ROLL
Internet-Draft
Intended status: Standards Track
Expires: June 24, 2019

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

December 21, 2018

Root initiated routing state in RPL
draft-ietf-roll-dao-projection-05

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The "Routing Protocol for Low Power and Lossy Networks" [RFC6550] (LLN)(RPL) is a generic Distance Vector protocol that is well suited 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 to install and maintain projected routes (P-Routes) within its DODAG, along a selected set of nodes that may or may not include self, for a chosen duration. This potentially enables routes that are more optimized than those obtained with the distributed operation of RPL, either in terms of the size of a source-route header or in terms of path length, which impacts both the latency and the packet delivery ratio. P-routes may be installed in either Storing and Non-Storing Modes Instances of the classical RPL operation, resulting in potentially hybrid situations where the mode of some P-routes is different from that of the other routes in the RPL Instance.

P-Routes must be used with the parsimony to limit the amount of state that is installed in each device to fit within its resources, and to limit the amount of rerouted traffic to fit within the capabilities of the transmission links. The algorithm used to compute the paths and the protocol used to learn the topology of the network and the resources that are available in devices and in the network are out of scope for this document. Possibly with the assistance of a Path Computation Element ([PCE]) that could have a better visibility on the larger system, the root computes which segment could be optimized and uses this draft to install the corresponding P-Routes.

2. Terminology

2.1. BCP 14

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

2.2. Subset of a 6LoWPAN Glossary

This document often uses the following acronyms:

6BBR: 6LoWPAN Backbone Router

6LBR: 6LoWPAN Border Router

6LN: 6LoWPAN Node

6LR: 6LoWPAN Router

6CIO: Capability Indication Option

EARO: (Extended) Address Registration Option -- (E)ARO

EDAR: (Extended) Duplicate Address Request -- (E)DAR

EDAC: (Extended) Duplicate Address Confirmation -- (E)DAC

DAD: Duplicate Address Detection

DODAG: Destination-Oriented Directed Acyclic Graph

LLN: Low-Power and Lossy Network

NA: Neighbor Advertisement

NCE: Neighbor Cache Entry

ND: Neighbor Discovery

NDP: Neighbor Discovery Protocol

NS: Neighbor Solicitation

RPL: IPv6 Routing Protocol for LLNs (pronounced ripple) [RFC6550]

RA: Router Advertisement

RS: Router Solicitation

2.3. New Terms

P-Route: A route that is installed remotely by a RPL root.

2.4. References

In this document, readers will encounter terms and concepts that are discussed in the following documents:

- "Routing Protocol for Low Power and Lossy Networks" [RFC6550], and

- "Terminology in Low power And Lossy Networks" [RFC7102].

3. Extending RFC 6550

Section 6.7 of RPL [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. In Non-Storing Mode, the TIO option is used in the DAO message to indicate the immediate parent of a given path. The TIO applies to the Target options that immedially preceed it. Options may be factorized; multiple TIOs may be present to indicate multiple routes to the one or more contiguous addressed indicated in the Target Options that immediately precede the TIOs in the RPL message.

This specification introduces two new Control Message Options referred to as Route Projection Options (RPO). One RPO is the Information option (VIO) and the other is the Source-Routed VIO (SRVIO). The VIO installs a route on each hop along a P-Route (in a fashion analogous to RPL Storing Mode) whereas the SRVIO installs a source-routing state at the ingress node, which uses it to insert a routing header in a fashion similar to Non-Storing Mode.

Like the TIO, the RPOs MUST be preceded by one or more RPL Target Options to which they apply, and they can be factorized: multiple contiguous RPOs indicate alternate paths to the target(s).

3.1. 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 of routing information. This draft conforms the instance model as follows:

- 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 (say, Traffic Engineered) route between a particular pair of nodes then it SHOULD use a Local Instance from the ingress node of that path. A packet associated with that instance will be routed along that path and MUST NOT be placed over a Global Instance again. A packet that is placed on a Global Instance may be injected in the Local Instance based on node policy and the Local Instance paramenters.

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.

3.2. New RPL Control Message Options

The format of RPOs 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 | Path Sequence | Path Lifetime |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
.                                                               .
.                     Via Address 1                             .
.                                                               .
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                              ....                             .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
.                                                               .
.                     Via Address n                             .
.                                                               .
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Via Information option format

Option Type: 0x0A for VIO, 0x0B for SRVIO (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 255 (0xFF) represents infinity. A value of zero (0x00) indicates a loss of reachability. A DAO message that contains a Via Information option with a Path Lifetime of zero 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 RPO. Via Addresses are indicated in the order of the data path from the ingress to the egress nodes.

An RPO MUST contain at least one Via Address, and a Via Address MUST NOT be present more than once, otherwise the RPO MUST be ignored.

3.3. Projected DAO

This draft adds a capability to RPL whereby the root of a DODAG projects a route by sending an extended DAO message called a Projected-DAO (P-DAO) to an arbitrary router in the DODAG, indicating one or more sequence(s) of routers inside the DODAG via which the target(s) indicated in the Target Information Option(s) (TIO) can be reached.

A P-DAO is sent from a global address of the root to a global address of the recipient, and MUST be confirmed by a DAO-ACK, which is sent back to a global address of the root.

A P-DAO message MUST contain at least one TIO and at least one RPO following it. There can be at most one such sequence of TIOs and then RPOs.

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 RPO as opposed to a TIO. Also, a Path Lifetime of 0 in an RPO indicates that a route is to be removed.

There are two kinds of operation for the P-Routes, the Storing Mode and the Non-Storing Mode.

- o The Non-Storing Mode is discussed in Section 3.3.1. It uses an SRVIO that carries a list of Via Addresses to be used as a source-routed path to the target. The recipient of the P-DAO is the ingress router of the source-routed path. Upon a Non-Storing Mode P-DAO, the ingress router installs a source-routed state to the target and replies to the root directly with a DAO-ACK message.

- o The Storing Mode is discussed in Section 3.3.2. It uses a VIO with one Via Address per consecutive hop, from the ingress to the egress of the path, including the list of all intermediate routers in the data path order. 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 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.

In case of a forwarding error along a P-Route, an ICMP error is sent to the root with a new Code "Error in Projected Route" (See Section 5.2). The root can then modify or remove the P-Route. The "Error in Projected Route" message has the same format as the "Destination Unreachable Message", as specified in RFC 4443 [RFC4443]. The portion of the invoking packet that is sent back in the ICMP message SHOULD record at least up to the routing header if one is present, and the routing header 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 outter header then that whole 6LoRH information SHOULD be present in the ICMP message. The sender and exact operation depend on the Mode and is described in Section 3.3.1 and Section 3.3.2 respectively.

3.3.1. Non-Storing Mode P-Route

As illustrated in Figure 2, a P-DAO that carries an SRVIO 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 P-route to any packet for which the current destination either is the said target or can be reached 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 route indicated by an SRVIO may be loose, meaning that the node that owns the next listed Via Address is not necessarily a neighbor. Without proper loop avoidance mechanisms, the interaction of loose source routing and other mechanisms may effectively cause loops. In order to avoid those loops, if the router that installs a P-route does not have a connected route (a direct adjacency) to the next soure routed hop and fails to locate it as a neighbor or a neighbor of a neighbor, then it MUST ensure that it has another P-Route to the next loose hop under the control of the same route computation system, otherwise the P-DAO is rejected.

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 the case of a loose source-routed path, there MUST be either a neighbor that is adjacent to the loose next hop, on which case the packet s forwarded to that neighbor, or a source-routed path to the loose next hop; in the latter case, another encapsulation takes place and the process possibly recurses; otherwise the packet is dropped.

In order to add a source-routing header, 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 SRVIO, 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" [RFC8138] 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.

In case of a 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 [RFC6550]. Upon this message, the encapsulating node SHOULD stop using the source route path for a period of time and it SHOULD send an ICMP message with a Code "Error in Projected 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.

3.3.2. Storing-Mode P-Route

As illustrated in Figure 3, the Storing Mode projected iq used by 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                 |     |                  |
         o o   o  o   o  o  o o   o          |  ^       | Projected    .
        o  o o  o o    o   o   o  o  o       |  | DAO   | Route        .
        o   o    o  o     o  o    o  o  o    | ^        |              .
       o  o   o  o   o         o   o o       v | DAO    v              .
       o          o   LLN   o   o     o                                |
           o o   o        o     o              Loose Source Route Path |
        o       o      o    o                 From Root To Destination v
```

Figure 3: Projecting a route

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. 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 P-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.

In case of a forwarding error along a Storing Mode P-Route, the node that fails to forward SHOULD send an ICMP error with a code "Error in Projected Route" to the root. Failure to do so may result in packet loss and wasted resources along the P-Route that is broken.

4. Security Considerations

This draft uses messages that are already present in RPL [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.

TODO: should probably consider how P-DAO messages could be abused by a) rogue nodes b) via replay of messages c) if use of P-DAO messages could in fact deal with any threats?

5. IANA Considerations

5.1. New RPL Control Codes

This document extends the IANA registry created by RFC 6550 for RPL Control Codes as follows:

Code	Description	Reference
0x0A	Via	This document
0x0B	Source-Routed Via	This document

RPL Control Codes

This document is updating the registry created by RFC 6550 for the RPL 3-bit Mode of Operation (MOP) as follows:

MOP value	Description	Reference
5	Non-Storing mode of operation with P-Routes	This document
6	Storing mode of operation with P-Routes	This document

DIO Mode of operation

5.2. Error in Projected Route ICMPv6 Code

In some cases RPL will return an ICMPv6 error message when a message cannot be forwarded along a P-Route. This ICMPv6 error message is "Error in Projected 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 Projected Route", with a suggested code value of 8, to be confirmed by IANA.

6. Acknowledgments

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

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <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>.

[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>.

[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>.

7.2. 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-19 (work in progress), December 2018.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-10 (work in progress), December 2018.

[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, <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>.

Appendix A. Applications

A.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 as represented in Figure 4. In that mode, a RPL node indicates a parent-child relationship to the root, using a Destination Advertisement Object (DAO) that is unicast from the node directly to the root, and the root typically builds a source routed path to a destination down the DODAG by recursively concatenating this information.

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

Figure 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.

A.2. Transversal Routes in storing and non-storing modes

RPL is optimized for Point-to-Multipoint (P2MP) and Multipoint-to-Point (MP2P), whereby routes are always installed along the RPL DODAG respectively from and towards the DODAG Root. Transversal Peer to Peer (P2P) routes in a RPL network will generally suffer from some elongated (stretched) path versus the best possible path, since routing between 2 nodes always happens via a common parent, as illustrated in Figure 5:

- 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.

- 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    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  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.

Appendix B. Examples

B.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.

B.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)
                    +-----+
                       | P-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: P-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)
                    +-----+
                     ^ P-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  o
           S  A  B  C   D         o   o o
           o          o             o     o
                             LLN
```

Figure 9: P-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  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

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

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

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

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

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

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

ROLL
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2019

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

October 14, 2018

Efficient Route Invalidation
draft-ietf-roll-efficient-npdao-09

Abstract

This document describes the problems associated with NPDAO messaging used in RPL for route invalidation and signaling changes to improve route invalidation efficiency.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RPL [RFC6550] (Routing Protocol for Low power and lossy networks) specifies a proactive distance-vector based routing scheme. RPL has an optional messaging in the form of DAO (Destination Advertisement Object) messages using which the 6LBR (6Lo Border Router) and 6LR

(6Lo Router) can learn route towards the downstream nodes. In storing mode, DAO messages would result in routing entries been created on all intermediate 6LRs from the node’s parent all the way towards the 6LBR.

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

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

1.1. Requirements Language and Terminology

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

6LR: 6LoWPAN Router. This is an intermediate 6lowpan router which allows traffic routing through itself in a multihop 6lo network.

DAG: Directed Acyclic Graph. A directed graph having the property that all edges are oriented in such a way that no cycles exist.

DODAG: Destination-oriented DAG. A DAG rooted at a single destination, i.e., at a single DAG root with no outgoing edges.

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

DAO: Destination Advertisement Object. DAO messaging allows downstream routes to the nodes to be established.

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

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

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

DCO: Destination Cleanup Object, A new RPL control message type defined by this draft. DCO messaging improves proactive route invalidation in RPL.

Regular DAO: A DAO message with non-zero lifetime.

LLN: Low Power and Lossy Networks.

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

This document also uses terminology described in [RFC6550].

1.2. Current NPDAO messaging

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

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 6LBR. Node (A) is the common ancestor for (D) for paths through (B)-(G) and (C)-(H). When (D) switches from (B) to (C), RPL allows sending NPDAO to (B) and regular DAO to (C).

1.3. Why NPDAO is important?

Nodes in LLNs may be resource constrained. There is limited memory available and routing entry records are one of the primary elements occupying dynamic memory in the nodes. Route invalidation helps 6LR nodes to decide which entries could be discarded to better achieve resource utilization. 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 updates needs to be done for the routing tables entries of (C),(H),(A),(G), and (B) with destination (D),(E) and (F). Without efficient route invalidation, a 6LR may have to hold a lot of stale route entries.

2. Problems with current NPDAO messaging

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

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

2.2. Invalidate routes of dependent nodes

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

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

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

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

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

3. Requirements for the NPDAO Optimization

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

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

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

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

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

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

4. Proposed changes to RPL signaling

4.1. Change in RPL route invalidation semantics

As described in Section 1.2, 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 adds new pro-active route invalidation message called as "Destination Cleanup Object" (DCO) that originates at a common ancestor node between the new and old path. The common ancestor node generates a DCO in response to the change in the next-hop on receiving a regular DAO with updated path sequence for the target.

In Figure 1, when node D decides to switch the path from B to C, it sends a regular DAO to node C with reachability information containing 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), should generate a DCO which traverses downwards in the network.

4.2. Transit Information Option 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 specify 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 Transit Information option to contain "Invalidate previous route" (I) bit. This I-bit signals the common ancestor node to generate a DCO on behalf of the target node. The I-bit is carried in the transit information option which augments the reachability information for a given set of RPL Target(s). Transit information option should be carried in the DAO message with I-bit set in case route invalidation is sought for the correspondig 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 common ancestor node SHOULD generate a DCO message in response to this I-bit when it sees that the routing adjacencies have changed for the target. I-bit governs the ownership of the DCO message in a way that the target node is still in control of its own route invalidation.

4.3. Destination Cleanup Object (DCO)

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

```
     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|   Flags   |   Reserved    | DCOSequence   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     +                                                               +
     |                                                               |
     +                            DODAGID(optional)                  +
     |                                                               |
     +                                                               +
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Option(s)...
     +-+-+-+-+-+-+-+-+
```

Figure 3: DCO base object

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

K: The 'K' flag indicates that the recipient is expected to send a DCO-ACK back. If the DCO-ACK is not received even after setting the 'K', an implementation may choose to retry the DCO at a later time. The number of retries are implementation and deployment dependent. This document recommends using retries similar to what will be set for DAO-ACK handling.

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

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

Reserved: 8-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

DCOSequence: Incremented at each unique DCO message from a node and echoed in the DCO-ACK message. The initial DCOSequence can be chosen randomly by the node.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field is only present when the 'D' flag is set. This field is typically only present when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID. When a global RPLInstanceID is in use, this field need not be present. Unassigned bits of the DCO Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

4.3.1. Secure DCO

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

4.3.2. DCO Options

The DCO message MAY carry valid options. This specification allows for the DCO message to carry the following options:

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

The DCO carries a Target option and an associated Transit Information option with a lifetime of 0x00000000 to indicate a loss of reachability to that Target.

4.3.3. Path Sequence number in the DCO

A DCO message may contain a Path Sequence in the transit information option to identify the freshness of the DCO message. The Path Sequence in the DCO MUST use the same Path Sequence number present in the regular DAO message when the DCO is generated in response to DAO message. The DAO and DCO path sequence are picked from the same sequence number set. Thus if a DCO is received by a 6LR and subsequently a DAO is received with old seqeunce number, then the DAO should be ignored.

4.3.4. Destination Cleanup Option Acknowledgement (DCO-ACK)

The DCO-ACK message may be sent as a unicast packet by a DCO recipient in response to a unicast DCO message.

```
 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 |D|  Reserved   |  DCOSequence  |    Status     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                            DODAGID(optional)                  +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: DCO-ACK base object

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

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

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

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

Status: Indicates the completion. Status 0 is defined as unqualified acceptance in this specification. The remaining status values are reserved as rejection codes.

DODAGID (optional): 128-bit unsigned integer set by a DODAG root that uniquely identifies a DODAG. This field is only present when the ’D’ flag is set. This field is typically only present when a local RPLInstanceID is in use, in order to identify the DODAGID that is associated with the RPLInstanceID. When a global RPLInstanceID is in use, this field need not be present. Unassigned bits of the DCO-Ack Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

4.3.5. Secure DCO-ACK

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

4.4. Other considerations

4.4.1. Dependent Nodes invalidation

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

4.4.2. NPDAO and DCO in the same network

Even with the changed semantics, the current NPDAO mechanism in [RFC6550] can still be used, for example, when the route lifetime expiry of the target happens or when the node simply decides to gracefully terminate the RPL session on graceful node shutdown. Moreover a deployment can have a mix of nodes supporting the proposed DCO and the existing NPDAO mechanism.

4.4.3. DCO with multiple preferred parents

[RFC6550] allows a node to select multiple preferred parents for route establishment. Section 9.2.1 of [RFC6550] specifies, "All DAOs generated at the same time for the same Target MUST be sent with the same Path Sequence in the Transit Information". Thus a DAO message with the same path sequence MUST be sent to all the parents. Subsequently when route invalidation has to be initiated, RPL mentions that an NPDAO must be initiated with updated path sequence to all the routes to be invalidated.

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

5. Acknowledgements

Many thanks to Cenk Gundogan, Simon Duquennoy, Georgios Papadopoulous, Peter Van Der Stok for their review and comments.

6. IANA Considerations

IANA is requested to allocate new ICMPv6 RPL control codes in RPL [RFC6550] for DCO and DCO-ACK messages.

Code	Description	Reference
0x04	Destination Cleanup Object	This document
0x05	Destination Cleanup Object Acknowledgement	This document
0x84	Secure Destination Cleanup Object	This document
0x85	Secure Destination Cleanup Object Acknowledgement	This document

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.

7. Security Considerations

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

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

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

1. Unsecured: In this mode, it is expected that the RPL control messages are secured by other security mechanisms, such as link-layer security. In this mode, the RPL control messages, including DCO, DCO-ACK, do not have Security sections.
2. Preinstalled: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.

3. Authenticated: In this mode, RPL uses secure messages. Thus secure versions of DCO, DCO-ACK MUST be used in this mode.

8. Normative References

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

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

Appendix A. Example Messaging

A.1. Example DCO 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. Node 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 DCO(tgt=D,pathseq=pathseq(DAO)) 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 DCO and invalidates routing entry of target D and forwards the (un)reachability information downstream to B.
7. Similarly, B processes the DCO by invalidating the routing entry of target D and forwards the (un)reachability information downstream to D.

8. D ignores the DCO since the target is itself.
9. The propagation of the DCO will stop at any node where the node does not have an routing information associated with the target. If the routing information is present and the pathseq associated is not older, then still the DCO is dropped.

A.2. Example DCO Messaging with multiple preferred parents

```
                                   (6LBR)
                                     |
                                     |
                                     |
                                   (N11)
                                    / \
                                   /   \
                                  /     \
                               (N21)   (N22)
                                 /      / \
                                /      /   \
                               /      /     \
                            (N31)  (N32)  (N33)
                                :    |    /
                                 :   |   /
                                  :  |  /
                                   (N41)
```

Figure 5: Sample topology 2

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

1. (N41) sends DAO(tgt=N41,PS=x,I_flag=1) to (N32) and (N33). Here I_flag refers to the Invalidation flag and PS refers to Path Sequence in Transit Information option.
2. (N32) sends DAO(tgt=N41,PS=x,I_flag=1) to (N22). (N33) also sends DAO(tgt=N41,PS=x,I_flag=1) to (N22). (N22) learns multiple routes for the same destination (N41) through multiple next-hops. The route table at N22 should contain (Dst,NextHop,PS): { (N41,N32,x), (N41,N33,x) }.
3. (N22) sends DAO(tgt=N41,PS=x,I_flag=1) to (N11).
4. (N11) sends DAO(tgt=N41,PS=x,I_flag=1) to (6LBR). Thus the complete path is established.
5. (N41) decides to change preferred parent set from { N32, N33 } to { N31, N32 }.
6. (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N32). (N41) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N31).
7. (N32) sends DAO(tgt=N41,PS=x+1,I_flag=1) to (N22). (N22) has multiple routes to destination (N41). It sees that a new path

sequence for Target=N41 is received and thus it waits for pre-determined time period to invalidate another route {(N41),(N33),x}. After time period, (N22) sends DCO(tgt=N41,PS=x+1) to (N33).

Authors' Addresses

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

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

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

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

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

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

Zhen Cao
Huawei
W Chang'an Ave
Beijing
China

Email: zhencao.ietf@gmail.com

ROLL Working Group M. Robles
Internet-Draft Aalto
Updates: 6553, 6550, 8138 (if approved) M. Richardson
Intended status: Standards Track SSW
Expires: July 27, 2019 P. Thubert
Cisco
January 23, 2019

Using RPL Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane

draft-ietf-roll-useofrplinfo-24

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 RFC 6553 (RPL Option Type), RFC 6554 (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 RFC 6553 adding a change to the RPL Option Type. Additionally, this document updates RFC 6550 to indicate about this change and updates RFC8138 as well to consider the new Option Type when 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. RFC 6553 [RFC6553] defines the "RPL option" (RPI), carried within the IPv6 Hop-by-Hop header to quickly identify inconsistencies (loops) in the routing topology. RFC 6554 [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 implementors agree when artifacts are necessary, or when they can be safely omitted, or removed.

An interim meeting went through the 24 cases defined here to discover if there were any shortcuts, and this document is the result of that discussion. This document clarifies examples that intend to illustrate the result of the normative language in RFC8200 and RFC6553. In other words, the examples are intended to be normative explanation of the results of executing that language.

A Routing Header Dispatch for 6LoWPAN (6LoRH)([RFC8138]) defines a mechanism for compressing RPL Option information and Routing Header type 3 [RFC6554], as well as an efficient IPv6-in-IPv6 technique.

1.1. Overview

The rest of the document is organized as follows: Section 2 describes the used terminology. Section 3 describes the updates to RFC6553, RFC6550 and RFC 8138. Section 4 provides the reference topology used for the uses cases. Section 5 describes the uses cases included. Section 6 describes the storing mode cases and section 7 the non-storing mode cases. Section 8 describes the operational considerations for the proposed change on RPL Option type. Section 9 depicts the 6LoRH Compression cases, section 10 the IANA considerations and then section 11 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: LBR, LLN, RPL, RPL Domain and ROLL.

RPL-node: A device which implements RPL, thus the device is RPL-aware. Please note that the device can be found inside the LLN or outside LLN. In this document a RPL-node which is a leaf of a (Destination Oriented Directed Acyclic Graph) DODAG is called RPL-aware-leaf (Raf).

RPL-not-capable: 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. In this document a not-RPL-aware node which is a leaf of a DODAG is called not-RPL-aware-leaf (~Raf).

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.

6LR, 6LBR are defined in [RFC6775].

Flag Day: A transition that involves have a network with different values of RPL Option Type. Thus the network do not work correctly.

Hop-by-hop IPv6-in-IPv6 headers: The term "hop-by-hop IPv6-in-IPv6" header refers to: adding a header that originates from a node to an adjacent node, using the addresses (usually the GUA or ULA, but could use the link-local addresses) of each node. If the packet must traverse multiple hops, then it must be decapsulated at each hop, and then re-encapsulated again in a similar fashion.

3. Updates to RFC6553, RFC6550 and RFC 8138

3.1. Updates to RFC 6553

This modification is required to be able to send, for example, IPv6 packets from a RPL-aware-leaf to a not-RPL-aware node through Internet (see Section 6.2.1), without requiring IPv6-in-IPv6 encapsulation.

[RFC6553] states as shown below, that in the Option Type field of the RPL Option header, the two high order bits must be set to '01' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node must discard the packet if it doesn't recognize the option type, and the third bit indicates that the Option Data may change in route. The remaining bits serve as the option type.

Hex Value	Binary Value act	chg	rest	Description	Reference
0x63	01	1	00011	RPL Option	[RFC6553]

Figure 1: Option Type in RPL Option.

Recent changes in [RFC8200] (section 4, page 8), states: "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". Processing of the Hop-by-Hop Options header (by IPv6 intermediate nodes) is now optional, but if they are configured to process the header, and if such nodes encounter an option with the first two bits set to 01, they will drop the packet (if they conform to [RFC8200]). Host systems should do the same, irrespective of the configuration.

Based on that, if an IPv6 (intermediate) node (RPL-not-capable) receives a packet with an RPL Option, it should ignore the HBH 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 RPL-aware-leaf to Internet (see Section 6.2.1).

Thus, this document updates the Option Type field to: 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 remaining bits serve as the option type and remain as 0x3. 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 [RFC6553] RPL Hop-by-Hop option known as RPI.

This is a significant update to [RFC6553]. [RFCXXXX] represents this document.

Hex Value	Binary Value act	chg	rest	Description	Reference
0x23	00	1	00011	RPL Option	[RFCXXXX]

Figure 2: Revised Option Type in RPL Option.

This change creates a flag day for existing networks which are currently using 0x63 as the RPI value. A move to 0x23 will not be understood by those networks. It is suggested that implementations accept both 0x63 and 0x23 when processing.

In the cases where a forwarding node is forwarding traffic that is not addressed directly to it (such as when the outer IPv6-in-IPv6 header is not a Link-Local address), then RFC8200 forbids changing the RPI type code when forwarding.

When forwarding traffic that is wrapped in Link-Local IPv6-in-IPv6 headers, the forwarding node is in effect originating new packets, and it MAY make a choice as to whether to use the old (0x63) RPI Type code, or the new (0x23) RPI Type code. In that situation, implementations SHOULD use the same value as was received. This allows to the network to be incrementally upgraded, and in some cases may allow the DODAG root to know which parts of the network are upgraded.

A network which is switching from straight 6lowpan compression mechanism to those described in [RFC8138] will experience a flag day in the data compression anyway, and if possible this change can be deployed at the same time.

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 therefore no longer a necessity to remove the artifacts when sending traffic to the Internet. This change clarifies when to use an IPv6-in-IPv6 header, and how to address them: The Hop-by-Hop Options Header containing the RPI option SHOULD always be added when 6LRs originate packets (without IPv6-in-IPv6 headers), and IPv6-in-IPv6 headers SHOULD always be added when a 6LR find that it needs to insert a Hop-by-Hop Options Header containing the RPI 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.

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].

In the non-storing case, dealing with non-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 non-RPL aware leaf nodes because it will receive a DAO about that node from the 6LR immediately above that non-RPL aware node. This means that the non-storing mode case can avoid ever using hop-by-hop IPv6-in-IPv6 headers for traffic originating from the root to leafs.

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.

3.2. Updates to RFC 8138

RPI-6LoRH header provides a compressed form for the RPL RPI [RFC8138] in section 6. A node that is decompressing this header MUST decompress using the RPL 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 DODAG Configuration Option described in the next section. 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 options.

In Storing Mode, for the examples of Flow from RPL-aware-leaf to non-RPL-aware-leaf and non-RPL-aware-leaf to non-RPL-aware-leaf comprise an IPv6-in-IPv6 and RPI compression headers. The use of the IPv6-in-IPv6 header is MANDATORY in this case, and it SHOULD be compressed with [RFC8138] section 7.

```
+--+-----+---+--------------+-----------+-----------+-----------+
|1 | 0|0 |TSE| 6LoRH Type 6 | Hop Limit | RPI-6LoRH |LOWPAN IPHC|
+--+-----+---+--------------+-----------+-----------+-----------+
```

Figure 3: Critical IPv6-in-IPv6 (RPI).

3.3. Updates to RFC 6550: 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 (0x23) and old RPI (0x63) nodes, this section defines a flag in the DIO Configuration Option, to indicate when then new RPI value can be safely used. Without this, there could be a mix of new nodes (which understand 0x23 and 0x63), and old nodes (which understand 0x63 only). A new node would not know if it was safe to use 0x23.

This is done via a DODAG Configuration Option flag which will propagate through the network. 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 (0x63) value.

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.

The DODAG Configuration Option has a Flag field which is modified by this document. Currently, the DODAG Configuration Option in [RFC6550] states: "the unused bits MUST be initialize to zero by the sender and MUST be ignored by the receiver".

Bit number three of the flag field in the DODAG Configuration option is to be used as follows:

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 4: DODAG Configuration Option Flag to indicate the RPI-flag-day.

In case of rebooting, the node (6LN or 6LR) does not remember if the flag is set, so DIO messages would be set with the flag unset until a DIO is received with the flag set.

4. Sample/reference topology

A RPL network in general is composed of a 6LBR (6LoWPAN Border Router), Backbone Router (6BBR), 6LR (6LoWPAN Router) and 6LN (6LoWPAN Node) as leaf logically organized in a DODAG structure.

Figure 4 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, thus the 6BBR is not present in the figure.

The 6LN leaves (Raf - "RPL aware leaf"-) marked as (F, H and I) are RPL nodes with no children hosts.

The leafs marked as ~Raf "not-RPL aware leaf" (G and J) are devices which do not speak RPL at all (not-RPL-aware), but uses Router-Advertisements, 6LowPAN DAR/DAC and efficient-ND only to participate in the network [RFC6775]. In the document these leafs (G and J) are also referred to as an IPv6 node.

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  |         |             |
          |       |    +------       |         |             |
          +---|---+    |     +---|---+         |             |
              |        |         |             |             |
              |        |         +--+          |             |
              |        |            |          |             |
              |        |            |          |             |
              |        |            |        I |          J  |
           F  |        | G          | H        |             |
        +-----+-+    +-|-----+  +---|--+   +---|---+     +---|---+
        |  Raf  |    | ~Raf  |  | Raf  |   |  Raf  |     | ~Raf  |
        |  6LN  |    |  6LN  |  | 6LN  |   |  6LN  |     |  6LN  |
        +-------+    +-------+  +------+   +-------+     +-------+
```

Figure 5: A reference RPL Topology.

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 5.

```
+--------------+
| Upper Layers |
|              |
+--------------+
|   RPL        |
|              |
+--------------+
|   ICMPv6     |
|              |
+--------------+
|   IPv6       |
|              |
+--------------+
|   6LoWPAN    |
|              |
+--------------+
|   PHY-MAC    |
|              |
+--------------+
```

Figure 6: RPL Stack.

RPL supports two modes of Downward traffic: in storing mode (RPL-SM), it is fully stateful; in non-storing (RPL-NSM), 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 storing and non-storing nodes is not supported with the current specifications at the time of writing this document.

5. 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.

This document assumes that the LLN is using the no-drop RPI option (0x23).

The uses cases describe the communication between RPL-aware-nodes, with the root (6LBR), and with Internet. This document also describe the communication between nodes acting as leaves that do not understand RPL, but are part of the LLN. these nodes are named as not-RPL-aware-leaf, mentioned previously. (e.g. Section 6.1.4 Flow from not-RPL-aware-leaf to root) This document describes also how is the communication inside of the LLN when it has the final destination addressed outside of the LLN e.g. with destination to Internet. (e.g. Section 6.2.3 Flow from not-RPL-aware-leaf to Internet)

The uses cases comprise as follow:

Interaction between Leaf and Root:

RPL-aware-leaf to root

root to RPL-aware-leaf

not-RPL-aware-leaf to root

root to not-RPL-aware-leaf

Interaction between Leaf and Internet:

RPL-aware-leaf to Internet

Internet to RPL-aware-leaf

not-RPL-aware-leaf to Internet

Internet to not-RPL-aware-leaf

Interaction between Leafs:

RPL-aware-leaf to RPL-aware-leaf (storing and non-storing)

RPL-aware-leaf to not-RPL-aware-leaf (non-storing)

not-RPL-aware-leaf to RPL-aware-leaf (storing and non-storing)

not-RPL-aware-leaf to not-RPL-aware-leaf (non-storing)

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]. Extensions may not be added or removed except by the sender or the receiver.

However, unlike [RFC6553], the Hop-by-Hop Option Header used for the RPI artifact has the first two bits set to '00'. This means that the RPI artifact will be ignored when received by a host or router that does not understand that option (Section 4.2 [RFC8200]).

This means that when the no-drop RPI option code 0x23 is used, a packet that leaves the RPL domain of an LLN (or that leaves the LLN entirely) will not be discarded when it contains the [RFC6553] RPL Hop-by-Hop option known as RPI. Thus, the RPI Hop-by-Hop option is left in place even if the end host does not understand it.

NOTE: There is some possible security risk when the RPI information is released to the Internet. At this point this is a theoretical situation; no clear attack has been described. At worst, it is clear that the RPI option would waste some network bandwidth when it escapes. This is traded off against the savings in the LLN by not having to encapsulate the packet in order to remove the artifact.

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 SHOULD 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 (RH3 or RPI 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 RH3 or RPI 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. When it does so, 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 RPI and RH3 headers may be modified in very specific ways by routers on the path of the packet without the need to add to 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.

RPI MUST be present in every single RPL data packet. There is one exception in non-storing mode: when a packet is going down from the root the RPI MAY be omitted. The rational is that in a downward non-storing mode, the entire route is written, so there can be no loops by construction, nor any confusion about which forwarding table to use (as the root has already made all routing decisions). However, there are still cases, such as in 6tisch, where the instanceID portion of the RPI header may still be needed [RFC8180] to pick an appropriate priority or channel at each hop.

Prior to [RFC8138], there was significant interest in removing the RPI for downward flows in non-storing mode. The exception covered a very small number of cases, and causes significant interoperability challenges, yet costed significant code and testing complexity. 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].

The earlier examples are more extensive to make sure that the process is clear, while later examples are more concise.

6. Storing mode

In storing mode (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 itemizes which headers are needed in each of the following scenarios. It indicate if an IPv6-in-IPv6 header MUST be inserted, and whether the destination address of the IPv6-in-IPv6 header is the next hop, or the final target address. There are these possible situations: hop-by-hop necessary (indicated by "hop"), or final target address possible (indicated by "tgt"). In all cases hop by hop MAY be used rather than the final target address.

In cases where no IPv6-in-IPv6 header is needed, the column states as "No".

In all cases the RPI headers are needed, since it identifies inconsistencies (loops) in the routing topology. In all cases the RH3 is not needed because it is not used in storing mode.

In each case, 6LR_i are the intermediate routers from source to destination. "1 <= i <= n", n is the number of routers (6LR) that the packet go through from source (6LN) to destination.

The leaf can be a router 6LR or a host, both indicated as 6LN (see Figure 5).

Interaction between	Use Case	IPv6-in-IPv6	v6-in-v6 dst
Leaf - Root	Raf to root	No	No
	root to Raf	No	No
	root to ~Raf	No	No
	~Raf to root	MUST	root
Leaf - Internet	Raf to Int	No	No
	Int to Raf	MUST	tgt (Raf)
	~Raf to Int	MUST	root
	Int to ~Raf	MUST	hop
Leaf - Leaf	Raf to Raf	No	No
	Raf to ~Raf	No	No
	~Raf to Raf	MUST	tgt (Raf)
	~Raf to ~Raf	Yes	hop

Figure 7: IPv6-in-IPv6 encapsulation in Storing mode.

6.1. Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in storing mode (SM) between,

RPL-aware-leaf to root

root to RPL-aware-leaf

not-RPL-aware-leaf to root

root to not-RPL-aware-leaf

6.1.1. SM: Example of Flow from RPL-aware-leaf to root

In storing mode, RFC 6553 (RPI) is used to send RPL Information instanceID and rank information.

As stated in Section 16.2 of [RFC6550] a RPL-aware-leaf node does not generally issue DIO messages; a leaf node accepts DIO messages from upstream. (When the inconsistency in routing occurs, a leaf node will generate a DIO with an infinite rank, to fix it). It may issue DAO and DIS messages though it generally ignores DAO and DIS messages.

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR_i --> root(6LBR)

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

As it was mentioned in this document 6LRs, 6LBR are always full-fledged RPL routers.

The 6LN (Node F) inserts the RPI header, and sends the packet to 6LR (Node E) which decrements the rank in 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 header can be removed by the 6LBR because the packet is addressed to the 6LBR. The 6LN must know that it is communicating with the 6LBR to make use of this scenario. The 6LN can know the address of the 6LBR because it knows the address of the root via the DODAGID in the DIO messages.

Header	6LN	6LR_i	6LBR
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 1: Storing: Summary of the use of headers from RPL-aware-leaf to root

6.1.2. SM: Example of Flow from root to RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RPL-aware-leaf (6LN)

For example, a communication flow could be: Node A root(6LBR) --> Node B --> Node D --> Node F

In this case the 6LBR inserts RPI header and sends the packet down, the 6LR is going to increment the rank in RPI (it examines the instanceID to identify the right forwarding table), the packet is processed in the 6LN and the RPI removed.

No IPv6-in-IPv6 header is required.

Header	6LBR	6LR_i	6LN
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 2: Storing: Summary of the use of headers from root to RPL-aware-leaf

6.1.3. SM: Example of Flow from root to not-RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR_i --> not-RPL-aware-leaf (IPv6)

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

As the RPI extension can be ignored by the not-RPL-aware leaf, this situation is identical to the previous scenario.

Header	6LBR	6LR_i	IPv6
Inserted headers	RPI	--	--
Removed headers	--	--	--
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	RPI (Ignored)

Table 3: Storing: Summary of the use of headers from root to not-RPL-aware-leaf

6.1.4. SM: Example of Flow from not-RPL-aware-leaf to root

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR_1 --> 6LR_i --> root (6LBR)

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

When the packet arrives from IPv6 node (Node G) to 6LR_1 (Node E), the 6LR_1 will insert a RPI header, encapsulated in a IPv6-in-IPv6 header. The IPv6-in-IPv6 header can be addressed to the next hop (Node B), or to the root (Node A). The root removes the header and processes the packet.

Header	IPv6	6LR_1	6LR_i	6LBR
Inserted headers	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(RPI)(1)	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	IPv6-in-IPv6(RPI)(1)	--
Modified headers	--	--	IPv6-in-IPv6(RPI)(2)	--
Untouched headers	--	--	--	--

Table 4: Storing: Summary of the use of headers from not-RPL-aware-leaf to root. (1) Case where the IPv6-in-IPv6 header is addressed to the next hop (Node B). (2) Case where the IPv6-in-IPv6 header is addressed to the root (Node A)

6.2. Storing Mode: Interaction between Leaf and Internet.

In this section is described the communication flow in storing mode (SM) between,

RPL-aware-leaf to Internet

Internet to RPL-aware-leaf

not-RPL-aware-leaf to Internet

Internet to not-RPL-aware-leaf

6.2.1. SM: Example of Flow from RPL-aware-leaf to Internet

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.

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR_i --> root (6LBR) --> Internet

For example, the communication flow could be: Node F --> Node D --> Node B --> Node A root(6LBR) --> Internet

No IPv6-in-IPv6 header is required.

Note: In this use case it is used a node as leaf, but this use case can be also applicable to any RPL-node type (e.g. 6LR)

Header	6LN	6LR_i	6LBR	Internet
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	RPI	RPI (Ignored)

Table 5: Storing: Summary of the use of headers from RPL-aware-leaf to Internet

6.2.2. SM: Example of Flow from Internet to RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RPL-aware-leaf (6LN)

For example, a communication flow could be: Internet --> Node A root(6LBR) --> Node B --> Node D --> Node F

When the packet arrives from Internet to 6LBR the RPI header is added in a outer IPv6-in-IPv6 header and sent to 6LR, which modifies the rank in the RPI. When the packet arrives at 6LN the RPI header is removed and the packet processed.

Header	Internet	6LBR	6LR_i	6LN
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)	--
Untouched headers	--	--	--	--

Table 6: Storing: Summary of the use of headers from Internet to RPL-aware-leaf

6.2.3. SM: Example of Flow from not-RPL-aware-leaf to Internet

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR_1 --> 6LR_i -->root (6LBR) --> Internet

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

The 6LR_1 (i=1) node will add an IPv6-in-IPv6(RPI) header addressed either to the root, or hop-by-hop such that the root can remove the RPI header before passing upwards. The IPv6-in-IPv6 addressed to the root cause less processing overhead. On the other hand, with hop-by-hop the intermediate routers can check the routing tables for a better routing path, thus it could be more efficient and faster. Implementation should decide which approach to take.

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.

Header	IPv6	6LR_1	6LR_i [i=2,..,n]_	6LBR	Internet
Inserted headers	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(RPI)(2)	--	--
Removed headers	--	--	IPv6-in-IPv6(RPI)(2)	IPv6-in-IPv6(RPI)(1)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)(1)	--	--
Untouched headers	--	--	--	--	--

Table 7: Storing: Summary of the use of headers from not-RPL-aware-leaf to Internet. (1) Case when packet is addressed to the root. (2) Case when the packet is addressed hop-by-hop.

6.2.4. SM: Example of Flow from Internet to non-RPL-aware-leaf.

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> not-RPL-aware-leaf (IPv6)

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

The 6LBR will have to add an RPI header within an IPv6-in-IPv6 header. The IPv6-in-IPv6 is addressed to the not-RPL-aware-leaf, leaving the RPI inside.

Note that there is a requirement that the final node be able to remove one or more IPv6-in-IPv6 headers which are all addressed to it, mentioned in [I-D.thubert-roll-unaware-leaves] :

"RPL data packets are often encapsulated using IP in IP. The 6LN MUST be able to decapsulate a packet when it is the destination of the outer header and process correctly the inner header."

The 6LBR MAY set the flow label on the inner IPv6-in-IPv6 header to zero in order to aid in compression.

Header	Internet	6LBR	6LR_i	IPv6
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)- RPI (Ignored)
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)	--
Untouched headers	--	--	--	--

Table 8: Storing: Summary of the use of headers from Internet to non-RPL-aware-leaf

6.3. Storing Mode: Interaction between Leaf and Leaf

In this section is described the communication flow in storing mode (SM) between,

RPL-aware-leaf to RPL-aware-leaf

RPL-aware-leaf to not-RPL-aware-leaf

not-RPL-aware-leaf to RPL-aware-leaf

not-RPL-aware-leaf to not-RPL-aware-leaf

6.3.1. SM: Example of Flow from RPL-aware-leaf to RPL-aware-leaf

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:

6LN --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> 6LN

For example, a communication flow could be: Node F --> Node D --> Node B --> Node E --> Node H

6LR_ia (Node D) are the intermediate routers from source to the common parent (6LR_x) (Node B) In this case, "1 <= ia <= n", n is the number of routers (6LR) that the packet go through from 6LN (Node F) to the common parent (6LR_x).

6LR_id (Node E) are the intermediate routers from the common parent (6LR_x) (Node B) to destination 6LN (Node H). In this case, "1 <= id <= m", m is the number of routers (6LR) that the packet go through from the common parent (6LR_x) to destination 6LN.

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 of RPI is changed (from increasing to decreasing the rank).

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. This may be done regardless of where the destination is, as the included RPI will be ignored by the receiver.

Header	6LN src	6LR_ia	6LR_x (common parent)	6LR_id	6LN dst
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	RPI
Re-added headers	--	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Untouched headers	--	--	--	--	--

Table 9: Storing: Summary of the use of headers for RPL-aware-leaf to RPL-aware-leaf

6.3.2. SM: Example of Flow from RPL-aware-leaf to non-RPL-aware-leaf

In this case the flow comprises:

6LN --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> not-RPL-aware 6LN (IPv6)

For example, a communication flow could be: Node F --> Node D --> Node B --> Node E --> Node G

6LR_ia are the intermediate routers from source (6LN) to the common parent (6LR_x) In this case, "1 <= ia <= n", n is the number of routers (6LR) that the packet go through from 6LN to the common parent (6LR_x).

6LR_id (Node E) are the intermediate routers from the common parent (6LR_x) (Node B) to destination not-RPL-aware 6LN (IPv6) (Node G). In this case, "1 <= id <= m", m is the number of routers (6LR) that the packet go through from the common parent (6LR_x) to destination 6LN.

This situation is identical to the previous situation Section 6.3.1

Header	6LN src	6LR_ia	6LR_x(common parent)	6LR_id	IPv6
Inserted headers	RPI	--	--	--	--
Removed headers	--	--	--	--	--
Re-added headers	--	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Untouched headers	--	--	--	--	RPI(Ignored)

Table 10: Storing: Summary of the use of headers for RPL-aware-leaf to non-RPL-aware-leaf

6.3.3. SM: Example of Flow from not-RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> 6LN

For example, a communication flow could be: Node G --> Node E --> Node B --> Node D --> Node F

6LR_ia (Node E) are the intermediate routers from source (not-RPL-aware 6LN (IPv6)) (Node G) to the common parent (6LR_x) (Node B). In this case, "1 <= ia <= n", n is the number of routers (6LR) that the packet go through from source to the common parent.

6LR_id (Node D) are the intermediate routers from the common parent (6LR_x) (Node B) to destination 6LN (Node F). In this case, "1 <= id <= m", m is the number of routers (6LR) that the packet go through from the common parent (6LR_x) to destination 6LN.

The 6LR_ia (ia=1) (Node E) receives the packet from the the IPv6 node (Node G) and inserts and the RPI header encapsulated in IPv6-in-IPv6 header. The IPv6-in-IPv6 header is addressed to the destination 6LN (Node F).

Header	IPv6	6LR_ia	common parent (6LRx)	6LR_id	6LN
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(RPI)	--
Untouched headers	--	--	--	--	--

Table 11: Storing: Summary of the use of headers from not-RPL-aware-leaf to RPL-aware-leaf

6.3.4. SM: Example of Flow from not-RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6 src)--> 6LR_1--> 6LR_ia --> 6LR_id --> not-RPL-aware 6LN (IPv6 dst)

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

Internal nodes 6LR_ia (e.g: Node E or Node B) are the intermediate routers from the not-RPL-aware source (Node G) to the root (6LBR) (Node A). In this case, "1 < ia <= n", n is the number of routers (6LR) that the packet go through from IPv6 src to the root.

6LR_id (C) are the intermediate routers from the root (Node A) to the destination Node J. In this case, "1 <= id <= m", m is the number of routers (6LR) that the packet go through from the root to destination (IPv6 dst).

Note that this flow is identical to Section 6.3.3, except for where the IPv6-in-IPv6 header is inserted.

The 6LR_1 (Node E) receives the packet from the the IPv6 node (Node G) and inserts the RPI header (RPI), encapsulated in an IPv6-in-IPv6 header. The IPv6-in-IPv6 header is addressed to the final destination (Node J).

Header	IPv6 src	6LR_1	6LR_ia	6LR_m	IPv6 dst
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--	--
Removed headers	--	--	--	--	IPv6-in-IPv6(RPI), RPI Ignored
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)	IPv6-in-IPv6(RPI)	--
Untouched headers	--	--	--	--	--

Table 12: Storing: Summary of the use of headers from not-RPL-aware-leaf to non-RPL-aware-leaf

7. 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 non-RPL aware nodes. Only the 6LBR needs to act if compensation is necessary for non-RPL aware receivers.

The following table summarizes what headers are needed in the following scenarios, and indicates when the RPI, RH3 and IPv6-in-IPv6 header are to be inserted. There are these possible situations: target destination address possible (indicated by "tgt"), to a 6LR, to a 6LN or to the root. In cases where no IPv6-in-IPv6 header is needed, the column states as "No".

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 3). In the Figure the (1) indicates a 6tisch case [RFC8180], where the instanceID portion of the RPI header may still be needed to pick an appropriate priority or channel at each hop.

Interaction between	Use Case	RPI	RH3	v6-in-v6	v6-in-v6 dst
Leaf - Root	Raf to root	Yes	No	No	No
	root to Raf	Opt	Yes	No	No
	root to ~Raf	No(1)	Yes	MUST	6LR
	~Raf to root	Yes	No	MUST	root
Leaf - Internet	Raf to Int	Yes	No	MUST	root
	Int to Raf	No(1)	Yes	MUST	tgt
	~Raf to Int	Yes	No	MUST	root
	Int to ~Raf	No(1)	Yes	MUST	6LR
Leaf - Leaf	Raf to Raf	Yes	Yes	MUST	root/tgt
	Raf to ~Raf	Yes	Yes	MUST	root/6LR
	~Raf to Raf	Yes	Yes	MUST	root/6LN
	~Raf to ~Raf	Yes	Yes	MUST	root/6LR

(1)-6tisch networks may need the RPI information.

Figure 8: Headers needed in Non-Storing mode: RPI, RH3, IPv6-in-IPv6 encapsulation.

7.1. Non-Storing Mode: Interaction between Leaf and Root

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RPL-aware-leaf to root

root to RPL-aware-leaf

not-RPL-aware-leaf to root

root to not-RPL-aware-leaf

7.1.1. Non-SM: Example of Flow from RPL-aware-leaf to root

In non-storing mode the leaf node uses default routing to send traffic to the root. The RPI header MUST be included since contains the rank information, which is used to avoid/detect loops.

RPL-aware-leaf (6LN) --> 6LR_i --> root(6LBR)

For example, a communication flow could be: Node F --> Node D --> Node B --> Node A (root)

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet go through from source (6LN) to destination (6LBR).

This situation is the same case as storing mode.

Header	6LN	6LR_i	6LBR
Inserted headers	RPI	--	--
Removed headers	--	--	RPI
Re-added headers	--	--	--
Modified headers	--	RPI	--
Untouched headers	--	--	--

Table 13: Non Storing: Summary of the use of headers from RPL-aware-leaf to root

7.1.2. Non-SM: Example of Flow from root to RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RPL-aware-leaf (6LN)

For example, a communication flow could be: Node A (root) --> Node B --> Node D --> Node F

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet go through from source (6LBR) to destination (6LN).

The 6LBR inserts an RH3, and a RPI header. No IPv6-in-IPv6 header is necessary as the traffic originates with an RPL aware node, the 6LBR. The destination is known to RPL-aware because, the root knows the whole topology in non-storing mode.

Header	6LBR	6LR_i	6LN
Inserted headers	(opt: RPI), RH3	--	--
Removed headers	--	--	RH3, (opt: RPI)
Re-added headers	--	--	--
Modified headers	--	RH3	--
Untouched headers	--	--	--

Table 14: Non Storing: Summary of the use of headers from root to RPL-aware-leaf

7.1.3. Non-SM: Example of Flow from root to not-RPL-aware-leaf

In this case the flow comprises:

root (6LBR) --> 6LR_i --> not-RPL-aware-leaf (IPv6)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet go through from source (6LBR) to destination (IPv6).

In 6LBR the RH3 is added, it is modified at each intermediate 6LR (6LR_1 and so on) and it is fully consumed in the last 6LR (6LR_n), but left there. If RPI is left by the previous 6LR, then the IPv6 node which does not understand the RPI, will ignore it (following RFC8200), thus encapsulation is not necessary. Due the complete knowledge of the topology at the root, the 6LBR may optionally address the IPv6-in-IPv6 header to the last 6LR, such that it is removed prior to the IPv6 node. Please see Section 8 for clarification of use of IPv6-in-IPv6 encapsulation.

Header	6LBR	6LR_i(i=1)	6LR_n(i=n)	IPv6
Inserted headers	(opt: RPI), RH3	--	--	--
Removed headers	--	--	RH3	--
Re-added headers	--	--	--	--
Modified headers	--	(opt: RPI), RH3	(opt: RPI)	--
Untouched headers	--	--	--	opt: RPI

Table 15: Non Storing: Summary of the use of headers from root to not-RPL-aware-leaf

7.1.4. Non-SM: Example of Flow from not-RPL-aware-leaf to root

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR_1 --> 6LR_i --> root (6LBR)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet go through from source (IPv6) to destination (6LBR). For example, 6LR_1 (i=1) is the router that receives the packets from the IPv6 node.

In this case the RPI is added by the first 6LR (6LR1) (Node E), encapsulated in an IPv6-in-IPv6 header, and is modified in the following 6LRs. The RPI and entire packet is consumed by the root.

Header	IPv6	6LR_1	6LR_i	6LBR
Inserted headers	--	IPv6-in-IPv6(RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6(RPI)	--
Untouched headers	--	--	--	--

Table 16: Non Storing: Summary of the use of headers from not-RPL-aware-leaf to root

7.2. Non-Storing Mode: Interaction between Leaf and Internet

This section will describe the communication flow in Non Storing Mode (Non-SM) between:

RPL-aware-leaf to Internet

Internet to RPL-aware-leaf

not-RPL-aware-leaf to Internet

Internet to not-RPL-aware-leaf

7.2.1. Non-SM: Example of Flow from RPL-aware-leaf to Internet

In this case the flow comprises:

RPL-aware-leaf (6LN) --> 6LR_i --> root (6LBR) --> Internet

For example, a communication flow could be: Node F --> Node D --> Node B --> Node A --> Internet

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet go through from source (6LN) to 6LBR.

This case is identical to storing-mode case.

The IPv6 flow label should be set to zero to aid in compression, and the 6LBR will set it to a non-zero value when sending towards the Internet.

Header	6LN	6LR_i	6LBR	Internet
Inserted headers	RPI	--	--	--
Removed headers	--	--	--	--
Re-added headers	--	--	--	--
Modified headers	--	RPI	--	--
Untouched headers	--	--	RPI	RPI (Ignored)

Table 17: Non Storing: Summary of the use of headers from RPL-aware-leaf to Internet

7.2.2. Non-SM: Example of Flow from Internet to RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RPL-aware-leaf (6LN)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 <= i <= n", n is the number of routers (6LR) that the packet go through from 6LBR to destination(6LN).

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.

Header	Internet	6LBR	6LR_i	6LN
Inserted headers	--	IPv6-in-IPv6 (RH3,RPI)	--	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3,RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3,RPI)	--
Untouched headers	--	--	--	--

Table 18: Non Storing: Summary of the use of headers from Internet to RPL-aware-leaf

7.2.3. Non-SM: Example of Flow from not-RPL-aware-leaf to Internet

In this case the flow comprises:

not-RPL-aware-leaf (IPv6) --> 6LR_1 --> 6LR_i -->root (6LBR) --> Internet

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet go through from source(IPv6) to 6LBR. e.g 6LR_1 (i=1).

In this case the flow label is recommended to be zero in the IPv6 node. As RPL headers are added in the IPv6 node packet, the first 6LR (6LR_1) will add a RPI header 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 6.2.3).

Header	IPv6	6LR_1	6LR_i [i=2,..,n]_	6LBR	Internet
Inserted headers	--	IPv6-in-IPv6 (RPI)	--	--	--
Removed headers	--	--	--	IPv6-in-IPv6 (RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6 (RPI)	--	--
Untouched headers	--	--	--	--	--

Table 19: Non Storing: Summary of the use of headers from not-RPL-aware-leaf to Internet

7.2.4. Non-SM: Example of Flow from Internet to not-RPL-aware-leaf

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> not-RPL-aware-leaf (IPv6)

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

6LR_i are the intermediate routers from source to destination. In this case, "1 < i <= n", n is the number of routers (6LR) that the packet go through from 6LBR to not-RPL-aware-leaf (IPv6).

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 an 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.

Header	Internet	6LBR	6LR_1	6LR_i(i=2,.., n)	IPv6
Inserted headers	--	IPv6-in-IPv6 (RH3, RPI)	--	--	--
Removed headers	--	--	--	IPv6-in-IPv6 (RH3,RPI)(1)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IPv6-in-IPv6 (RH3,RPI)	IPv6-in-IPv6 (RH3, RPI)	--
Untouched headers	--	--	--	--	--

Table 20: NonStoring: Summary of the use of headers from Internet to non-RPL-aware-leaf (1) The last 6LR before the IPv6 node.

7.3. Non-Storing Mode: Interaction between Leafs

In this section is described the communication flow in Non Storing Mode (Non-SM) between,

RPL-aware-leaf to RPL-aware-leaf

RPL-aware-leaf to not-RPL-aware-leaf

not-RPL-aware-leaf to RPL-aware-leaf

not-RPL-aware-leaf to not-RPL-aware-leaf

7.3.1. Non-SM: Example of Flow from RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

6LN src --> 6LR_ia --> root (6LBR) --> 6LR_id --> 6LN dst

For example, a communication flow could be: Node F --> Node D --> Node B --> Node A (root) --> Node B --> Node E --> Node H

6LR_ia are the intermediate routers from source to the root In this case, "1 <= ia <= n", n is the number of routers (6LR) that the packet go through from 6LN to the root.

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

This case involves only nodes in same RPL Domain. The originating node will add a RPI header to the original packet, and send the packet upwards.

The originating node SHOULD put the RPI into an IPv6-in-IPv6 header addressed to the root, so that the 6LBR can remove that header. If it does not, then additional resources are wasted on the way down to carry the useless RPI option.

The 6LBR will need to insert an RH3 header, which requires that it add an IPv6-in-IPv6 header. It SHOULD be able to remove the RPI, as it was contained in an IPv6-in-IPv6 header addressed to it. Otherwise, there MAY be a RPI header buried inside the inner IP header, which should get ignored.

Networks that use the RPL P2P extension [RFC6997] are essentially non-storing DODAGs and fall into this scenario or scenario Section 7.1.2, with the originating node acting as 6LBR.

Header	6LN src	6LR_ia	6LBR	6LR_id	6LN dst
Inserted headers	IPv6-in-IPv6 (RPI1)	--	IPv6-in-IPv6 (RH3->6LN, opt RPI2)	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI1)	--	IPv6-in-IPv6 (RH3, opt RPI2)
Re-added headers	--	--	--	--	--
Modified headers	--	RPI1	--	RPI2	--
Untouched headers	--	--	--	--	--

Table 21: Non Storing: Summary of the use of headers for RPL-aware-leaf to RPL-aware-leaf

7.3.2. Non-SM: Example of Flow from RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

6LN --> 6LR_ia --> root (6LBR) --> 6LR_id --> not-RPL-aware (IPv6)

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

6LR_ia are the intermediate routers from source to the root In this case, "1 <= ia <= n", n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

As in the previous case, the 6LN will insert a RPI (RPI_1) 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 6LR_id. The RPI is optional from 6LBR to 6LR_id (RPI_2).

Header	6LN	6LR_1	6LBR	6LR_id	IPv6
Inserted headers	IPv6-in-IPv6 (RPI1)	--	IPv6-in-IPv6 (RH3, opt RPI_2)	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI_1)	IPv6-in-IPv6 (RH3, opt RPI_2)	--
Re-added headers	--	--	--	--	--
Modified headers	--	IPv6-in-IPv6 (RPI_1)	--	IPv6-in-IPv6 (RH3, opt RPI_2)	--
Untouched headers	--	--	--	--	opt RPI_2

Table 22: Non Storing: Summary of the use of headers from RPL-aware-leaf to not-RPL-aware-leaf

7.3.3. Non-SM: Example of Flow from not-RPL-aware-leaf to RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6) --> 6LR_ia --> root (6LBR) --> 6LR_id --> 6LN

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

6LR_ia are the intermediate routers from source to the root. In this case, "1 <= ia <= n", n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

This scenario is mostly identical to the previous one. The RPI 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 it's own IPv6-in-IPv6 header containing an RH3 header and optional RPI (RPI_2).

Header	IPv6	6LR_1	6LBR	6LR_id	6LN
Inserted headers	--	IPv6-in-IPv6 (RPI_1)	IPv6-in-IPv6 (RH3, opt RPI_2)	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI_1)	--	IPv6-in-IPv6 (RH3, opt RPI_2)
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	IPv6-in-IPv6 (RH3, opt RPI_2)	--
Untouched headers	--	--	--	--	--

Table 23: Non Storing: Summary of the use of headers from not-RPL-aware-leaf to RPL-aware-leaf

7.3.4. Non-SM: Example of Flow from not-RPL-aware-leaf to not-RPL-aware-leaf

In this case the flow comprises:

not-RPL-aware 6LN (IPv6 src)--> 6LR_ia --> root (6LBR) --> 6LR_id --> not-RPL-aware (IPv6 dst)

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

6LR_ia are the intermediate routers from source to the root. In this case, "1 <= ia <= n", n is the number of intermediate routers (6LR)

6LR_id are the intermediate routers from the root to the destination. In this case, "1 <= ia <= m", m is the number of the intermediate routers (6LR).

This scenario is the combination of the previous two cases.

Header	IPv6 src	6LR_1	6LBR	6LR_id	IPv6 dst
Inserted headers	--	IPv6-in-IPv6 (RPI_1)	IPv6-in-IPv6 (RH3, opt RPI_2)	--	--
Removed headers	--	--	IPv6-in-IPv6 (RPI_1)	IPv6-in-IPv6 (RH3, opt RPI_2)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	--	--
Untouched headers	--	--	--	--	--

Table 24: Non Storing: Summary of the use of headers from not-RPL-aware-leaf to not-RPL-aware-leaf

8. Operational Considerations of supporting not-RPL-aware-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 hosts that are pre-RFC8200, or simply intolerant. Those hosts will drop packets that continue to have RPL artifacts in them. In general, such hosts 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 be create the RH3 header), and therefore knows what the 6LR prior to the leaf --- the 6LR_n.

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 either 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 lot 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.

9. Operational considerations of introducing 0x23

This section describes the operational considerations

9.1. Has deployment been discussed?

There are no known multivendor deployments outside of the research groups! All known deployments of RPL are in market verticals, with a single vendor providing all components. Research groups typically are using Contiki, RiotOS, or OpenWSN., and these are easily adapted to 0x23 functionality. Compatibility issue of RPL Option type not seems to be quite relevant for research groups.

9.2. Has installation and initial setup been discussed??

During bootstrapping the node get the DIO with the information of RPL Option Type and 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 gradually, through DIO messages and when all the nodes are set with the new value. The DODAG should change to a new DODAG version. Not able to send data plane messages. Should drop the packet. In case of rebooting, the node does not remember the RPL Option Type. Thus, the DIO is sent with flag indicating the new RPI value.

9.3. Has the migration path been discussed?

TBD

9.4. Have the Requirements on other protocols and functional components been discussed?

It allows to send packets to non-RPL nodes, the 0x23?..

TBD

9.5. Has the impact on network operation been discussed?

Missconfiguration in case that a node join.

TBD

9.6. Have suggestions for verifying correct operation been discussed?

TBD

9.7. Has management interoperability been discussed?

TBD

9.8. Are there fault or threshold conditions that should be reported?

TBD

9.9. Is configuration discussed?

TBD

10. IANA Considerations

This document updates the registration made in [RFC6553] Destination Options and Hop-by-Hop Options registry from 0x63 to 0x23.

[RFCXXXX] represents this document.

Hex Value	Binary Value act	chg	rest	Description	Reference
0x23	00	1	00011	RPL Option	[RFCXXXX]
0x63	01	1	00011	RPL Option(DEPRECATED)	[RFC6553][RFCXXXX]

Figure 9: Option Type in RPL Option.

The DODAG Configuration Option Flags in the DODAG Configuration option is updated as follows:

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 10: DODAG Configuration Option Flag to indicate the RPI-flag-day.

11. 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, they could still have a significant impact, particularly if the attack was on 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 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-secure-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 will 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.

[RFC2473] suggests that tunnel entry and exit points can be secured, via the "Use IPsec". The suggested solution has all the problems that [RFC5406] goes into. In an LLN such a solution would degenerate into every node having a tunnel with every other node. It would provide a small amount of origin address authentication at a very high cost; doing BCP38 at every node (linking layer-3 addresses to layer-2 addresses, and to already present layer-2 cryptographic mechanisms) would be cheaper should RPL be run in an environment where hostile nodes are likely to be a part of the LLN.

The RH3 header usage described here can be abused in equivalent ways with an IPv6-in-IPv6 header to add the needed RH3 header. As such, the attacker's RH3 header will not be seen by the network until it reaches the end host, which will decapsulate it. An end-host SHOULD be suspicious about a RH3 header which has additional hops which have not yet been processed, and SHOULD ignore such a second RH3 header.

In addition, the LLN will likely use [RFC8138] to compress the IPv6-in-IPv6 and RH3 headers. As such, the compressor at the RPL-root will see the second RH3 header and MAY choose to discard the packet if the RH3 header has not been completely consumed. 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. It should just be noted that an incoming RH3 must be fully consumed, or very carefully inspected.

The RPI header, 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 instanceID, but a change of instanceID would permit an attacker to bypass such filtering. Like the RH3, a RPI header 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 header therefore will not be seen by the network. Upon reaching the destination node the RPI header has no further meaning and is just skipped; the presence of a second RPI header will have no meaning to the end node as the packet has already been identified as being at it's final destination.

The RH3 and RPI headers could be abused by an attacker inside of the network to route packets on non-obvious ways, perhaps eluding observation. This usage is in fact part 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.

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 do a deep packet inspection wherein it walks 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 ([RFC2827]) 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.

12. Acknowledgments

We are very thankful to the grant by the Finnish Foundation for Technology Promotion (Tekniikan Edistaemissaeaetioen - TES), and the grant by the FP7 Marie Curie Initial Training Network (ITN) METRICS project (grant agreement No. 607728)

A special BIG thanks to C. M. Heard for the help with the Section 3. 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): Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, Rahul Jadhav, Matthias Kovatsch, Peter van der Stok, Xavier Vilajosana and Thomas Watteyne.

13. References

13.1. Normative References

[I-D.thubert-roll-unaware-leaves]
Thubert, P., "Routing for RPL Leaves", draft-thubert-roll-unaware-leaves-06 (work in progress), November 2018.

[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>.

[RFC2827] 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/rfc2827>.

[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>.

[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>.

13.2. 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-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-10 (work in progress), January 2019.

[I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.

[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-18 (work in progress), August 2018.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-18 (work in progress), January 2019.

[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>.

[RFC4192] Baker, F., Lear, E., and R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", RFC 4192, DOI 10.17487/RFC4192, September 2005, <https://www.rfc-editor.org/info/rfc4192>.

[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>.

[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>.

Authors' Addresses

Maria Ines Robles
Aalto University
Innopoli
Espoo 02150
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
Village d'Entreprises Green Side 400, Avenue de Roumanille
Batiment T3, Biot - Sophia Antipolis 06410
France

Email: pthubert@cisco.com

ROLL R. Koutsiamanis, Ed.
Internet-Draft G. Papadopoulos
Intended status: Standards Track N. Montavont
Expires: April 29, 2018 IMT Atlantique
P. Thubert
Cisco
October 26, 2017

RPL DAG Metric Container (MC) Node State and Attribute (NSA) object type extension
draft-pkm-roll-nsa-extension-00

Abstract

Implementing 6TiSCH Packet Replication and Elimination from / to the RPL root requires the ability to forward copies of packets over different paths via different RPL parents. Selecting the appropriate parents to achieve ultra-low latency and jitter requires information about a node’s parents. This document details what information needs to be transmitted and how it is encoded within a packet to enable this functionality.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Industrial network applications have stringent requirements on reliability and predictability, and typically leverage 1+1 redundancy, aka Packet Replication and Elimination (PRE) [I-D.papadopoulos-6tisch-pre-reqs] to achieve their goal. In order for wireless networks to be able to be used in such applications, the principles of Deterministic Networking [I-D.ietf-detnet-architecture] lead to designs that aim at maximizing packet delivery rate and minimizing latency and jitter. Additionally, given that the network nodes often do not have an unlimited power supply, energy consumption needs to be minimized as well.

To meet this goal, IEEE Std. 802.15.4 [IEEE802154-2015] provides Time-Slotted Channel Hopping (TSCH), a mode of operation which uses a fixed communication schedule to allow deterministic medium access as well as channel hopping to work around radio interference. However, since TSCH uses retransmissions in the event of a failed transmission, end-to-end delay and jitter performance can deteriorate.

The 6TiSCH working group, focusing on IPv6 over IEEE Std. 802.15.4-TSCH, has worked on the issues previously highlighted and produced the "6TiSCH Architecture" [I-D.ietf-6tisch-architecture] to address that case. Building on this architecture, "Exploiting Packet Replication and Elimination in Complex Tracks in 6TiSCH LLNs" [I-D.papadopoulos-6tisch-pre-reqs] leverages PRE to improve the Packet Delivery Ratio (PDR), provide a hard bound to the end-to-end latency, and limit jitter.

PRE achieves a controlled redundancy by laying multiple forwarding paths through the network and using them in parallel for different copies of a same packet. PRE can follow the Destination-Oriented Directed Acyclic Graph (DODAG) formed by RPL from a node to the root. Building a multi-path DODAG can be achieved based on the RPL capability of having multiple parents for each node in a network, a subset of which is used to forward packets. In order for this subset to be defined, a RPL parent subset selection mechanism, which falls within the remit of the RPL Objective Function (OF), needs to have specific path information. The specification of the transmission of this information is the focus of this document.

More concretely, this specification focuses on the extensions to the DAG Metric Container [RFC6551] required for providing the PRE mechanism a part of the information it needs to operate. This information is the RPL [RFC6550] parent node address set of a node and it must be sent to potential children nodes of the node. The RPL DIO Control Message is the canonical way of broadcasting this kind of information and therefore its DAG Metric Container [RFC6551] field is used to append a Node State and Attribute (NSA) object. The node's parent node address set is stored as an optional TLV within the NSA object. This specification defines the type value and structure for this TLV.

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].

3. Tracks

3.1. Tracks Overview

The concept of Track is introduced in the "6TiSCH Architecture" [I-D.ietf-6tisch-architecture], defined as a sequence of elements, each consisting of the 3-tuple of a transmitter, a receiver, and a given timeslot expressed as a slotOffset/channelOffset tuple. A simple Track is intended to provide the full resources required to allow the transmission of a single packet from a source 6TiSCH node to a destination 6TiSCH node across a 6TiSCH multihop path.

3.2. Complex Tracks

Similarly to, but as a generalization of a simple Track, a Complex Track is defined in the "6TiSCH Architecture" [I-D.ietf-6tisch-architecture] as a DODAG starting at a source 6TiSCH node and leading to a sink 6TiSCH node in order to support multi-path forwarding. Multiple independent paths may be produced by using techniques for Packet Replication and Elimination (PRE) [I-D.papadopoulos-6tisch-pre-reqs] based on DetNet [I-D.ietf-detnet-architecture] principles. As an example, a complex Track allows for branching off and rejoining over non-congruent paths.

In the following Section, we will detail Deterministic Networks PRE techniques.

4. Packet Replication and Elimination principles

The idea behind Packet Replication and Elimination (PRE) is to transmit the same data packet through parallel and adjacent paths in a network with the aim of improving reliability and predictability through redundancy.

The process of replication consists of identifying multiple potential paths, selecting a subset to use, and sending copies of a single packet through each path. When receiving packets the process of elimination is required so that multiple copies of the same packet are not replicated again, to avoid an exponential growth in unnecessary traffic. Combined together, these processes enable controlled redundancy which in turn can be used to achieve the previously stated goals of reliability (i.e., ultra-high packet delivery rate) and predictability (i.e., ultra-low end-to-end delay and jitter) in wireless networks. For example, in Figure 1, the source 6TiSCH node S is sending the data packet to its what is called RPL Default Parent (DP) and Alternative Parent (AP), nodes A and B, in two different timeslots.

```
                ===> (A) ====> (C) ===
              //         \\  //       \\
    source (S)             \\           (R) (root)
              \\         //  \\       //
                ===> (B) ====> (D) ===
```

Figure 1: Packet Replication: S transmits twice the same data packet, to its DP (A) and to its AP (B).

In Exploiting Packet Replication and Elimination in Complex Tracks in 6TiSCH LLNs [I-D.papadopoulos-6tisch-pre-reqs], the concept of PRE is further expanded along with its requirements.

5. Alternative Parent Selection Issue

In the RPL protocol, each node maintains a list of potential parents. For PRE, the DP node is defined as the RPL DODAG preferred parent node. Furthermore, to construct an alternative path toward the root, in addition to the DP node, each 6TiSCH node in the network registers an AP node as well. There are multiple alternative methods of selecting the AP node, functionality which is included in operation of the RPL Objective Function (OF). In Exploiting Packet Replication and Elimination in Complex Tracks in 6TiSCH LLNs [I-D.papadopoulos-6tisch-pre-reqs], a scheme which allows the two paths to remain correlated is detailed. More specifically, in this scheme a 6TiSCH node will select an alternative parent node close to its default parent node to allow the operation of overhearing between parents. To do so, the node will check if its Default Grand Parent (DGP), the DP of its DP, is in the set of parents of a potential AP. If multiple potential APs match this condition, the AP with the lowest rank will be registered.

For instance, in Figure 1, source 6TiSCH node S must know its grandparent sets both through node A and through node B. In this scenario, node A and node B have the same parent sets, nodes C and D, and therefore for node S, the grandparent set through A is the same as the grandparent set through B.

In order to select their AP node, 6TiSCH nodes need to be aware of their grandparent node sets. Within RPL [RFC6550], the nodes use the DODAG Information Object (DIO) Control Message to broadcast information about themselves to potential children. However, RPL [RFC6550], does not define how to propagate parent set related information, which is what this document addresses.

6. Node State and Attribute (NSA) object type extension

For supporting PRE, nodes need to report their parent node set to their potential children. DIO messages can carry multiple options, out of which the DAG Metric Container option [RFC6551] is the most suitable structurally and semantically for the purpose of carrying the parent node set. The DAG Metric Container option itself can carry different nested objects, out of which the Node State and Attribute (NSA) [RFC6551] is appropriate for transferring generic node state data. Within the Node State and Attribute it is possible to store optional TLVs representing various node characteristics. As per the Node State and Attribute (NSA) [RFC6551] description, no TLV have been defined for use. This document defines one TLV for the purpose of transmitting a node's parent node set.

```
 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 |Version Number |             Rank              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G|0| MOP | Prf |     DTSN      |     Flags     |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                            DODAGID                            +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| DAGMC Type (2)| DAGMC Length  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
//                  DAG Metric Container data                  //
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Example DIO Message with a DAG Metric Container option

The structure of the DIO Control Message when a DAG Metric Container option is included is shown in Figure 2. The DAG Metric Container option type (DAGMC Type in Figure 2) has the value 0x02 as per the IANA registry for the RPL Control Message Options, and is defined in [RFC6550]. The DAG Metric Container option length (DAGMC Length in Figure 2) expresses the the DAG Metric Container length in bytes. DAG Metric Container data holds the actual data and is shown further expanded in Figure 3.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Routing-MC-Type|Res Flags|P|C|O|R| A   |  Prec | Length (bytes)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Res       |  Flags    |A|O| PNS type (1)  | PNS Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PNS IPv6 address(es) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: DAG Metric Container data with Node State and Attribute (NSA) object body and a TLV

The structure of the DAG Metric Container data in the form of a Node State and Attribute (NSA) object with a TLV in the NSA Optional TLVs field is shown in Figure 3. The DAG Metric Container fields up to the first 48 bits (including the O flag) are defined in [RFC6551] as part of the Node State and Attribute (NSA) object body. This document defines a new TLV, which CAN be carried in the Node State and Attribute (NSA) object Optional TLVs field. The TLV is named Parent Node Set and is abbreviated as PNS in Figure 3.

PNS type: The type of the Parent Node Set TLV. The value is 1.

PNS Length: The total length of the TLV value field (PNS IPv6 address(es)) in bytes.

PNS IPv6 address(es): A sequence of zero or more IPv6 addresses belonging to a node’s parent set. Each address requires 16 bytes. The order of the parents in the parent set is in decreasing preference based on the Objective Function [RFC6550] used by the node.

6.1. Compression

The PNS IPv6 address(es) field in the Parent Node Set TLV MAY be compressed using any compression method available to conserve space.

7. Security Considerations

TODO.

8. IANA Considerations

TBA.

9. References

9.1. 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-12 (work in progress), August 2017.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-03 (work in progress), August 2017.

[I-D.papadopoulos-6tisch-pre-reqs]
Papadopoulos, G., Montavont, N., and P. Thubert, "Exploiting Packet Replication and Elimination in Complex Tracks in 6TiSCH LLNs", draft-papadopoulos-6tisch-pre-reqs-00 (work in progress), July 2017.

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

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

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

9.2. Other Informative References

[IEEE802154-2015]
IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.

Authors' Addresses

Remous-Aris Koutsiamanis (editor)
IMT Atlantique
Office B00 - 126A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 49
Email: aris@ariskou.com

Georgios Papadopoulos
IMT Atlantique
Office B00 - 114A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 04
Email: georgios.papadopoulos@imt-atlantique.fr

Nicolas Montavont
IMT Atlantique
Office B00 - 106A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 23
Email: nicolas.montavont@imt-atlantique.fr

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

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

roll
Internet-Draft M.Qasem
Intended Status: Standards Track A.Al-Dubai
Expires: May 2, 2018 I.Romdhani
B.Ghaleb
Edinburgh Napier University
J. Hou
R. Jadhav
Huawei Technologies
October 29, 2017

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

Copyright and License Notice

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 nodes	Child nodes	Shared 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