which such a detection might be put to work:

i. There is P2P traffic from the previous sub-DODAG to any of nodes in the sub-tree which has switched the path. In the above example, lets consider that Node (G) has P2P traffic for either of nodes (D), (E), or (F). In this case, Node (B) will detect forwarding error while forwarding the packets from Node (B) to (D). But dependence on P2P traffic may not be an optimal way to solve this problem considering the reactive approach of the scheme. The P2P traffic pattern might be sparse and thus such a detection might kick-in too late.

ii. The other case is where Node (B) explicitly employs some mechanism to probe directly attached downstream child nodes. Such kind of schemes are seldom used.

4.2. NP-DAO can be generated once the link is restored to the previous parent

This scheme solves a specific scenario of transient links. The child node can detect that the connection to previous parent is restored and then transmit an NP-DAO to the previous parent to invalidate the route. This scheme is stateful, thus requires more memory and solves a specific scenario.

5. Proposed changes to NPDAO signaling

5.1. Change in NPDAO semantics

As described in Section 1.2, currently the NPDAO originates at the node switching the parent and traverses upstream towards the root. In order to solve the problems as mentioned in Section 2, the draft proposes to change the way NPDAO originates and traverses the network. The new NPDAO proposed does not originate at the node but instead originates at a common ancestor node between the new and old path. The trigger for the common ancestor node to generate this NPDAO is the change in the next hop for the node on reception of an update message in the form of regular DAO for the target.

In the 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 target as address of D and a incremented path sequence
number. Node C will update the routing table based on the
reachability information in 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. the 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), may generate an NPDAO which traverses
downwards in the network. The document in the subsequent section
will explain the message format changes to handle this downward flow
of NPDAO.

5.2. DAO message format changes

   Every RPL message is divided into base message fields and additional
   Options. 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 specifies the reachability information for the given
   targets. Similarly, a Transit Information option may be associated
   with a set of RPL Target options.

   The draft proposes a change in DAO message to contain "Invalidate
   previous route" (I) bit. This I-bit which is carried in regular DAO
   message, signals the common ancestor node to generate a downstream
   NPDAO on behalf of the target node. The I-bit is carried in the
   transit container option which augments the reachability information
   for a given set of RPL Target(s).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 0x06 |  Option Length |E|I|  Flags   | Path Control |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path Sequence | Path Lifetime  |                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                              +
|                                                               |
+                                                               +
|                                                               |
+                         Parent Address*                       +
|                                                               |
+                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

I (Invalidate previous route) bit: 1 bit flag. The 'I' flag is set
by the target node to indicate that it wishes to invalidate the
previous route by a common ancestor node between the two paths.

The NPDAO thus generated by the common ancestor node needs to
traverse downstream. An additional flag called as "Reverse NPDAO"
(R) is added in the base DAO object to signal this change.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RPLInstanceID |K|D|R| Flags    |   Reserved    | DAOSequence   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                          DODAGID*                             +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option(s)...
+-+-+-+-+-+-+-+-+
```

        Figure 3: Updated DAO base object (New R flag added)

 R (Reverse DAO) bit: 1 bit flag. The 'R' flag is used to signal that
 the DAO traverses downwards.

5.2.1. Path Sequence number in the reverse NPDAO

   Every DAO message may contain a Path Sequence in the transit
   information option to identify the freshness of the DAO message. The
   Path Sequence in the downward NPDAO generated by common ancestor
   should use the same Path Sequence number present in the regular DAO
   message.

5.3. Example messaging

   In Figure 1, node (D) switches its parent from (B) to (C). The
   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 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.

   4. Similar to C, node H checks for 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 H in a
       DAO.

   5. A receives the DAO, and checks for routing entry on behalf of D.
       It finds a routing entry but checks that the next hop for target
       D is now changed. Node A checks the I_flag and generates
       downstream NPDAO(tgt=D,pathseq=x+1,R_flag=1) to previous next hop
       for target D which is G. Subsequently, A updates the routing
       entry and forwards the reachability information of target D
       upstream DAO(tgt=D,pathseq=x+1,I_flag=x) (the I_flag carries no
       significance henceforth).

   6. Node G receives the downstream NPDAO and invalidates routing
       entry of target D and then checks the reverse (R) flag and
       forwards the (un)reachability information downstream to B.

   7. Similarly, B processes the downstream NPDAO by invalidating the
       routing entry of target D and then checks the reverse (R) flag
       and forwards the (un)reachability information downstream to D.

   8. D ignores the downstream NPDAO since the target is itself.

5.4. Other considerations

5.4.1. Dependent Nodes invalidation

   Current RPL [RFC6550] does not provide a mechanism for route
   invalidation for dependent nodes.

   This section describes approaches for invalidating routes of
   dependent nodes if the implementation chooses to solve this problem.
   The common ancestor node realizes that the paths for dependent nodes
   have changed (based on next hop change) when it receives a regular
   DAO on behalf of the dependent nodes. Thus dependent nodes route
   invalidation can be handled in the same way as the switching node.
   Note that there is no way that dependent nodes can set the I_flag in
   the DAO message selectively since they are unaware that their parent/
   grand parent node is switching paths. There are two ways to handle
   dependent node route invalidation:

   1. One way to resolve is that the common ancestor does not depend
       upon the I_flag to generate the reverse NPDAO. The only factor
       it makes the decision will be based on next_hop change for an
       existing target to generate the NPDAO. Thus when the switching
       nodes and all the below dependent nodes advertise a regular DAO,
       the common ancestor node will detect a change in next hop and
       generate NPDAO for the same target as in the regular DAO.

   2. Another way is that the nodes always set the I_flag whenever they
       send regular DAO. Thus common ancestor will first check whether
       I_flag is set and then check whether the next_hop has changed and
       subsequently trigger NPDAO if required.

   This document recommends the approach in point 2. The advantage with
   I_flag is that the generation of downstream NPDAO is still controlled
   by the target node and thus is still in control of its own routing
   state.

6. Acknowledgements

   We would like to thank Cenk Gundogan, Simon Duquennoy and Pascal
   Thubert for their review and insightful comments.

7. IANA Considerations

   IANA is requested to allocate bit 11 in the DAO base object defined
   in RPL [RFC6550] section 6.4 for reverse 'R' NPDAO flag.

IANA is requested to allocate bit 18 in the Transit Information
Option defined in RPL [RFC6550] section 6.7.8 for Invalidate route
'I' flag.

8. Security Considerations

   TBA

9. References

9.1. Normative References

   [CONTIKI] Thingsquare, "Contiki: The Open Source OS for IoT", 2012,
              <http://www.contiki-os.org>.

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

9.2. Informative References

   [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
              Text on Security Considerations", BCP 72, RFC 3552,
              DOI 10.17487/RFC3552, July 2003,
              <http://www.rfc-editor.org/info/rfc3552>.

   [RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H.,
              and A. Yegin, "Protocol for Carrying Authentication for
              Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191,
              May 2008, <http://www.rfc-editor.org/info/rfc5191>.

   [RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Ed., and
              A. Yegin, "Protocol for Carrying Authentication for
              Network Access (PANA) Relay Element", RFC 6345,
              DOI 10.17487/RFC6345, August 2011,
              <http://www.rfc-editor.org/info/rfc6345>.

   [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J.,
              Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur,
              JP., and R. Alexander, "RPL: IPv6 Routing Protocol for
              Low-Power and Lossy Networks", RFC 6550,
              DOI 10.17487/RFC6550, March 2012,
              <http://www.rfc-editor.org/info/rfc6550>.

   [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,
              <http://www.rfc-editor.org/info/rfc6775>.

Appendix A. Additional Stuff

   This becomes an Appendix.

Authors' Addresses

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

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

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

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

   Zhen Cao
   Huawei Tech
   W Chang'an Ave
   Beijing 560037
   China

   Email: zhencao.ietf@gmail.com

                  Load Balancing Objective Function in RPL
                    draft-qasem-roll-rpl-load-balancing-02

Abstract

   This document proposes an extended Objective Function(OF)that
   balances the number of child nodes of the parent nodes to avoid the
   overloading problem and ensure node lifetime maximization in the IPv6
   Routing Protocol for Low-Power and Lossy Networks (RPL). The standard
   OFs are used to build a Destination Oriented Directed Acyclic Graph
   (DODAG) where the bottleneck nodes may suffer from unbalanced traffic
   load. As a result, a part of the network may be disconnected as the
   energy of the overloaded preferred parent node will drain much faster
   than other candidate parents. Thus, a new RPL metric has been
   introduced to balance the traffic load over the network. Finally, the
   potential extra overhead has been mitigated using a new utilization
   technique.

   http://www.ietf.org/1id-abstracts.html

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html

Table of Contents

1 Introduction

IPv6 Routing Protocol for LLNs (RPL) [RFC6550] defined two OFs to optimize the path selection towards the root node, namely, the OF zero (OF0) [RFC6552], and the Minimum Rank with Hysteresis OF (MRHOF)[RFC6719]. The Destination Oriented Directed Acyclic Graph (DODAG) construction is built by the RPL OF, that specify how nodes select the preferred parent node by translating one or more metrics into the rank value.

The used OF calculates the rank based on some routing metrics [RFC 6551] such as hop-count, delay, energy, and so forth. The parent node in RPL can serve more than one child if it is chosen by them as preferred parent. Consequently, the overloaded preferred parents will become fragile nodes as their energy risks to drain much quicker than other nodes.

Having conducted simulation experiments and rigours analysis, it is concluded that the current OFs lead to build a topology that suffers from an unbalanced load traffic in bottleneck nodes especially for the first hop nodes (i.e., from the root). Consequently, this problem has a crucial impact on the lifetime of these types of nodes. The battery depletion of that overloaded parent node may affect the network reliability negatively.

This challenging problem is still an open issue. In an attempt to overcome this problem, this draft proposes a new OF to mitigate the overusing of the bottleneck node to prolong its battery lifetime.

This draft proposes an extended Objective Function(OF) that balances the number of children nodes for the overloaded nodes to ensure node lifetime maximization in RPL and can be summarized as follows. First, a new RPL metric has been used to balance the load traffic among the bottleneck nodes. Second, the DODAG Information Object (DIO) message has been amended by injecting the IP address of the chosen parent before broadcasting it. Third, a new utilization technique has been proposed for the amended DIO message to avoid increasing the overhead of the handshaking and acknowledgment processes. Simulation experiments have been conducted to validate the extended OF performance as detailed in Appendix A.

1.1 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].

2 DODAG construction in a nutshell

RPL is a proactive distance vector routing protocol designed for LLNs [RFC6550], it constructs a DODAG using a certain OF that suits the application requirements. Essentially, RPL relies on a DODAG Information Object (DIO) control message to build the DODAG.

Thus, the starting point begins when the root node broadcasts the DIO message to the downstream neighbor nodes. As soon as the closest node receives the message, it can decide whether to join this DODAG or not based on the calculated rank according to the equations (1) and (2) [RFC6719].

Rank(N) = Rank(PN) + RankIncrease   (1)

RankIncrease = Step * MinHopRankIncrease   (2)

Where Step represents a scalar value and MinHopRankIncrease represents the minimum RPL parameter. If the node decides to join, then it adds the DIO sender to the candidate parent list. Next, the preferred parent, i.e. the next hop to the root, will be chosen based on the rank from this list to receive all traffic from the child node. Then, it computes its own rank with a monotonical increase according to the selected OF.

After that, the node propagates its own DIO with all updated information to all its neighbors including the preferred parent. [RFC 6551] defined the number of node metrics/constraints (e.g. hop count and energy) and the link metrics/constraints (e.g. ETX and throughput) that might be used in the OFs [RFC6552][RFC6719].

3 Load balancing in RPL

RPL is designed with several robust features such as exiguous delay, quick configuration, loop-free topology, and self-healing. However, the load imbalance is considered as a significant weakness in this protocol. More specifically, RPL is dealing with non-uniform distribution in large-scale LLNs, which may lead to unequal data traffic distribution. Consequently, the energy of the overloaded nodes will be drained much faster than other nodes. Furthermore, this problem has more harmful impacts if the overloaded node is a bottleneck node (i.e. with the first hop to the root) as shown in Figure 1 for node A and B.

```
         ------+---------
              |             Internet
              |
         +-----+
         |     |  Border Router
         |     |   (RPL Root)
         +-----+
              |
       o    N  / \
      o o M F A   B
        o o G E C D H
       o  o   P  R J K o o o
   o   o  o   o   o   o
     o   o    o       o
   o   o     o
     o   o o
```

            Figure 1: Bottleneck nodes in RPL

Figure 2 depicts the selection of the preferred parent for those nodes
are within the first hop from nodes A or B. Clearly, node A has more
children as it is surrounded by the nodes (N,M,F,G,E,P). Despite the
fact that A has more children, it dominates the shred nodes (C,D,R,J)
that are also located within the shared area of node B (i.e., within the
transmission range of A and B). That unbalanced parent selection
approach in RPL left node B only with two children, while node A has ten
children.

```
+---------------------------------------------------------+
|  Parent    |     Child nodes      |  Shared nodes   |
|  nodes     |                      | between A and B |
|------------------------------------------------------|
|    A       | N,M,F,G,E,P,C,D.R,J  | C,D,R,J         |
|------------------------------------------------------|
|    B       | H,K                  | C,D,R,J         |
+---------------------------------------------------------+
```

            Figure 2: The selection of the preferred parents

It is notable that the connection of all nodes through A is fragile as
it is the only link to the root with an overloaded bottleneck node,
thus, disconnecting part of the network if node A dies. In particular,
this serious problem occurs in RPL due to omitting the number of
children in existing parent selection technique.

To this end, the node sticks with the current preferred parent and influences its rank, even if this parent deteriorates with more load (i.e. being a parent for more children). The only conceivable scenarios to change the current parent to another candidate parent are as follow: first, if the current parent dies due to battery depletion. The second possibility, when the lossy percentage becomes higher than before, so no acknowledgement message can be heard from the preferred parent for a certain period of time.

4 The proposed objective function

The proposed OF leverages the lifetime of the entire network. The load balanced OF (LB-OF) balances the data traffic by taking into account the number of children for each candidate parent.

4.1 Balancing the load traffic

As aforementioned, being a preferred parent for more children means more overhead and unbalanced load, that results in a drain its own energy much faster than other candidate parents. To solve this problem, a new metric has been proposed. The children set created in section 4.2 provides each preferred parent with the number of children it has. Based on that, the number of children in the rank calculation in formula (1) is considered.

Specifically, the parent with the least number of children will be elected as preferred parent. To this end, the balance has been achieved by declining the number of children of the overloaded bottleneck node. As a result, the majority of children (i.e., the shared nodes between A and B) will choose another preferred parent according to the lower rank, and surely has less number of children.

However, it is expected to increase the churn or oscillation as a result of changing the parent. It is a trade-off between unfairness and oscillation, however, this oscillation can be minimized in two techniques to enhance the stability:

a) using the number of children along with another metric(s)(e.g. ETX, number of hops, energy, etc., according to the application requirements).

b) Using the hysteresis threshold for the number of children(in a lexical manner)to switch from parent to another, the selected threshold depends on the application requirements.

4.2 A new utilization technique for DIO message

Generally, in the upward routes the root initiates the DODAG construction by sending the first DIO message. Once other nodes receive this DIO, they select the sender as a preferred parent, and then they start calculating their own ranks based on the assigned OF. After that, each node broadcasts its own DIO message (i.e. the updated DIO that contains the new calculated rank value) to all neighbors including the chosen preferred parent which sent the original DIO message. In the standard OFs, the preferred parent ignores the DIOs that come from its child based on the rank.

In this stage, the aim is to allow each parent to count its number of children to avoid later possible overloading situations. However, that is not possible in the upward routes (i.e., while maintaining the DODAG through DIOs), as the only control message that can be acknowledged by the destination is the Destination Advertisement Object (DAO) message in the downward routes to recognize the number of children for each parent.

Alternatively, setting up an acknowledgement mechanism between parent and children to count the number of children for each parent. However, this acknowledgement also brings an extra overhead for the entire network and subsequently increases the power consumption massively. To overcome this problem, LB-OF using a new technique is proposed as detailed below. In LB-OF algorithm, the received DIO from the child node is counted by the preferred parent node. Each DIO contains the IP address of the chosen preferred parent as detailed in section 4.3. Thus, for each received DIO, the node matches its own IP address with the preferred parent IP address which is inserted in the DIO message, then increments the number of children by ONE for this node if there is a matching.

Hence, this technique evades increasing any extra overhead, additionally, the coming DIOs from the child nodes has been utilized to allow each preferred parent to distinguish the number of its children during the DODAG construction stage to optimize the routing.

4.3 Proposed New Metric for Parent Selection

Typically, the DIO carries the RPL InstanceID, DODAG identifier, version number, Rank and the OF that has been used to calculate the rank, in addition to other identifiers [RFC6550]. This section introduces the number of child nodes as a new metric/constraint in the DAG Metric Container, which includes the selected parent address in the option field within the DIO message. The newly added information is 2 octets named by Child Node Count (CNC) which is per this document defined in the DAG Metric Container.

The Child Node Count (CNC) object is used to provide information related

to the number of child nodes in the DIO source node, and may be used as a metric or as constraint.

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

```
 0                   1                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Flags     |P|    CNC      |    CNC_MAX    |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
|                                                             |
+                                                             +
|                                                             |
+                    Parent Address                           +
|                                                             |
+                               +-+-+-+-+-+-+-+               +
|                               |                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
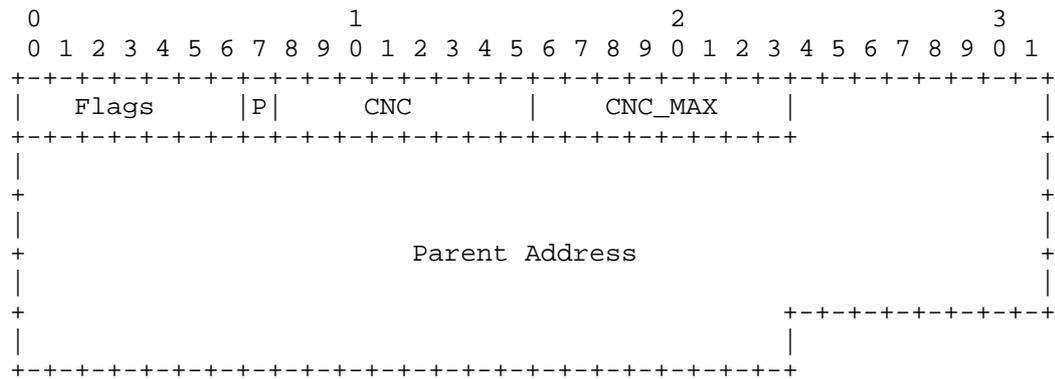
Figure 3: Child Node Count Object Body Format

Flags field (8 bits). The following one bits of the CNC object are currently defined:

'P'flag: Parent Address State. This, if set to 1, indicates there is a parent address field in the CNC object.

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

MAX_CNC: 8-bits. The Maximum Child Node Count is encoded in 8 bits. The MAX_CNC field indicates the maximum number of children a node can hold. This parameter is set by implementers based on neighbor cache entry or the size limit of routing table. Nodes should not hold child nodes more than MAX_CNC.

Parent Address (optional): 128-bit IPv6 address of parent node. This field is only present when the'P'flag is set to 1.

In the storing mode, DAO can be used for child nodes registration while No-PathDAO can be used for de-registration, and this gives a way to count the number of child nodes. Thus, to minimize traffic load, the

Parent Address field in the CNC object should not be present in the
storing mode.

In the non-storing mode, NS/NA could be an optional way for child node
counting. When the 'P' flag is set, the Parent Address in the CNC object
should be used for child node counting according to the technique
illustrated in section 4.2.

When this CNC metric is used, RANK computing reflects the ability of
each node to hold more child nodes. Also, a new way for the RANK
computing has been suggested: RANK = CNC / CNC_MAX * 255. A node with
smaller RANK has high priority to accept new child nodes, a node with
RANK = 255 should not hold new child nodes any more.

5 Security Considerations

Since the routing metrics/constraints are carried within RPL message,
the security routing mechanisms defined in [RFC6550] apply here.

6 IANA Considerations

IANA is requested to allocate a new value for the new metric type "CNC"
in the Routing Metric/Constraint Type in the DAG Metric Container.

7 References

7.1 Normative 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, March 2012.

    [RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing
               Protocol for Low-Power and Lossy Networks (RPL)", RFC
               6552, March 2012.

    [RFC6719] Gnawali, O. and P. Levis, "The Minimum Rank with
               Hysteresis Objective Function", RFC 6719, DOI
               10.17487/RFC6719, September 2012.

    [RFC6551] Vasseur, J., 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, March 2012.

7.2 Informative References

    [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
               IANA Considerations Section in RFCs", BCP 26, RFC 5226,
               May 2008.

    [Contiki] Contik O.S and cooja simulatoer http://www.contiki-os.org/

Appendix A.

               The protocol has been simulated with Cooja simulator based
               on Instant Contiki 2.7 operating system [Contiki].
               Collected results corroborate the superiority of our OF
               over the existing ones in terms of lifetime, power
               consumption and packet delivery ratio.

Authors' Addresses

               Mamoun Qasem (editor)
               Edinburgh Napier University
               10 Colinton Road, EH10 5DT, Edinburgh, UK

               Phone: +44 131 455 2772
               Email: m.qasem@napier.ac.uk

               Ahmed Al-Dubai
               Edinburgh Napier University
               10 Colinton Road, EH10 5DT, Edinburgh, UK

               Phone: +44 131 455 2796
               Email: a.al-dubai@napier.ac.uk

               Imed Romdhani
               Edinburgh Napier University
               10 Colinton Road, EH10 5DT, Edinburgh, UK

               Phone: +44 131 455 2726
               Email: i.romdhani@napier.ac.uk

               Baraq Ghaleb
               Edinburgh Napier University

10 Colinton Road, EH10 5DT, Edinburgh, UK

Email: b.ghaleb@napier.ac.uk

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

Phone: +86-15852944235
Email: houjianqiang@huawei.com

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

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