ROLL
Internet-Draft
Intended status: Standards Track
Expires: March 13, 2018

S. Anamalamudi
Huaiyin Institute of Technology
M. Zhang
AR. Sangi
Huawei Technologies
C. Perkins
Futurewei
S.V.R.Anand
Indian Institute of Science

September 9, 2017

Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs) draft-ietf-roll-aodv-rpl-02

Abstract

Route discovery for symmetric and asymmetric Point-to-Point (P2P) traffic flows is a desirable feature in Low power and Lossy Networks (LLNs). For that purpose, this document specifies a reactive P2P route discovery mechanism for hop-by-hop routing (storing mode) based on Ad Hoc On-demand Distance Vector Routing (AODV) based RPL protocol. Two separate Instances are used to construct directional paths in case some of the links between source and target node are asymmetric.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 13, 2018.

Copyright Notice

Table of Contents

1. Introduction

RPL[RFC6550], the IPv6 distance vector routing protocol for Low-power and Lossy Networks (LLNs), is designed to support multiple traffic flows through a root-based Destination-Oriented Directed Acyclic Graph (DODAG). For traffic flows between routers within the DODAG (i.e., Point-to-Point (P2P) traffic), this means that data packets either have to traverse the root in non-storing mode (source routing), or traverse a common ancestor in storing mode (hop-by-hop routing). Such P2P traffic is thereby likely to flow along sub-optimal routes and may suffer severe traffic congestion near the DAG root [RFC6997], [RFC6998].

To discover optimal paths for P2P traffic flows in RPL, P2P-RPL [RFC6997] specifies a temporary DODAG where the source acts as temporary root. The source initiates "P2P Route Discovery mode (P2P-RDO)" with an address vector for both non-storing mode (H=0) and

storing mode (H=1). Subsequently, each intermediate router adds its IP address and multicasts the P2P-RDO message, until the message reaches the target node (TargNode). TargNode sends the "Discovery Reply" option. P2P-RPL is efficient for source routing, but much less efficient for hop-by-hop routing due to the extra address vector overhead. In fact, when the P2P-RDO message is being multicast from the source hop-by-hop, receiving nodes are able to determine a next hop towards the source in symmetric links. When TargNode subsequently replies to the source along the established forward route, receiving nodes can determine the next hop towards TargNode. In other words, it is efficient to use only routing tables for P2P-RDO message instead of "Address vector" for hop-by-hop routes (H=1) in symmetric links.

RPL and P2P-RPL both specify the use of a single DODAG in networks of symmetric links. But, application-specific routing requirements that are defined in IETF ROLL Working Group [RFC5548], [RFC5673], [RFC5826] and [RFC5867] may need routing metrics and constraints enabling use of asymmetric bidirectional links. For this purpose, [I-D.thubert-roll-asymlink] describes bidirectional asymmetric links for RPL [RFC6550] with Paired DODAGs, for which the DAG root (DODAGID) is common for two Instances. This can satisfy application-specific routing requirements for bidirectional asymmetric links in base RPL [RFC6550]. P2P-RPL for Paired DODAGs, on the other hand, requires two DAG roots: one for the source and another for the target node due to temporary DODAG formation. For networks composed of bidirectional asymmetric links (see Section 4), AODV-RPL specifies P2P route discovery, utilizing RPL with a new MoP. AODV-RPL makes use of two multicast messages to discover possibly asymmetric routes. AODV-RPL eliminates the need for address vector control overhead, significantly reducing the control packet size which is important for Constrained LLN networks. Both discovered routes meet the application specific metrics and constraints that are defined in the Objective Function for each Instance [RFC6552].

The route discovery process in AODV-RPL is modeled on the analogous process that has been specified in AODV [RFC6550]. The on-demand nature of AODV route discovery is natural for the needs of peer-to-peer routing as envisioned for RPL-based LLNs. Similar terminology has been adopted for use with the discovery messages, namely RREQ for Route Request, and RREP for Route Reply. AODV-RPL is, at heart, a simpler protocol than AODV, since there are no analogous operations for flagging Route Errors, blacklisting unidirectional links, multihoming, or handling unnumbered interfaces. Some of the simpler features of AODV, on the other hand, have been imported into AODV-RPL -- for instance, prefix advertisement is allowed on RREP and RREQ message where appropriate.

2. Terminology

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

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

AODV-Instance
: Either the RREQ-Instance or RREP-Instance

Bi-directional Asymmetric Link
: A link that can be used in both directions but with different link characteristics (see [I-D.thubert-roll-asymlink]).

DODAG RREQ-Instance (or simply RREQ-Instance)
: AODV Instance built using the RREQ option; used for control transmission from OrigNode to TargNode, thus enabling data transmission from TargNode to OrigNode.

DODAG RREP-Instance (or simply RREP-Instance)
: AODV Instance built using the RREP option; used for control transmission from TargNode to OrigNode thus enabling data transmission from OrigNode to TargNode.

downstream
: Routing along the direction from OrigNode to TargNode.

hop-by-hop routing
: Routing when each node stores routing information about the next hop.

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

Paired DODAGs
: Two DODAGs for a single application.

P2P
: Point-to-Point -- in other words, not constrained to traverse a common ancestor.

RREQ message

An AODV-RPL MoP DIO message containing the RREQ option. The InstanceID in the DIO object of the RREQ option MUST be always an odd number.

RREP message
An AODV-RPL MoP DIO message containing the RREP option. The InstanceID in the DIO object of the RREP option MUST be always an even number (usually, InstanceID of RREQ-Instance+1).

source routing
The mechanism by which the source supplies the complete route towards the target node along with each data packet. [RFC6997].

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

upstream
Routing along the direction from TargNode to OrigNode.

3. Overview of AODV-RPL

With AODV-RPL, routes from OrigNode to TargNode within the LLN network established are "on-demand". In other words, the route discovery mechanism in AODV-RPL is invoked reactively when OrigNode has data for delivery to the TargNode but existing routes do not satisfy the application’s requirements. The routes discovered by AODV-RPL are point-to-point; in other words the routes are not constrained to traverse a common ancestor. Unlike base RPL [RFC6550] and P2P-RPL [RFC6997], AODV-RPL can enable asymmetric communication paths in networks with bidirectional asymmetric links. For this purpose, AODV-RPL enables discovery of two routes: namely, one from OrigNode to TargNode, and another from TargNode to OrigNode. When possible, AODV-RPL also enables symmetric routing along Paired DODAGs (see Section 4).

4. AODV-RPL Mode of Operation (MoP)

In AODV-RPL, route discovery is initiated by forming a temporary DAG rooted at the OrigNode. Paired DODAGs (Instances) are constructed according to a new AODV-RPL Mode of Operation (MoP) during route formation between the OrigNode and TargNode. The RREQ-Instance is formed by route control messages from OrigNode to TargNode whereas the RREP-Instance is formed by route control messages from TargNode to OrigNode (as shown in Figure 2). Intermediate routers join the Paired DODAGs based on the rank as calculated from the DIO message. Henceforth in this document, the RREQ-Instance message means the AODV-RPL DIO message from OrigNode to TargNode, containing the RREQ

option. Similarly, the RREP-Instance message means the AODV-RPL DIO message from TargNode to OrigNode, containing the RREP option. Subsequently, the RREQ-Instance is used for data transmission from TargNode to OrigNode and RREP-Instance is used for Data transmission from OrigNode to TargNode.

The AODV-RPL Mode of Operation defines a new bit, the Symmetric bit ('S'), which is added to the base DIO message as illustrated in Figure 1. OrigNode sets the the 'S' bit to 1 in the RREQ-Instance message when initiating route discovery.

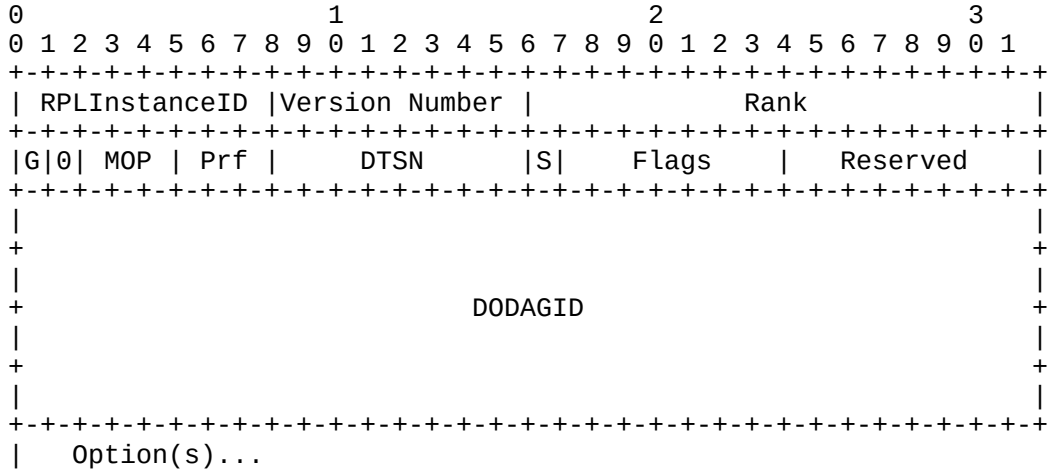

Figure 1: DIO modification to support asymmetric route discovery

A device originating a AODV-RPL message supplies the following information in the DIO header of the message:

'S' bit

Symmetric bit in the DIO base object

MOP

MOP operation in the DIO object MUST be set to "5(TBD1)" for AODV-RPL DIO messages

RPLInstanceID

RPLInstanceID in the DIO object MUST be the InstanceID of AODV-Instance(RREQ-Instance). The InstanceID for RREQ-Instance MUST be always an odd number.

DODAGID

For RREQ-Instance :

DODAGID in the DIO object MUST be the IPv6 address of the device that initiates the RREQ-Instance.

For RREP-Instance

DODAGID in the DIO object MUST be the IPv6 address of the device that initiates the RREP-Instance.

Rank

Rank in the DIO object MUST be the the rank of the AODV-Instance (RREQ-Instance).

Metric Container Options

AODV-Instance(RREQ-Instance) messages MAY carry one or more Metric Container options to indicate the relevant routing metrics.

The 'S' bit is set to mean that the route is symmetric. If the RREQ-Instance arrives over an interface that is known to be symmetric, and the 'S' bit is set to 1, then it remains set at 1, as illustrated in Figure 2. In Figure 2 and Figure 3, BR is the BorderRouter, S is the OrigNode, R is an intermediate node, and D is the TargNode.

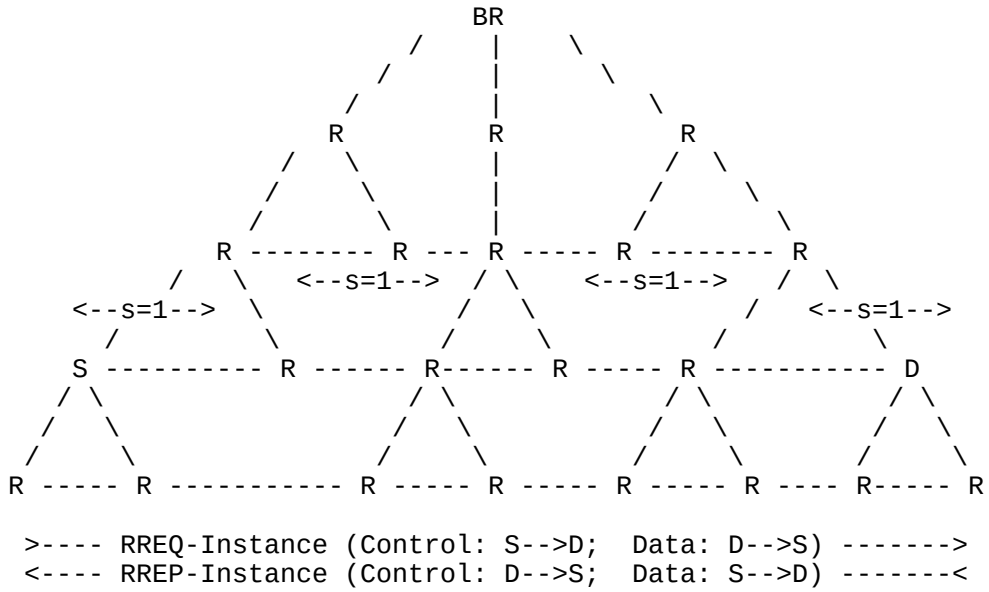

Figure 2: AODV-RPL with Symmetric Paired Instances

If the RREQ-Instance arrives over an interface that is not known to be symmetric, or is known to be asymmetric, the 'S' bit is set to be 0. Moreover, if the 'S' bit arrives already set to be '0', it is set to be '0' on retransmission (Figure 3). Based on the 'S' bit received in RREQ-Instance, the TargNode decides whether or not the route is symmetric before transmitting the RREP-Instance message upstream towards the OrigNode. The metric used to determine symmetry (i.e., set the "S" bit to be "1" (Symmetric) or "0" (asymmetric)) is implementation specific. We used ETX/RSSI to verify the feasibility of the protocol operations in this draft, as discussed in Appendix A.

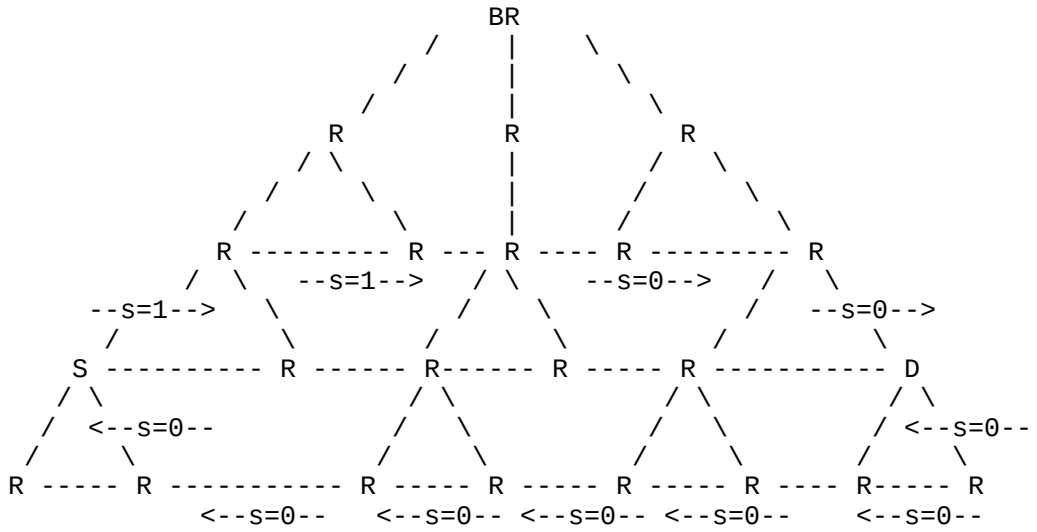

Figure 3: AODV-RPL with Asymmetric Paired Instances

5. RREQ Message

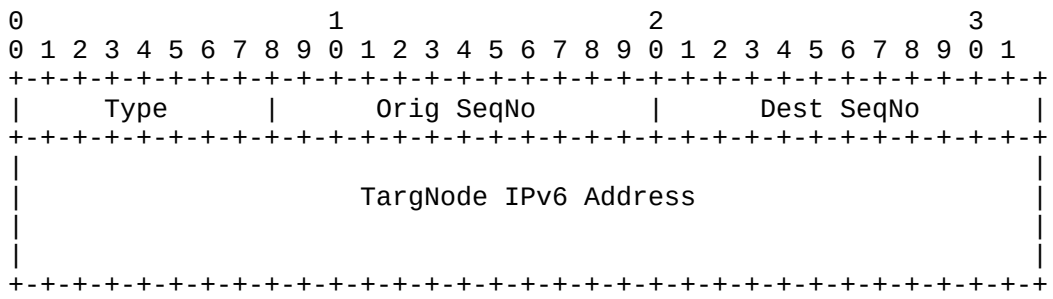

Figure 4: DIO RREQ option format for AODV-RPL MoP

OrigNode supplies the following information in the RREQ option of the RREQ-Instance message:

Type

The type of the RREQ option(see Section 9.2).

Orig SeqNo

Sequence Number of OrigNode.

Dest SeqNo

If nonzero, the last known Sequence Number for TargNode for which a route is desired.

TargNode IPv6 Address

IPv6 address of the TargNode that receives RREQ-Instance message.

In order to establish the upstream route from TargNode to OrigNode, OrigNode multicasts the RREQ-Instance message (see Figure 4) to its one-hop neighbours. In order to enable intermediate nodes R_i to associate a future RREP message to an incoming RREQ message, the InstanceID of RREQ-Instance MUST assign an odd number.

Each intermediate node R_i computes the rank for RREQ-Instance and creates a routing table entry for the upstream route towards the source if the routing metrics/constraints are satisfied. For this purpose R_i must use the asymmetric link metric measured in the upstream direction, from R_i to its upstream neighbor that multicasted the RREQ-Instance message.

When an intermediate node R_i receives a RREQ message in storing mode, it MUST store the OrigNode's InstanceID (RREQ-Instance) along with the other routing information needed to establish the route back to the OrigNode. This will enable R_i to determine that a future RREP message (containing a paired InstanceID for the TargNode) must be transmitted back to the OrigNode's IP address.

If the paths to and from TargNode are not known, the intermediate node multicasts the RREQ-Instance message with updated rank to its next-hop neighbors until the message reaches TargNode (Figure 2). Based on the 'S' bit in the received RREQ message, the TargNode will decide whether to unicast or multicast the RREP message back to OrigNode.

As described in Section 7, in certain circumstances R_i MAY unicast a Gratuitous RREP towards OrigNode, thereby helping to minimize multicast overhead during the Route Discovery process.

6. RREP Message

The TargNode supplies the following information in the RREP message:

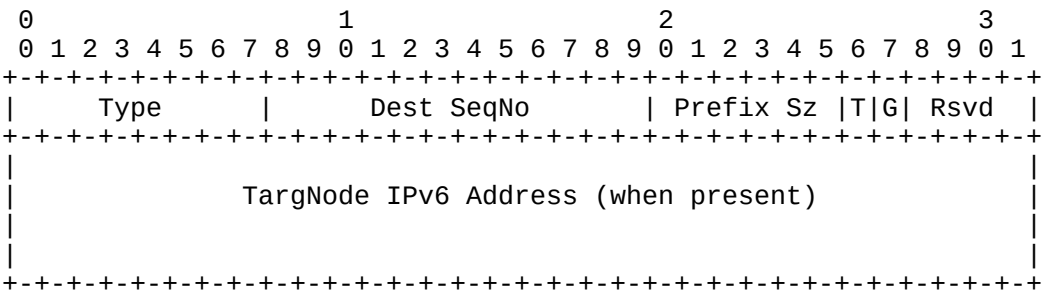

Figure 5: DIO RREP option format for AODV-RPL MoP

Type

The type of the RREP option (see Section 9.2)

Dest SeqNo

The Sequence Number for the TargNode for which a route is established.

Prefix Sz

The size of the prefix which the route to the TargNode is available. This allows routing to other nodes on the same subnet as the TargNode.

'T' bit

'T' is set to true to indicate that the TargNode IPv6 Address field is present

'G' bit

(see Section 7)

TargNode IPv6 Address (when present)

IPv6 address of the TargNode that receives RREP-Instance message.

In order to reduce the need for the TargNode IPv6 Address to be included with the RREP message, the InstanceID of the RREP-Instance is paired, whenever possible, with the InstanceID from the RREQ message, which is always an odd number. The pairing is accomplished by adding one to the InstanceID from the RREQ message and using that, whenever possible, as the InstanceID for the RREP message. If this is not possible (for instance because the incremented InstanceID is

still a valid InstanceID for another route to the TargNode from an earlier Route Discovery operation), then the 'T' bit is set and an alternative even number is chosen for the InstanceID of RREP from TargNode.

The OrigNode IP address for RREQ-Instance is available as the DODAGID in the DIO base message (see Figure 1). When TargNode receives a RREQ message with the 'S' bit set to 1 (as illustrated in Figure 2), it unicasts the RREP message with the 'S' bit set to 1. In this case, route control messages and application data between OrigNode and TargNode for both RREQ-Instance and RREP-Instance are transmitted along symmetric links. When the 'T' bit is set to "1" in the RREP-Instance, then the TargNode IPv6 Address is transmitted in the RREP option. Otherwise, the TargNode IPv6 Address is elided in the RREP option.

When (as illustrated in Figure 3) the TargNode receives RREQ message with the 'S' bit set to 0, it also multicasts the RREP message with the 'S' bit set to 0. Intermediate nodes create a routing table entry for the path towards the TargNode while processing the RREP message to OrigNode. Once OrigNode receives the RREP message, it starts transmitting the application data to TargNode along the path as discovered through RREP messages. On the other hand, application data from TargNode to OrigNode is transmitted through the path that is discovered from RREQ message.

7. Gratuitous RREP

Under some circumstances, an Intermediate Node that receives a RREQ message MAY transmit a "Gratuitous" RREP message back to OrigNode instead of continuing to multicast the RREQ message towards TargNode. For these circumstances, the 'G' bit of the RREP option is provided to distinguish the Gratuitous RREP sent by the Intermediate node from the RREP sent by TargNode.

When an Intermediate node R receives a RREQ message and has recent information about the cost of an upstream route from TargNode to R, then R MAY unicast the Gratuitous RREP (GRREP) message to OrigNode. R determines whether its information is sufficiently recent by comparing the value it has stored for the Sequence Number of TargNode against the DestSeqno in the incoming RREQ message. R also must have information about the metric information of the upstream route from TargNode. The GRREP message MUST have PrefixSz == 0 and the 'G' bit set to 1. R SHOULD also unicast the RREQ message to TargNode, to make sure that TargNode will have a route to OrigNode.

8. Operation of Trickle Timer

The trickle timer operation to control RREQ-Instance/RREP-Instance multicast is similar to that in P2P-RPL [RFC6997].

9. IANA Considerations

9.1. New Mode of Operation: AODV-RPL

IANA is required to assign a new Mode of Operation, named "AODV-RPL" for Point-to-Point(P2P) hop-by-hop routing under the RPL registry. The value of TBD1 is assigned from the "Mode of Operation" space [RFC6550].

Value	Description	Reference
TBD1 (5)	AODV-RPL	This document

Figure 6: Mode of Operation

9.2. AODV-RPL Options: RREQ and RREP

Two entries are required for new AODV-RPL options "RREQ-Instance" and "RREQ-Instance", with values of TBD2 (0x0A) and TBD3 (0x0B) from the "RPL Control Message Options" space [RFC6550].

Value	Meaning	Reference
TBD2 (0x0A)	RREQ Option	This document
TBD3 (0x0B)	RREP Option	This document

Figure 7: AODV-RPL Options

10. Security Considerations

This document does not introduce additional security issues compared to base RPL. For general RPL security considerations, see [RFC6550].

11. Future Work

It may become feasible in the future to design a non-storing version of AODV-RPL’s route discovery protocol. Under the current assumption of route asymmetry across bidirectional links, the specification is

expected to be straightforward. It should be possible to re-use the same methods of incremental construction for source routes within analogous fields within AODV-RPL's RREQ and RREP messages as is currently done for DAO messages -- in other words the RPL messages for DODAG construction.

There has been some discussion about how to determine the initial state of a link after an AODV-RPL-based network has begun operation. The current draft operates as if the links are symmetric until additional metric information is collected. The means for making link metric information is considered out of scope for AODV-RPL. In the future, RREQ and RREP messages could be equipped with new fields for use in verifying link metrics. In particular, it is possible to identify unidirectional links; an RREQ received across a unidirectional link has to be dropped, since the destination node cannot make use of the received DODAG to route packets back to the source node that originated the route discovery operation. This is roughly the same as considering a unidirectional link to present an infinite cost metric that automatically disqualifies it for use in the reverse direction.

12. References

12.1. Normative References

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

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

[RFC5548] Dohler, M., Ed., Watteyne, T., Ed., Winter, T., Ed., and D. Barthel, Ed., "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, DOI 10.17487/RFC5548, May 2009, <https://www.rfc-editor.org/info/rfc5548>.

[RFC5673] Pister, K., Ed., Thubert, P., Ed., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, DOI 10.17487/RFC5673, October 2009, <https://www.rfc-editor.org/info/rfc5673>.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, DOI 10.17487/RFC5826, April 2010, <https://www.rfc-editor.org/info/rfc5826>.

[RFC5867] Martocci, J., Ed., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, DOI 10.17487/RFC5867, June 2010, <https://www.rfc-editor.org/info/rfc5867>.

[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>.

[RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <https://www.rfc-editor.org/info/rfc6552>.

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

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

12.2. Informative References

[I-D.thubert-roll-asymlink] Thubert, P., "RPL adaptation for asymmetrical links", draft-thubert-roll-asymlink-02 (work in progress), December 2011.

Appendix A. ETX/RSSI Values to select S bit

We have tested the combination of "RSSI(downstream)" and "ETX (upstream)" to decide whether the link is symmetric or asymmetric at the intermediate nodes. The example of how the ETX and RSSI values are used in conjuction is explained below:

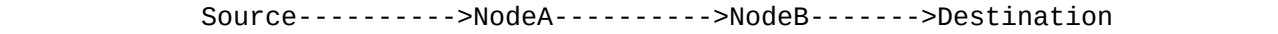

Figure 8: Communication link from Source to Destination

RSSI at NodeA for NodeB	Expected ETX at NodeA for nodeB->nodeA
> -15	150
-25 to -15	192
-35 to -25	226
-45 to -35	662
-55 to -45	993

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

We tested the operations in this specification by making the following experiment, using the above parameters. In our experiment, a communication link is considered as symmetric if the ETX value of NodeA->NodeB and NodeB->NodeA (See Figure.8) are, say, within 1:3 ratio. This ratio should be taken as a notional metric for deciding link symmetric/asymmetric nature, and precise definition of the ratio is beyond the scope of the draft. In general, NodeA can only know the ETX value in the direction of NodeA -> NodeB but it has no direct way of knowing the value of ETX from NodeB->NodeA. Using physical testbed experiments and realistic wireless channel propagation models, one can determine a relationship between RSSI and ETX representable as an expression or a mapping table. Such a relationship in turn can be used to estimate ETX value at nodeA for link NodeB--->NodeA from the received RSSI from NodeB. Whenever nodeA determines that the link towards the nodeB is bi-directional asymmetric then the "S" bit is set to "S=0". Later on, the link from NodeA to Destination is asymmetric with "S" bit remains to "0".

Authors’ Addresses

Satish Anamalamudi
Huaiyin Institute of Technology
No.89 North Beijing Road, Qinghe District
Huaian 223001
China

Email: satishnaidu80@gmail.com

Mingui Zhang
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
China

Email: zhangmingui@huawei.com

Abdur Rashid Sangi
Huawei Technologies
No.156 Beiqing Rd. Haidian District
Beijing 100095
P.R. China

Email: sangi_bahrian@yahoo.com

Charles E. Perkins
Futurewei
2330 Central Expressway
Santa Clara 95050
Unites States

Email: charliep@computer.org

S.V.R Anand
Indian Institute of Science
Bangalore 560012
India

Email: anand@ece.iisc.ernet.in

ROLL
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

P. Thubert, Ed.
J. Pylakutty
Cisco
March 10, 2017

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2017.

Table of Contents

1. Introduction

The Routing Protocol for Low Power and Lossy Networks (LLN)(RPL) [RFC6550] is a generic Distance Vector protocol that is well suited for application in a variety of low energy Internet of Things (IoT) networks. RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) in which the root often acts as the Border Router to connect the RPL domain to the Internet. The root is responsible to select the RPL Instance that is used to forward a packet coming from the Internet into the RPL domain and set the related RPL information in the packets.

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

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

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

2. Terminology

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

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

3. New RPL Control Message Options

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

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

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

3.1. Via Information Option

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

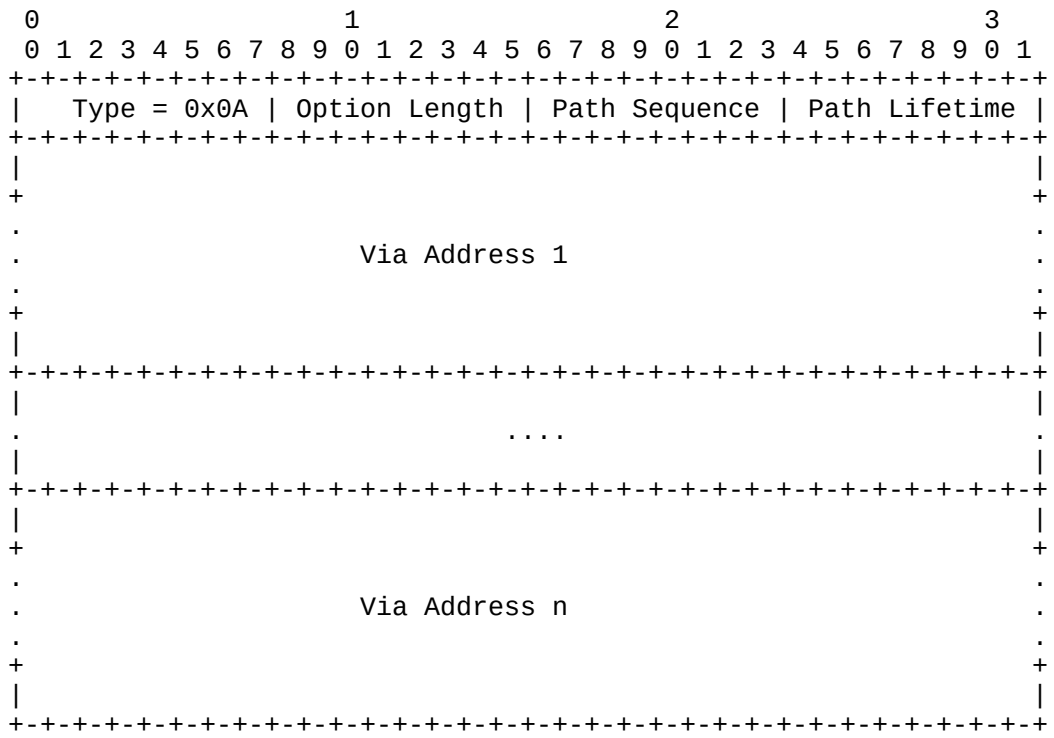

Figure 1: Via Information option format

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

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

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

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

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

4. Projected DAO

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

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

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

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

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

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

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

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

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

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

4.1. Non-storing Mode Projected DAO

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

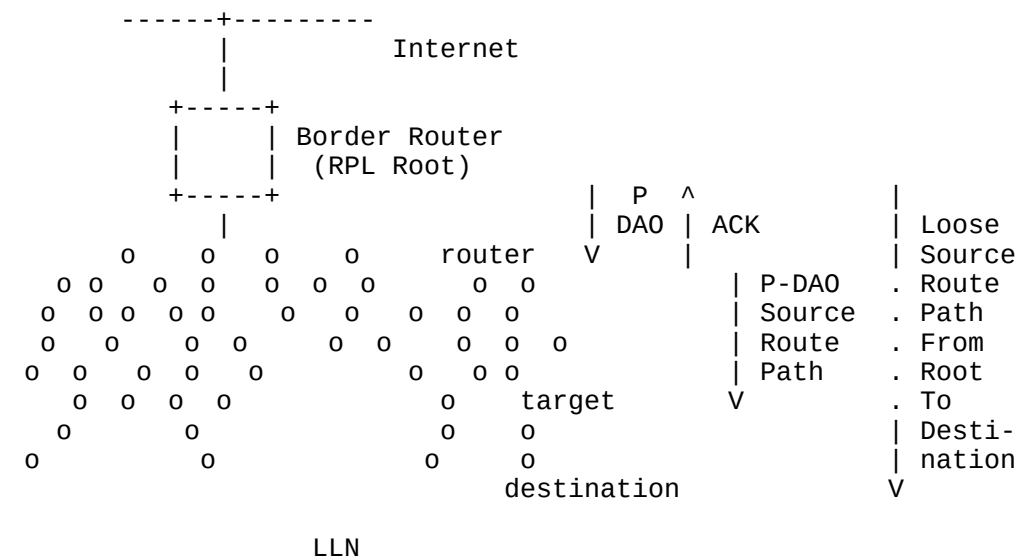

Figure 2: Projecting a non-storing route

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

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

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

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

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

4.2. Storing-Mode Projected DAO

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

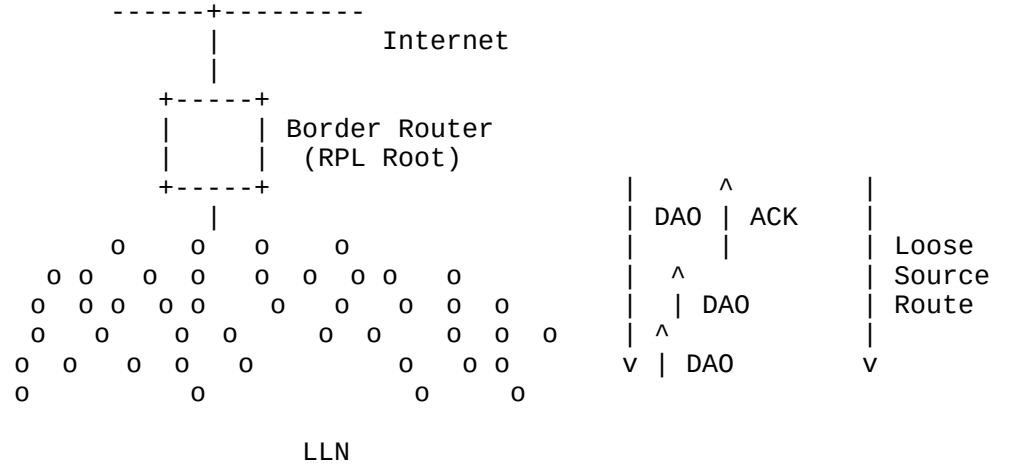

Figure 3: Projecting a route

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

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

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

The root sends the P-DAO directly to the egress node of the segment, which In that P-DAO, the destination IP address matches the Via Address in the last VIO. This is how the egress recognizes its role. In a similar fashion, the ingress node recognizes its role as it matches Via Address in the first VIO.

The egress node of the segment is the only node in the path that does not install a route in response to the P-DAO; it is expected to be already able to route to the target(s) on its own. It may either be the target, or may have some existing information to reach the target(s), such as a connected route or an already installed projected route. If one of the targets cannot be located, the node MUST answer to the root with a negative DAO-ACK listing the target(s) that could not be located (suggested status 10 to be confirmed by IANA).

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

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

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

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

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

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

A Path Lifetime of 0 in a Via Information option is used to clean up the state. The P-DAO is forwarded as described above, but the DAO is interpreted as a No-Path DAO and results in cleaning up existing state as opposed to refreshing an existing one or installing a new one.

5. Applications

5.1. Loose Source Routing in Non-storing Mode

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

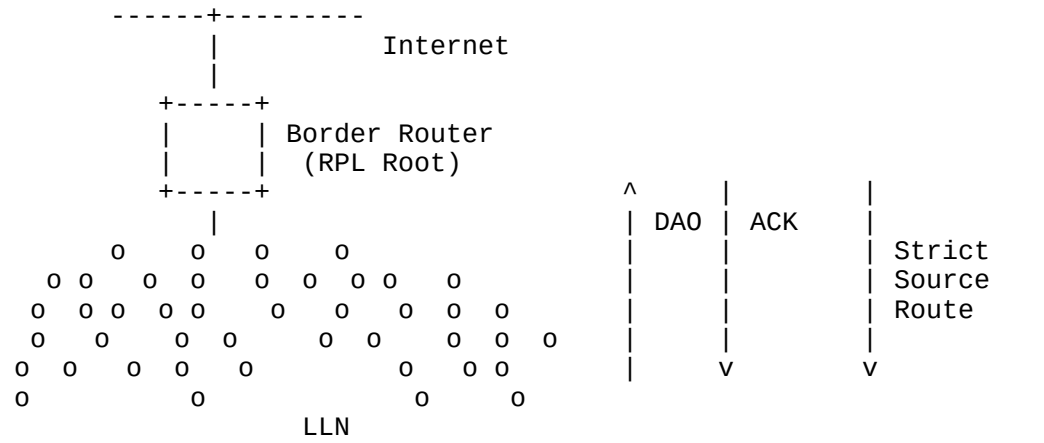

Figure 4: RPL non-storing mode of operation

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

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

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

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

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

5.2. Transversal Routes in storing and non-storing modes

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

- 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.

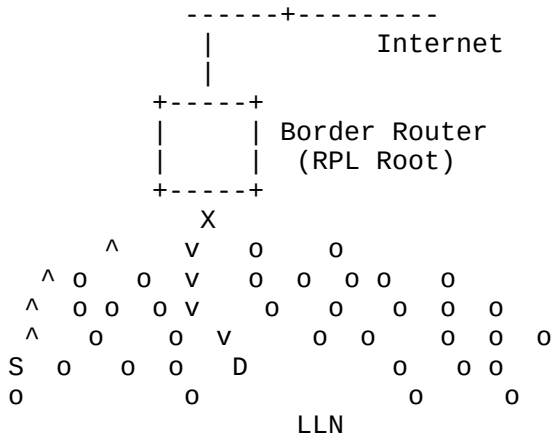

Figure 5: Routing Stretch between S and D via common parent X

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.

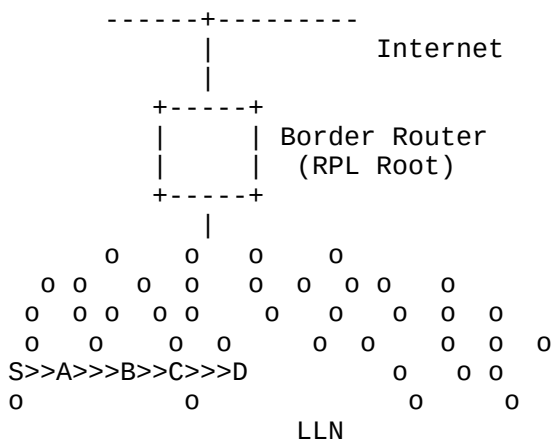

Figure 6: Projected Transversal Route

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

6. RPL Instances

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

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

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

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

7. Security Considerations

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

8. IANA Considerations

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

4: Non-Storing with Projected routes [this]

This document updates IANA registry for the RPL Control Message Options

0x0A: Via descriptor [this]

9. Acknowledgments

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

10. References

10.1. Normative References

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

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

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

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

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

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

10.2. Informative References

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

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

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

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

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

Appendix A. Examples

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

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

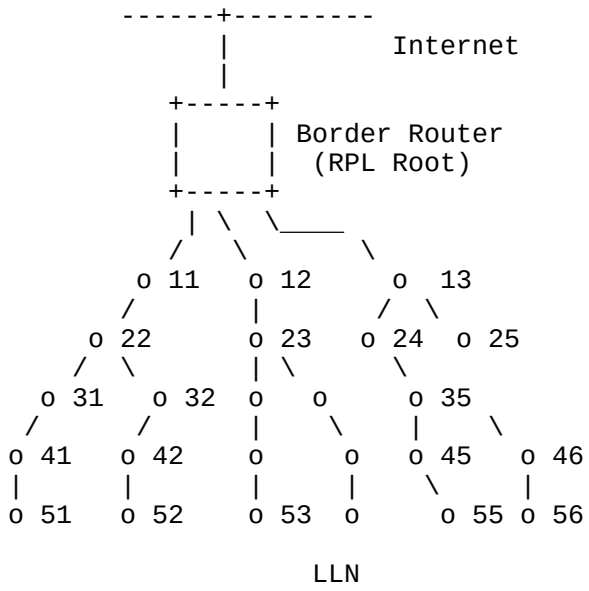

Figure 7: Example DODAG forming a logical tree topology

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

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

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

A.2. Projecting a storing-mode transversal route

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

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

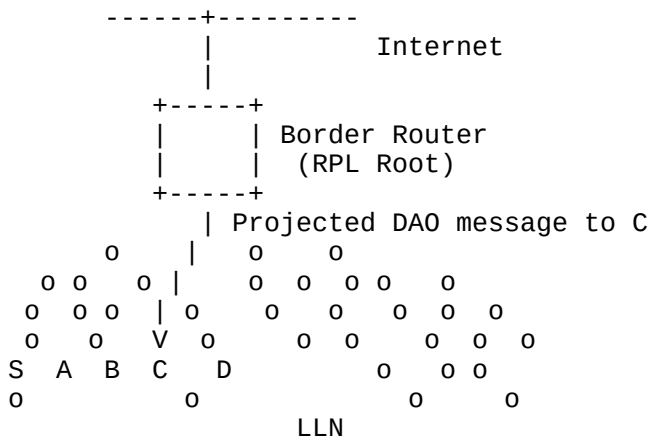

Figure 8: Projected DAO from root

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

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

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

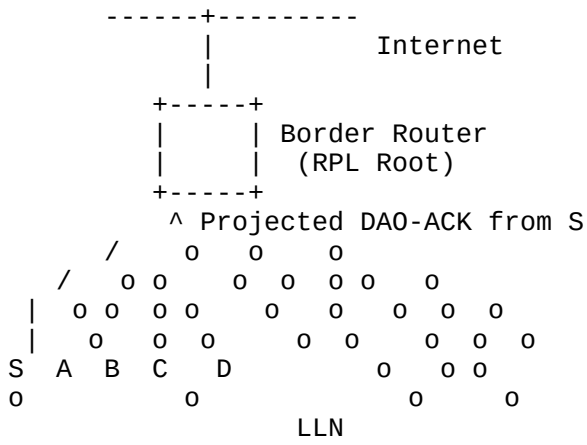

Figure 9: Projected DAO-ACK to root

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

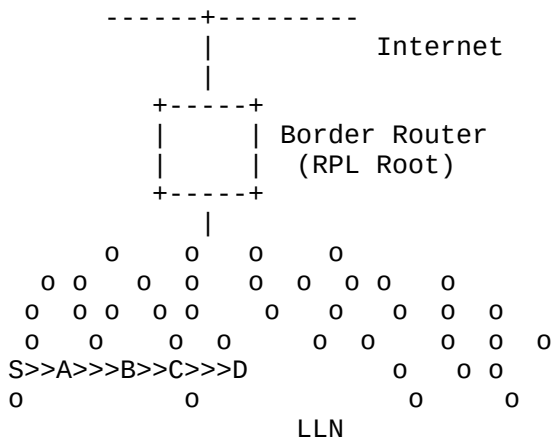

Figure 10: Projected Transversal Route

Authors' Addresses

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

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

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

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

ROLL
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2018

P. Thubert, Ed.
J. Pylakutty
Cisco
September 19, 2017

Root initiated routing state in RPL
draft-ietf-roll-dao-projection-02

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

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

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

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

This Internet-Draft will expire on March 23, 2018.

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 for application in a variety of low energy Internet of Things (IoT) networks. RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) in which the root often acts as the Border Router to connect the RPL domain to the Internet. The root is responsible to select the RPL Instance that is used to forward a packet coming from the Internet into the RPL domain and set the related RPL information in the packets.

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

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

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

2. Terminology

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

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

3. New RPL Control Message Options

Section 6.7 of 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; the former indicates a node to be reached and the latter specifies a parent that can be used to reach that node. Options may be factorized; one or more contiguous TIOs apply to the one or more contiguous Target options that immediately precede the TIOs in the RPL message.

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

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

3.1. Via Information Option

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

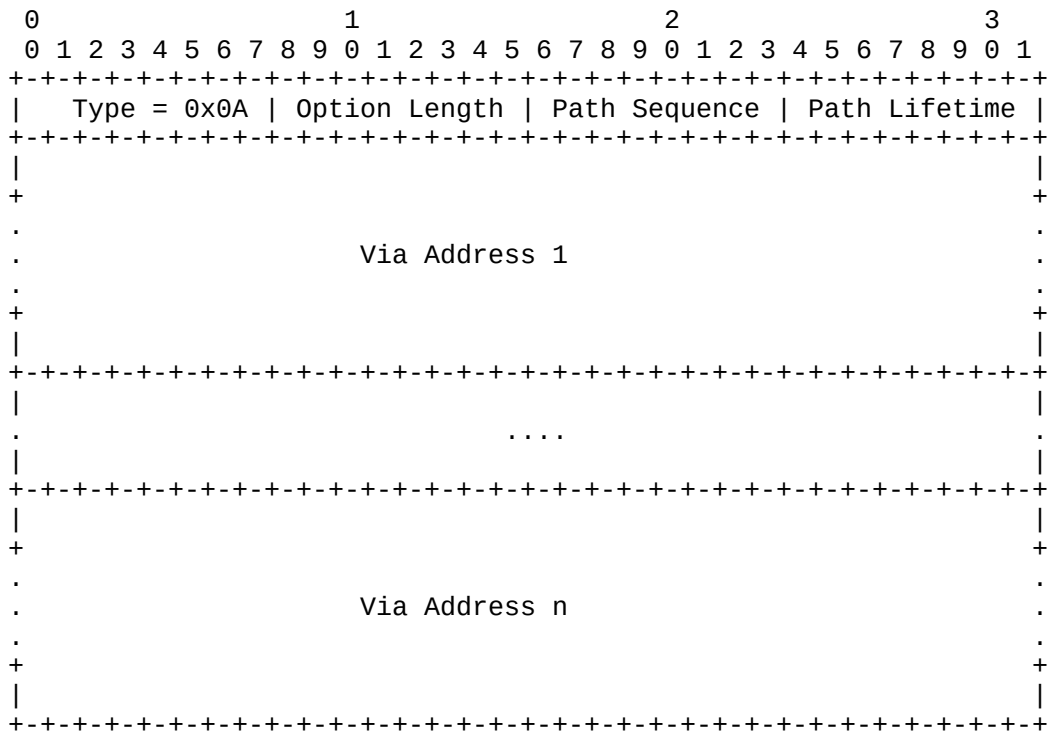

Figure 1: Via Information option format

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

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

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

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

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

4. Projected DAO

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

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

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

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

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

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

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

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

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

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

4.1. Non-storing Mode Projected DAO

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

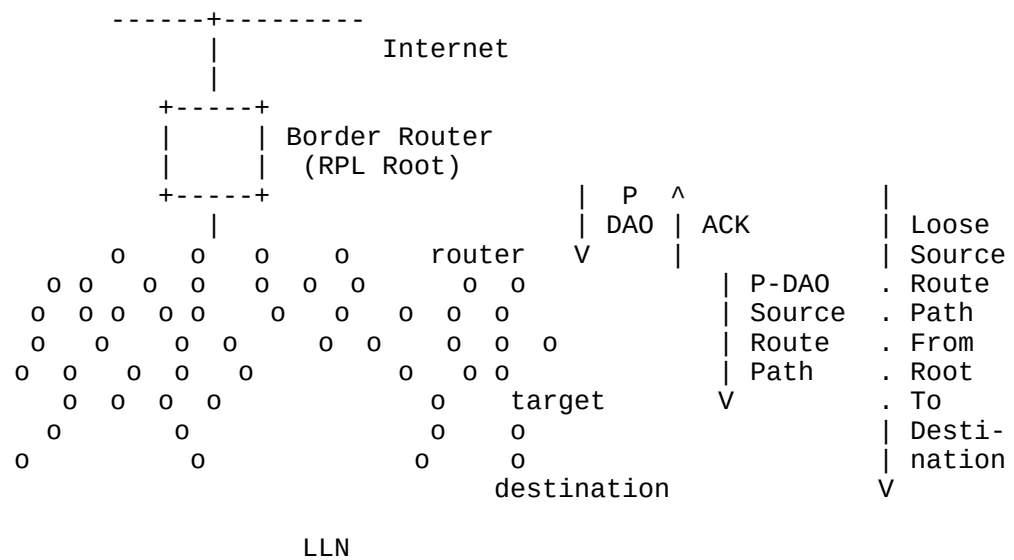

Figure 2: Projecting a Non-Storing route

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

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

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

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

In practice, the router will normally use the "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch" [RFC8025] to compress the RPL artifacts as indicated in the "6LoWPAN Routing Header" [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.

4.2. Storing-Mode Projected DAO

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

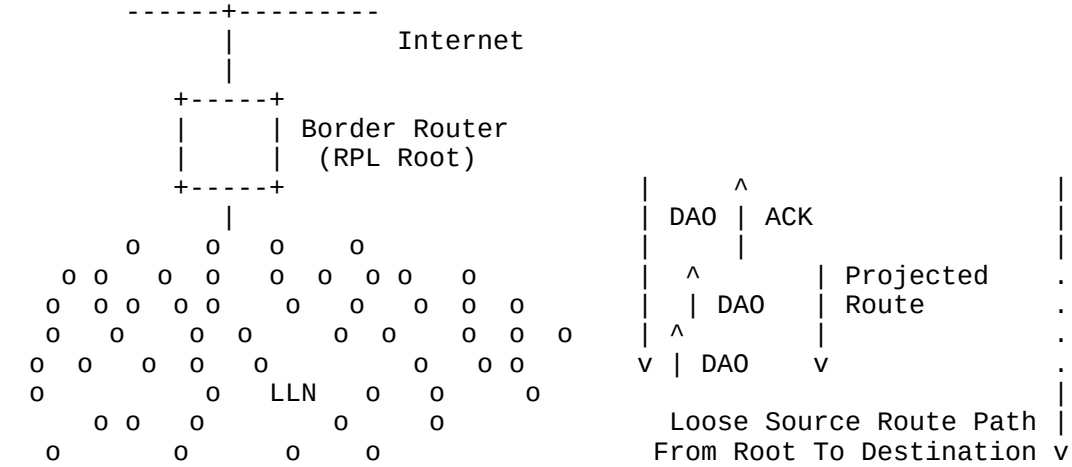

Figure 3: Projecting a route

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

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

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

The root sends the P-DAO directly to the egress node of the segment, which In that P-DAO, the destination IP address matches the Via Address in the last VIO. This is how the egress recognizes its role. In a similar fashion, the ingress node recognizes its role as it matches Via Address in the first VIO.

The egress node of the segment is the only node in the path that does not install a route in response to the P-DAO; it is expected to be already able to route to the target(s) on its own. It may either be the target, or may have some existing information to reach the target(s), such as a connected route or an already installed projected route. If one of the targets cannot be located, the node MUST answer to the root with a negative DAO-ACK listing the target(s) that could not be located (suggested status 10 to be confirmed by IANA).

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

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

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

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

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

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

A Path Lifetime of 0 in a Via Information option is used to clean up the state. The P-DAO is forwarded as described above, but the DAO is interpreted as a No-Path DAO and results in cleaning up existing state as opposed to refreshing an existing one or installing a new one.

5. Applications

5.1. Loose Source Routing in Non-storing Mode

A RPL implementation operating in a very constrained LLN typically uses the Non-Storing Mode of Operation 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.

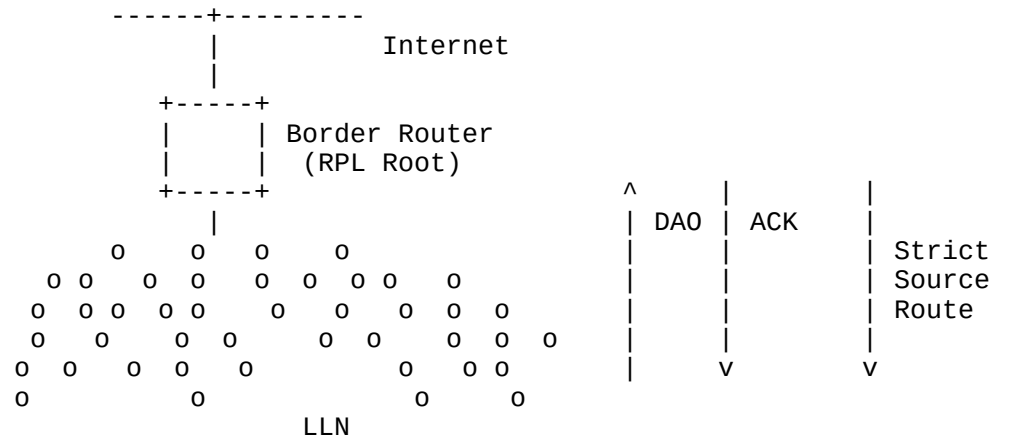

Figure 4: RPL non-storing mode of operation

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

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

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

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

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

5.2. Transversal Routes in storing and non-storing modes

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

- 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.

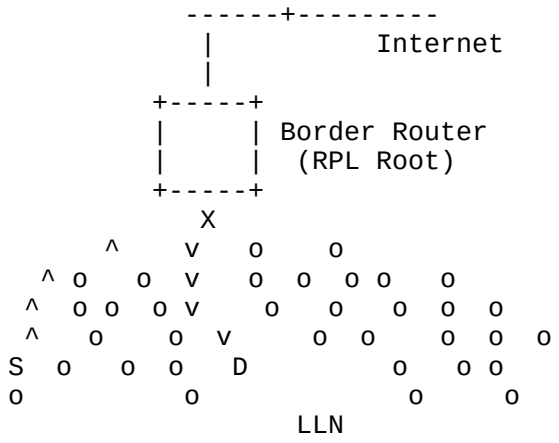

Figure 5: Routing Stretch between S and D via common parent X

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.

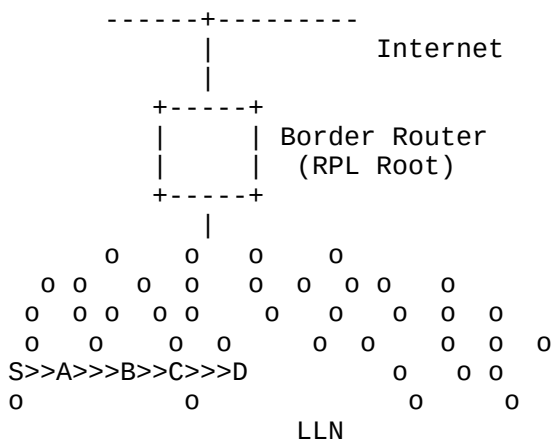

Figure 6: Projected Transversal Route

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

6. RPL Instances

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

- 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.

7. 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.

8. IANA Considerations

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

Code	Description	Reference
0x0A	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 Projected routes	This document
6	Storing mode of operation with Projected routes	This document

DIO Mode of operation

9. Acknowledgments

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

10. References

10.1. Normative References

[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>.

[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>.

10.2. Informative References

[I-D.ietf-6tisch-architecture] Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-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.

[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. Examples

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

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

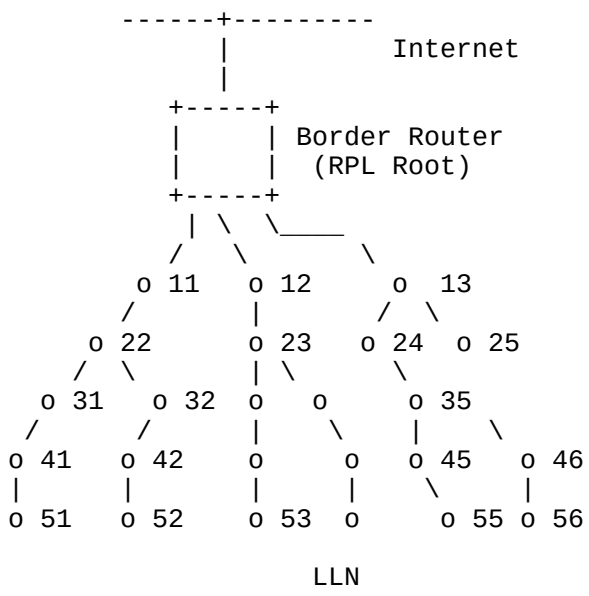

Figure 7: Example DODAG forming a logical tree topology

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

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

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

A.2. Projecting a storing-mode transversal route

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

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

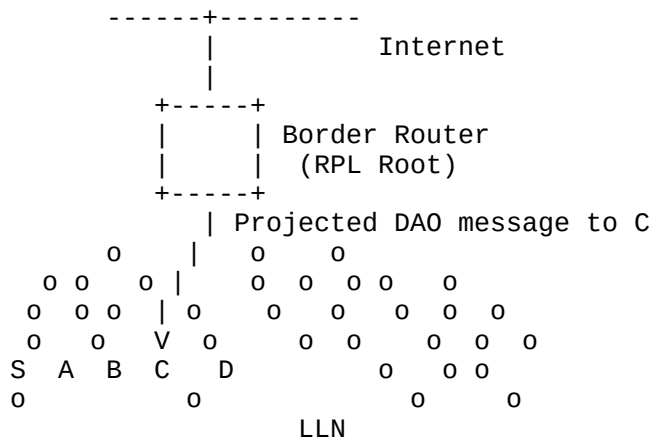

Figure 8: Projected DAO from root

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

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

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

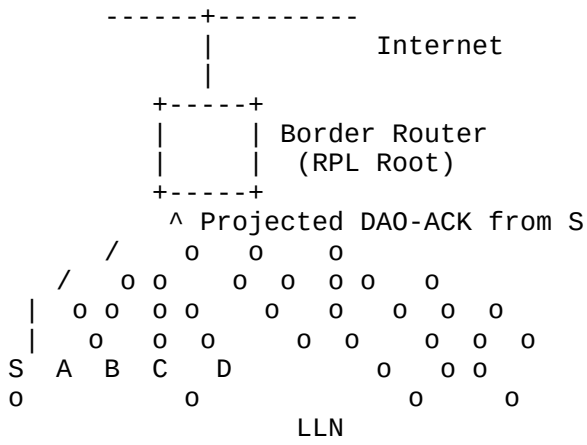

Figure 9: Projected DAO-ACK to root

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

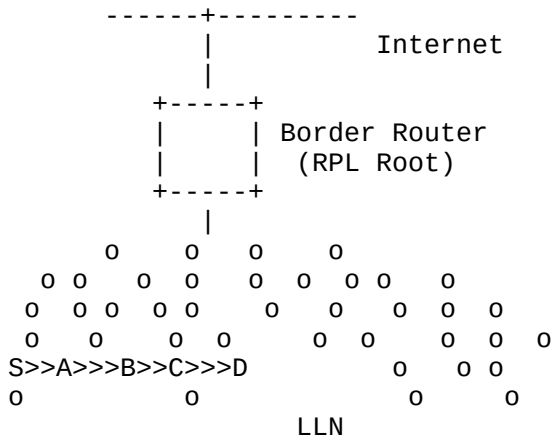

Figure 10: Projected Transversal Route

Authors' Addresses

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

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

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

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

ROLL Working Group
Internet-Draft
Updates: 6553, 6550, 8138 (if approved)
Intended status: Standards Track
Expires: May 3, 2018

M. Robles
Ericsson
M. Richardson
SSW
P. Thubert
Cisco

October 30, 2017

When to use RFC 6553, 6554 and IPv6-in-IPv6
draft-ietf-roll-useofrplinfo-19

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, RFC 6554 and IPv6-in-IPv6 encapsulation is required. This analysis provides the basis on which to design efficient compression of these headers. Additionally, this document updates the RFC 6553 adding a change to the RPL Option Type and the RFC 6550 to indicate about this change.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

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 what is the correct and the incorrect behaviour.

The related document A Routing Header Dispatch for 6LoWPAN (6LoRH) [RFC8138] defines a method to compress RPL Option information and Routing Header type 3 [RFC6554], an efficient IP-in-IP technique, and use cases proposed for the [Second6TischPlugtest] involving 6loRH.

2. Terminology and Requirements Language

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].

Terminology defined in [RFC7102] applies to this document: LBR, LLN, RPL, RPL Domain and ROLL.

RPL-node: A device which implements RPL, thus we can say that the device is RPL-capable or 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 DODAG is called RPL-aware-leaf.

RPL-not-capable: A device which does not implement RPL, thus we can say that 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.

pledge: a new device which seeks admission to a network. (from [I-D.ietf-anima-bootstrapping-keyinfra])

Join Registrar and Coordinator (JRC): a device which brings new nodes (pledges) into a network. (from [I-D.ietf-anima-bootstrapping-keyinfra])

Flag day: A "flag day" is a procedure in which the network, or a part of it, is changed during a planned outage, or suddenly, causing an outage while the network recovers [RFC4192]

2.1. 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

[RFC6553] states as showed 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 en route. The remaining bits serve as the option type.

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

Figure 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).

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].

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. When forwarding packets, implementations SHOULD use the same value as it was received. When originating new packets, implementations SHOULD have an option to determine which value to originate with, this option is controlled by the DIO option described below.

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.

3.2. Updates to RFC 8138

RPI-6LoRH header provides a compressed form for the RPL RPI [RFC8138]. It should be considered when the Option Type in RPL Option is decompressed, should take the value of 0x23 instead of 0x63.

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, when there is a mix of new nodes and old nodes, the new nodes may be put into a compatibilit mode until all of the old nodes are replaced or upgraded.

This can be done via a DODAG Configuration Option flag which will propogate through the network. Failure to receive this information

will cause new nodes to remain in compatibility mode, and originate 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 is as follows. The Flag field is the interesting field. The remaining fields states as defined in [RFC6550].

Flags: The 4-bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

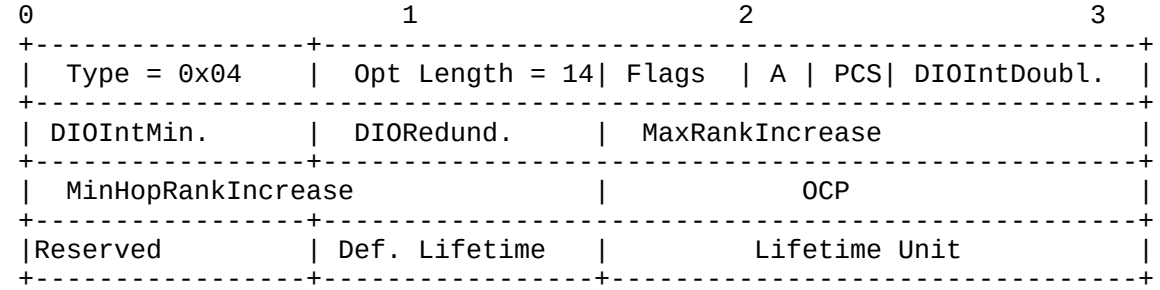

Figure 3: DODAG Configuration Option.

Bit number three of 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 DIO is sent with flag indicating the new RPI value.

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. (Destination Oriented Directed Acyclic Graph).

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 showed in Figure 1.

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.

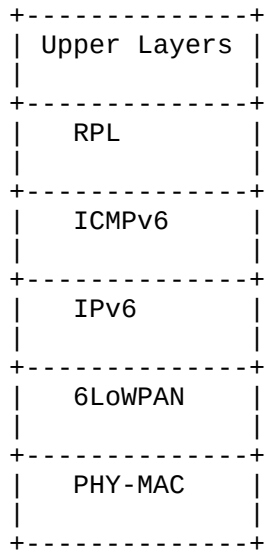

Figure 5: RPL Stack.

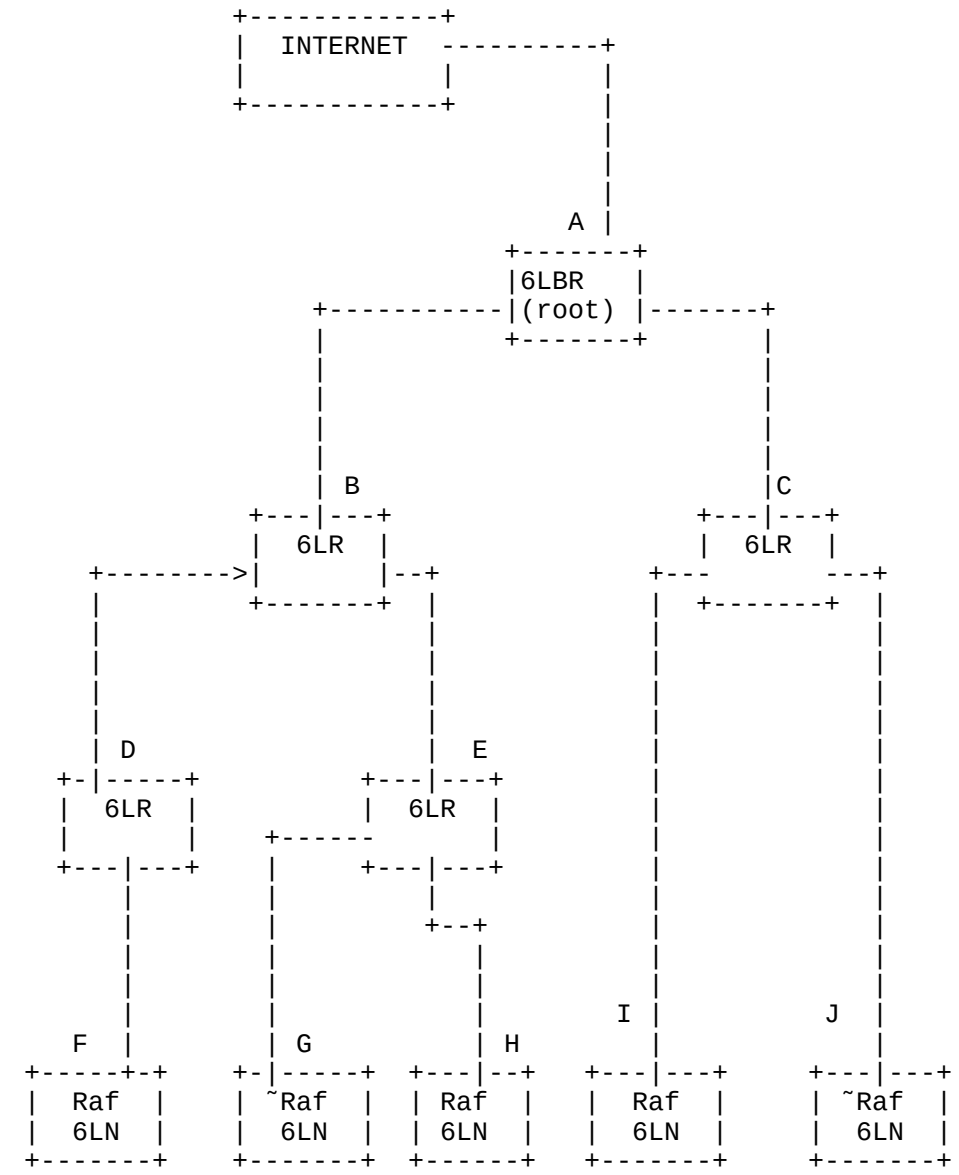

Figure 6: A reference RPL Topology.

Figure 2 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.

But, 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 refered to as an IPv6 node.

The 6LBR ("A") in the figure is the root of the Global DODAG.

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. We name these nodes as not-RPL-aware-leaf. (e.g. Section 6.1.4 Flow from not-RPL-aware-leaf to root) We describe 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 [RFC2460]. 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 MAY be 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.

Despite being legal to leave the RPI artifact in place, an intermediate router that needs to add an extension header (SHR3 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 corollory is that an SHR3 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 addresses.

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 RH 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 should 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. 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 to pick an appropriate priority or channel at each hop.

In the tables present in this document, the term "RPL aware leaf" is has been shortened to "Raf", and "not-RPL aware leaf" has been shortened to "˜Raf" to make the table fit in available space.

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 PIO option.

The following table itemizes which headers are needed in the following scenarios, and indicates if the IP-in-IP header must be inserted on a hop-by-hop basis, or when it can target the destination node directly. There are these possible situations: hop-by-hop necessary (indicated by "hop"), or destination address possible (indicated by "dst"). In all cases hop by hop MAY be used.

In cases where no IP-in-IP header is needed, the column is left blank.

In all cases the RPI headers are needed, since it identifies inconsistencies (loops) in the routing topology. In all cases the RH3 is not need because we do not indicate the route 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 6).

Interaction between	Use Case	IP-in-IP	IP-in-IP dst
Leaf - Root	Raf to root	No	--
	root to Raf	No	--
	root to ~Raf	No	--
	~Raf to root	Yes	root
Leaf - Internet	Raf to Int	No	--
	Int to Raf	Yes	Raf
	~Raf to Int	Yes	root
	Int to ~Raf	Yes	hop
Leaf - Leaf	Raf to Raf	No	--
	Raf to ~Raf	No	--
	~Raf to Raf	Yes	dst
	~Raf to ~Raf	Yes	hop

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

6.1. Storing Mode: Interaction between Leaf and Root

In this section we are going to describe 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] an 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 IP-in-IP 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	--	--	--

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 IP-in-IP header is required.

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

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)

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, encapsuladed 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	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

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

6.2. Storing Mode: Interaction between Leaf and Internet

In this section we are going to describe 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 IP-in-IP header is required.

Note: In this use case we use 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)

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	Interne t	6LBR	6LR_i	6LN
Inserted headers	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

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 IP-in-IP(RPI) header addressed either to the root, or hop-by-hop such that the root can remove the RPI header before passing upwards. (EDNOTE: we SHOULD recommend one or the other)

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	IPv 6	6LR_1	6LR_i [i=2,..,n]_	6LBR	Interne t
Inserte d headers	--	IP-in-IP(RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP(RPI)	--
Re-added headers	--	--	--	--	--
Modifie d headers	--	--	IP-in-IP(RPI)	--	--
Untouch ed headers	--	--	--	--	--

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

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 IP-in-IP header. The IP-in-IP 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 IPIP headers which are all addressed to it. (EDNOTE: this should go into [I-D.ietf-6man-rfc6434-bis])

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

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

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 we are going to describe 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 assume 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 IP-in-IP 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	--	--	--	--	--

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	--	--	--	--	RPI
Re-added headers	--	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Untouched headers	--	--	--	--	RPI(Ignored)

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 IP-in-IP header is addressed to the destination 6LN (Node F).

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

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 IPIP header is inserted.

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

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

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 IP-in-IP

header must be inserted. There are these possible situations: destination address possible (indicated by "dst"), to a 6LR, to a 6LN or to the root. In cases where no IP-in-IP header is needed, the column is left blank.

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 3).

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

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

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

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

In this section we are going to describe 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 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	--	--	--

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

7.1.2. on-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 will insert an RH3, and may optionally insert an RPI header. No IP-in-IP 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,RPI
Re-added headers	--	--	--
Modified headers	--	RH3	--
Untouched headers	--	--	--

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 present, the IPv6 node which does not understand it will ignore it (following 2460bis), thus encapsulation is not necesary. Due the complete knowledge of the topology at the root, the 6LBR may optionally address the IP-in-IP header to the last 6LR, such that it is removed prior to the IPv6 node.

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), RH3	--
Untouched headers	--	--	--	RPI

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 IP-in-IP 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	--	IP-in-IP(RPI)	--	--
Removed headers	--	--	--	IP-in-IP(RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP(RPI)	--
Untouched headers	--	--	--	--

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)

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 IP-in-IP header to that node. The 6LBR will zero the flow label upon entry in order to aid compression.

The RPI may be added or not as required by networks such as 6tisch. The RPI is unnecessary for loop detection.

Header	Internet	6LBR	6LR_i	6LN
Inserted headers	--	IP-in-IP (RH 3,opt:RPI)	--	--
Removed headers	--	--	--	IP-in-IP (RH3,opt:RPI)
Re-added headers	--	--	--	--
Modified headers	--	--	IP-in-IP (RH3,opt:RPI)	--
Untouched headers	--	--	--	--

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, the first 6LR (6LR_1) will add an RPI header inside a new IP-in-IP header. The IP-in-IP 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	--	IP-in-IP (RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP (RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP (RPI)	--	--
Untouched headers	--	--	--	--	--

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 IP-in-IP 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 IP-in-IP 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	--	IP-in-IP (RH3, opt:RPI)	--	--	--
Removed headers	--	--	--	IP-in-IP (RH3,RPI)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	IP-in-IP (RH3,RPI)	IP-in-IP (RH3,RPI)	--
Untouched headers	--	--	--	--	RPI

NonStoring: Summary of the use of headers from Internet to non-RPL-aware-leaf

7.3. Non-Storing Mode: Interaction between Leafs

In this section we are going to describe 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 an RPI header to the original packet, and send the packet upwards.

The originating node SHOULD put the RPI into an IP-in-IP 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 IP-in-IP header. It SHOULD be able to remove the RPI, as it was contained in an IP-in-IP header addressed to it. Otherwise, there MAY be an 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	IP-in-IP (RPI1)	--	IP-in-IP (RH3->6LN, opt RPI2)	--	--
Removed headers	--	--	IP-in-IP (RPI1)	--	IP-in-IP (RH3, opt RPI2)
Re-added headers	--	--	--	--	--
Modified headers	--	RPI1	--	RPI2	--
Untouched headers	--	--	--	--	--

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 an RPI (RPI_1) header which MUST be in an IP-in-IP header addressed to the root so that the 6LBR can remove this RPI. The 6LBR will then insert an RH3 inside a new IP-in-IP header addressed to the 6LN destination node. The RPI is optional from 6LBR to 6LR_id (RPI_2).

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

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 IP-in-IP header addressed to the root. The 6LBR will remove this RPI, and add it’s own IP-in-IP header containing an RH3 header and optional RPI (RPI_2).

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

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	--	IP-in-IP (RPI_1)	IP-in-IP (RH3)	--	--
Removed headers	--	--	IP-in-IP (RPI_1)	IP-in-IP (RH3, opt RPI_2)	--
Re-added headers	--	--	--	--	--
Modified headers	--	--	--	--	--
Untouched headers	--	--	--	--	--

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

8. Observations about the cases

8.1. Storing mode

[RFC8138] shows that the hop-by-hop IP-in-IP header can be compressed using IP-in-IP 6LoRH (IP-in-IP-6LoRH) header as described in Section 7 of the document.

There are potential significant advantages to having a single code path that always processes IP-in-IP headers with no options.

Thanks to the change of the RPI option type from 0x63 to 0x23, there is no longer any uncertainty about when to use an IP-in-IP header in the storing mode. A Hop-by-Hop Options Header containing the RPI option SHOULD always be added when 6LRs originate packets (without IP-in-IP headers), and IP-in-IP headers should always be added (addressed to the root when on the way up, to the end-host when on the way down) when a 6LR find that it needs to insert a Hop-by-Hop Options Header containing the RPI option.

8.2. Non-Storing mode

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 6LN immediately above that node. This means that the non-storing mode case can avoid ever using hop-by-hop IP-in-IP 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.

9. 6LoRH Compression cases

The [RFC8138] proposes a compression method for RPI, RH3 and IPv6-in-IPv6.

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 IP-in-IP and RPI compression headers. The type of this case is critical since IP-in-IP is encapsulating a RPI header.

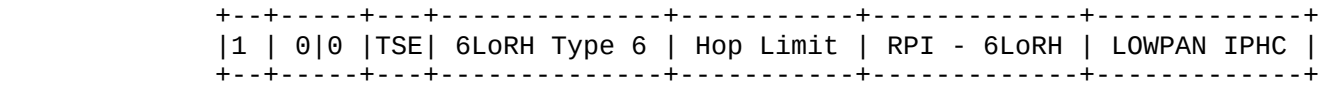

Figure 9: Critical IP-in-IP (RPI).

10. IANA Considerations

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

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 10: 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 11: DODAG Configuration Option Flag to indicate the RPI-flag-day.

11. Security Considerations

The security considerations covering of [RFC6553] and [RFC6554] apply when the packets get into RPL Domain.

The IPIP 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.

Nodes outside of the LLN will need to pass IPIP traffic through the RPL root to perform this attack. To counter, the RPL root SHOULD either restrict ingress of IPIP 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 described in [I-D.ietf-6lo-backbone-router]. In this case the BCP38 filtering needs to take this into account.

Nodes with the LLN can use the IPIP 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.

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 IPIP traffic entering the LLN as described above will make sure that any attack that is mounted must originate 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 IPIP traffic SHOULD be allowed to pass through the RPL root, such as the IPIP 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 IPIP 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 IPIP headers using forged source addresses. If the attack requires bi-directional communication, then IPIP provides no advantages.

[RFC2473] suggests that tunnel entry and exit points can be secured, via the "Use IPsec". This 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 IPIP 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 IPIP 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 RPL instanceID, 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, an RPI header is to be inserted by the RPL root on traffic entering the LLN by first inserting an IPIP 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 IPIP headers, and this document does not change that analysis.

12. Acknowledgments

This work is partially funded by the FP7 Marie Curie Initial Training Network (ITN) METRICS project (grant agreement No. 607728).

The authors would like to acknowledge the review, feedback, and comments of (alphabetical order): Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, C. M. Heard, Rahul Jadhav, Matthias Kovatsch, Peter van der Stok, Xavier Vilajosana and Thomas Watteyne.

13. References

13.1. 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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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., "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-04 (work in progress), July 2017.

[I-D.ietf-6man-rfc6434-bis]
Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", draft-ietf-6man-rfc6434-bis-02 (work in progress), October 2017.

[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-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]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-12 (work in progress), October 2017.

[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-08 (work in progress), October 2017.

[I-D.ietf-roll-dao-projection]
Thubert, P. and J. Pylakutty, "Root initiated routing state in RPL", draft-ietf-roll-dao-projection-02 (work in progress), September 2017.

[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>.

[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>.

[Second6TischPlugtest] "2nd 6Tisch Plugtest", <http://www.ietf.org/mail-archive/web/6tisch/current/pdfgDMQcdCkRz.pdf>.

Authors' Addresses

Maria Ines Robles
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: maria.ines.robles@ericsson.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
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2017

R. Jadhav, Ed.
R. Sahoo
Z. Cao
Huawei Tech

February 14, 2017

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 18, 2017.

Copyright Notice

Table of Contents

1. Introduction

RPL [RFC6550] specifies a proactive distance-vector based routing scheme. The specification has an optional messaging in the form of DAO messages using which the 6LBR can learn route towards any of the nodes. In storing mode, DAO messages would result in routing entries

been created on all intermediate hops from the node's parent all the way towards the 6LBR.

RPL also allows use of No-Path DAO (NPDAO) messaging to invalidate a routing path and thus releasing of any resources utilized on that path. A No-Path DAO is a DAO message with route lifetime of zero, signaling route invalidation for the given target. This document studies the problems associated with the current use of No-Path DAO messaging, which creates route inefficiency and inconsistence. This document also discusses the requirements for an optimized No-Path DAO messaging scheme.

1.1. Requirements Language and Terminology

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

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

Common Ancestor node: 6LR node which is the first common node on the old and new path for the child node.

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

New parent: Parent 6LR node after switching to the new path.

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

Reverse NPDAO: A No-Path DAO message which traverses downstream in the network.

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

This document also uses terminology described in [RFC6550] and [RFC6775].

1.2. Current No-Path DAO messaging

RPL introduced No-Path DAO messaging in the storing mode so that the node switching its current parent can inform its parents and ancestors to invalidate the existing route. Subsequently parents or ancestors would release any resources (such as the routing entry) it maintains on behalf of that child node. The No-Path DAO message

always traverses the RPL tree in upward direction, originating at the target node itself.

For the rest of this document consider the following topology:

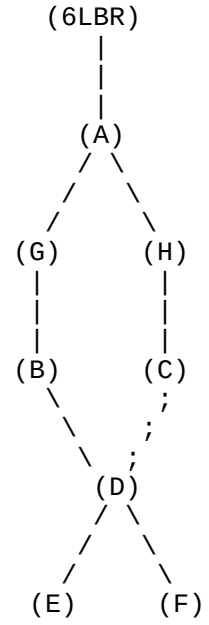

Figure 1: Sample topology

Node (D) is connected via preferred parent (B). (D) has an alternate path via (C) towards the BR. Node (A) is the common ancestor for (D) for paths through (B)-(G) and (C)-(H). When (D) switches from (B) to (C), [RFC6550] suggests sending No-Path DAO to (B) and regular DAO to (C).

1.3. Cases when No-Path DAO may be used

There are following cases in which a node switches its parent and may employ No-Path DAO messaging:

Case I: Current parent becomes unavailable because of transient or permanent link or parent node failure.

Case II: The node finds a better parent node i.e. the metrics of another parent is better than its current parent.

Case III: The node switches to a new parent whom it "thinks" has a better metric but does not in reality.

The usual steps of operation when the node switches the parent is that the node sends a No-Path DAO message via its current parent to invalidate its current route and subsequently it tries to establish a new routing path by sending a new DAO via its new parent.

1.4. Why No-Path DAO is important?

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

2. Problems with current No-Path DAO messaging

There are following problems with the usage of current NP-DAO messaging

2.1. Lost NP-DAO due to link break to the previous parent

When the node switches its parent, the NPDAO is to be sent via its previous parent and a regular DAO via its new parent. In cases where the node switches its parent because of transient or permanent parent link/node failure then the NPDAO message is bound to fail. RPL assumes communication link with the previous parent for No-Path DAO messaging.

RPL mentions use of route lifetime to remove unwanted routes in case the routes could not be refreshed. But route lifetimes in case of LLNs could be substantially high and thus the route entries would be stuck for long.

2.2. Invalidate routes to dependent nodes of the switching node

No-path DAO is sent by the node who has switched the parent but it does not work for the dependent child nodes below it. The specification does not specify how route invalidation will work for sub-childs, resulting in stale routing entries on behalf of the sub-childs on the previous route. The only way for 6LR to invalidate the route entries for dependent nodes would be to use route lifetime expiry which could be substantially high for LLNs. In the example

topology, when Node (D) switches its parent, Node (D) generates an NPDAO on its behalf. Post switching, Node (D) transmits a DIO with incremented DTSN so that child nodes, node (E) and (F), generate DAOs to trigger route update on the new path for themselves. There is no NPDAO generated by these child nodes through the previous path resulting in stale entries on nodes (B) and (G) for nodes (E) and (F).

2.3. Route downtime caused by asynchronous operation of NPDAO and DAO

A switching node may generate both an NPDAO and DAO via two different paths at almost the same time. There is a possibility that an NPDAO generated may invalidate the previous route and the regular DAO sent via the new path gets lost on the way. This may result in route downtime thus impacting downward traffic for the switching node. In the example topology, consider Node (D) switches from parent (B) to (C) because the metrics of the path via (C) are better. Note that the previous path via (B) may still be available (albeit at relatively bad metrics). An NPDAO sent from previous route may invalidate the existing route whereas there is no way to determine whether the new DAO has successfully updated the route entries on the new path.

An implementation technique to avoid this problem is to further delay the route invalidation by a fixed time interval after receiving an NPDAO, considering the time taken for the new path to be established. Coming up with such a time interval is tricky since the new route may also not be available and it may subsequently require more parent switches to establish a new path.

3. Requirements for the No-Path DAO Optimization

We identify the following requirements for the NP-DAO optimization.

3.1. Req#1: Tolerant to the link failures to the previous parents

When the switching node send the NP-DAO message to the previous parent, it is normal that the link to the previous parent is prone to failure. Therefore, it is required that the NP-DAO message MUST be tolerant to the link failure during the switching.

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

While switching the parent node and sending NP-DAO message, it is required that the routing entries to the dependent nodes of the switching node will be updated accordingly on the previous parents and other relevant upstream nodes.

3.3. Req#3: No impact on traffic while NP-DAO operation in progress

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

4. Existing Solution

4.1. NP-DAO can be generated by the parent node who detects link failure to the child

RPL states mechanisms which could be utilized to clear DAO states in a sub-DODAG. [RFC6550] Section 11.2.2.3 states "With DAO inconsistency loop recovery, a packet can be used to recursively explore and clean up the obsolete DAO states along a sub-DODAG".

Thus in the sample topology in Figure 1, when Node (B) detects link failure to (D), (B) has an option of generating an NP-DAO on behalf of Node (D) and its sub-childs, (E) and (F).

This section explains why generation of an NP-DAO in such cases may not function as desired. Primarily the DAO state information in the form of Path Sequence plays a major role here. Every target is associated with a Path Sequence number which relates to the latest state of the target. [RFC6550] Section 7.1 explains the semantics of Path Sequence number. The target node increments the Path Sequence number every time it generates a new DAO. The router nodes en-route utilize this Path Sequence number to decide the freshness of target information. If a non-target node has to generate an NP-DAO then it could use following two possibilities with Path Sequence number:

Let the Path Sequence number of old regular DAO that flowed through (B) be x. The subsequent regular DAO generated by Node (D) will have sequence number x+1.

i. Node (B) uses the previous Path Sequence number from the regular DAO i.e. NP-DAO(pathseq=x)

ii. Node (B) increments the Path Sequence number i.e. NP-DAO(pathseq=x+1)

In case i, the NP-DAO(pathseq=x) will be dropped by all the intermediate nodes since the semantics of Path Sequence number dictates that any DAO with an older Path Sequence number be dropped.

In case ii, there is a risk that the NP-DAO(pathseq=x+1) traverses up the DODAG and invalidates all the routes till the root and then the regular DAO(pathseq=x+1) from the target traverses upwards. In this case the regular DAO(pathseq=x+1) will be dropped from common ancestor node to the root. This will result in route downtime.

Another problem with this scheme is its dependence on the upstream neighbor to detect that the downstream neighbor is unavailable. There are two possibilities by which such a detection might be put to work:

i. There is P2P traffic from the previous sub-DODAG to any of nodes in the sub-tree which has switched the path. In the above example, lets consider that Node (G) has P2P traffic for either of nodes (D), (E), or (F). In this case, Node (B) will detect forwarding error while forwarding the packets from Node (B) to (D). But dependence on P2P traffic may not be an optimal way to solve this problem considering the reactive approach of the scheme. The P2P traffic pattern might be sparse and thus such a detection might kick-in too late.

ii. The other case is where Node (B) explicitly employs some mechanism to probe directly attached downstream child nodes. Such kind of schemes are seldom used.

4.2. NP-DAO can be generated once the link is restored to the previous parent

This scheme solves a specific scenario of transient links. The child node can detect that the connection to previous parent is restored and then transmit an NP-DAO to the previous parent to invalidate the route. This scheme is stateful, thus requires more memory and solves a specific scenario.

5. Proposed changes to NPDAO signaling

5.1. Change in NPDAO semantics

As described in Section 1.2, currently the NPDAO originates at the node switching the parent and traverses upstream towards the root. In order to solve the problems as mentioned in Section 2, the draft proposes to change the way NPDAO originates and traverses the network. The new NPDAO proposed does not originate at the node but instead originates at a common ancestor node between the new and old path. The trigger for the common ancestor node to generate this NPDAO is the change in the next hop for the node on reception of an update message in the form of regular DAO for the target.

In the Figure 1, when node D decides to switch the path from B to C, it sends a regular DAO to node C with reachability information containing target as address of D and a incremented path sequence number. Node C will update the routing table based on the reachability information in DAO and in turn generate another DAO with the same reachability information and forward it to H. Node H also follows the same procedure as Node C and forwards it to node A. When node A receives the regular DAO, it finds that it already has a routing table entry on behalf of the target address of node D. It finds however that the next hop information for reaching node D has changed i.e. the node D has decided to change the paths. In this case, Node A which is the common ancestor node for node D along the two paths (previous and new), may generate an NPDAO which traverses downwards in the network. The document in the subsequent section will explain the message format changes to handle this downward flow of NPDAO.

5.2. DAO message format changes

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

The draft proposes a change in DAO message to contain "Invalidate previous route" (I) bit. This I-bit which is carried in regular DAO message, signals the common ancestor node to generate a downstream NPDAO on behalf of the target node. The I-bit is carried in the transit container option which augments the reachability information for a given set of RPL Target(s).

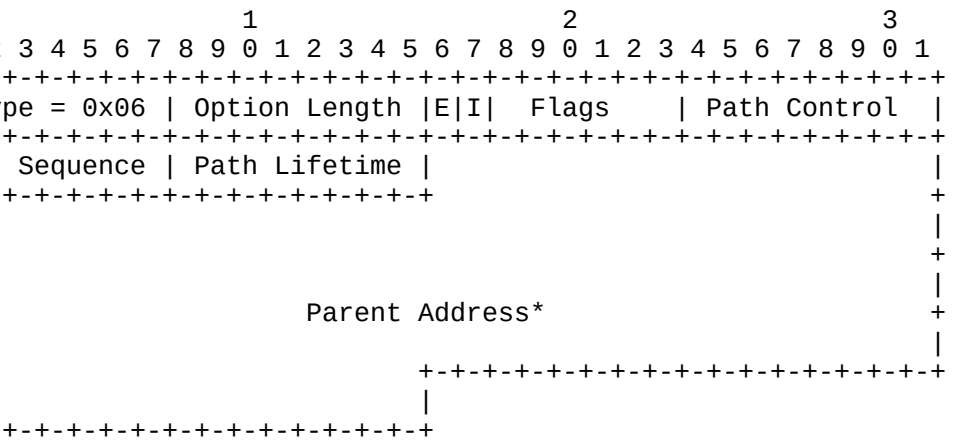

Figure 2: Updated Transit Information Option (New I flag added)

I (Invalidate previous route) bit: 1 bit flag. The 'I' flag is set by the target node to indicate that it wishes to invalidate the previous route by a common ancestor node between the two paths.

The NPDAO thus generated by the common ancestor node needs to traverse downstream. An additional flag called as "Reverse NPDAO" (R) is added in the base DAO object to signal this change.

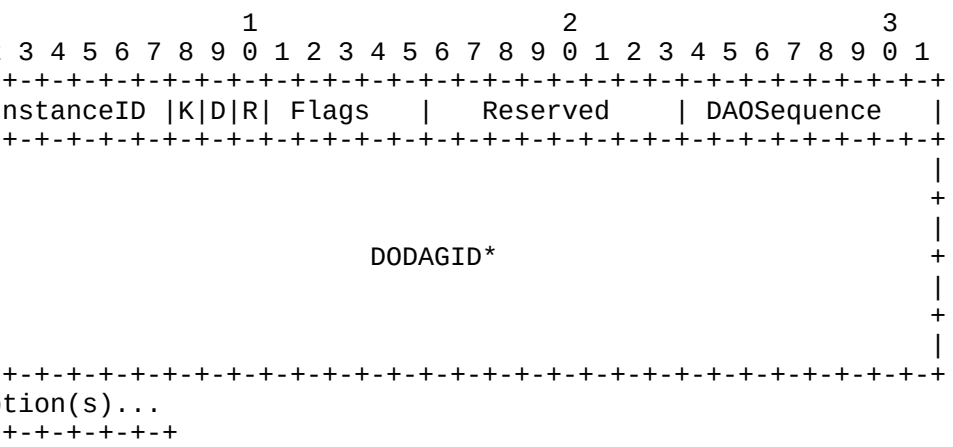

Figure 3: Updated DAO base object (New R flag added)

R (Reverse DAO) bit: 1 bit flag. The 'R' flag is used to signal that the DAO traverses downwards.

5.2.1. Path Sequence number in the reverse NPDAO

Every DAO message may contain a Path Sequence in the transit information option to identify the freshness of the DAO message. The Path Sequence in the downward NPDAO generated by common ancestor should use the same Path Sequence number present in the regular DAO message.

5.3. Example messaging

In Figure 1, node (D) switches its parent from (B) to (C). The sequence of actions is as follows:

1. Node D switches its parent from node B to node C

2. D sends a regular DAO(tgt=D,pathseq=x+1,I_flag=1) in the updated path to C

3. C checks for routing entry on behalf of D, since it cannot find an entry on behalf of D it creates a new routing entry and forwards the reachability information of the target D to H in a DAO.

4. Similar to C, node H checks for routing entry on behalf of D, cannot find an entry and hence creates a new routing entry and forwards the reachability information of the target D to H in a DAO.

5. A receives the DAO, and checks for routing entry on behalf of D. It finds a routing entry but checks that the next hop for target D is now changed. Node A checks the I_flag and generates downstream NPDAO(tgt=D,pathseq=x+1,R_flag=1) to previous next hop for target D which is G. Subsequently, A updates the routing entry and forwards the reachability information of target D upstream DAO(tgt=D,pathseq=x+1,I_flag=x) (the I_flag carries no significance henceforth).

6. Node G receives the downstream NPDAO and invalidates routing entry of target D and then checks the reverse (R) flag and forwards the (un)reachability information downstream to B.

7. Similarly, B processes the downstream NPDAO by invalidating the routing entry of target D and then checks the reverse (R) flag and forwards the (un)reachability information downstream to D.

8. D ignores the downstream NPDAO since the target is itself.

5.4. Other considerations

5.4.1. Dependent Nodes invalidation

Current RPL [RFC6550] does not provide a mechanism for route invalidation for dependent nodes.

This section describes approaches for invalidating routes of dependent nodes if the implementation chooses to solve this problem. The common ancestor node realizes that the paths for dependent nodes have changed (based on next hop change) when it receives a regular DAO on behalf of the dependent nodes. Thus dependent nodes route invalidation can be handled in the same way as the switching node. Note that there is no way that dependent nodes can set the I_flag in the DAO message selectively since they are unaware that their parent/ grand parent node is switching paths. There are two ways to handle dependent node route invalidation:

1. One way to resolve is that the common ancestor does not depend upon the I_flag to generate the reverse NPDAO. The only factor it makes the decision will be based on next_hop change for an existing target to generate the NPDAO. Thus when the switching nodes and all the below dependent nodes advertise a regular DAO, the common ancestor node will detect a change in next hop and generate NPDAO for the same target as in the regular DAO.

2. Another way is that the nodes always set the I_flag whenever they send regular DAO. Thus common ancestor will first check whether I_flag is set and then check whether the next_hop has changed and subsequently trigger NPDAO if required.

This document recommends the approach in point 2. The advantage with I_flag is that the generation of downstream NPDAO is still controlled by the target node and thus is still in control of its own routing state.

6. Acknowledgements

We would like to thank Cenk Gundogan, Simon Duquennoy and Pascal Thubert for their review and insightful comments.

7. IANA Considerations

IANA is requested to allocate bit 11 in the DAO base object defined in RPL [RFC6550] section 6.4 for reverse ’R’ NPDAO flag.

IANA is requested to allocate bit 18 in the Transit Information Option defined in RPL [RFC6550] section 6.7.8 for Invalidate route 'I' flag.

8. Security Considerations

TBA

9. References

9.1. Normative References

[CONTIKI] Thingsquare, "Contiki: The Open Source OS for IoT", 2012, <http://www.contiki-os.org>.

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

9.2. Informative References

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <http://www.rfc-editor.org/info/rfc5191>.

[RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Ed., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", RFC 6345, DOI 10.17487/RFC6345, August 2011, <http://www.rfc-editor.org/info/rfc6345>.

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

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <http://www.rfc-editor.org/info/rfc6775>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

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

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

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

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

Zhen Cao
Huawei Tech
W Chang'an Ave
Beijing 560037
China

Email: zhencao.ietf@gmail.com

roll
Internet-Draft
Intended Status: Standards Track
Expires: May 2, 2018

M.Qasem
A.Al-Dubai
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.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at

http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1 Introduction

IPv6 Routing Protocol for LLNs (RPL) [RFC6550] defined two OFs to optimize the path selection towards the root node, namely, the OF zero (OF0) [RFC6552], and the Minimum Rank with Hysteresis OF (MRHOF)[RFC6719]. The Destination Oriented Directed Acyclic Graph (DODAG) construction is built by the RPL OF, that specify how nodes select the preferred parent node by translating one or more metrics into the rank value.

The used OF calculates the rank based on some routing metrics [RFC 6551] such as hop-count, delay, energy, and so forth. The parent node in RPL can serve more than one child if it is chosen by them as preferred parent. Consequently, the overloaded preferred parents will become fragile nodes as their energy risks to drain much quicker than other nodes.

Having conducted simulation experiments and rigours analysis, it is concluded that the current OFs lead to build a topology that suffers from an unbalanced load traffic in bottleneck nodes especially for the first hop nodes (i.e., from the root). Consequently, this problem has a crucial impact on the lifetime of these types of nodes. The battery depletion of that overloaded parent node may affect the network reliability negatively.

This challenging problem is still an open issue. In an attempt to overcome this problem, this draft proposes a new OF to mitigate the overusing of the bottleneck node to prolong its battery lifetime.

This draft proposes an extended Objective Function(OF) that balances the number of children nodes for the overloaded nodes to ensure node lifetime maximization in RPL and can be summarized as follows. First, a new RPL metric has been used to balance the load traffic among the bottleneck nodes. Second, the DODAG Information Object (DIO) message has been amended by injecting the IP address of the chosen parent before broadcasting it. Third, a new utilization technique has been proposed for the amended DIO message to avoid increasing the overhead of the handshaking and acknowledgment processes. Simulation experiments have been conducted to validate the extended OF performance as detailed in Appendix A.

1.1 Terminology

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

2 DODAG construction in a nutshell

RPL is a proactive distance vector routing protocol designed for LLNs [RFC6550], it constructs a DODAG using a certain OF that suits the application requirements. Essentially, RPL relies on a DODAG Information Object (DIO) control message to build the DODAG.

Thus, the starting point begins when the root node broadcasts the DIO message to the downstream neighbor nodes. As soon as the closest node receives the message, it can decide whether to join this DODAG or not based on the calculated rank according to the equations (1) and (2) [RFC6719].

Rank(N) = Rank(PN) + RankIncrease (1)

RankIncrease = Step * MinHopRankIncrease (2)

Where Step represents a scalar value and MinHopRankIncrease represents the minimum RPL parameter. If the node decides to join, then it adds the DIO sender to the candidate parent list. Next, the preferred parent, i.e. the next hop to the root, will be chosen based on the rank from this list to receive all traffic from the child node. Then, it computes its own rank with a monotonical increase according to the selected OF.

After that, the node propagates its own DIO with all updated information to all its neighbors including the preferred parent. [RFC 6551] defined the number of node metrics/constraints (e.g. hop count and energy) and the link metrics/constraints (e.g. ETX and throughput) that might be used in the OFs [RFC6552][RFC6719].

3 Load balancing in RPL

RPL is designed with several robust features such as exiguous delay, quick configuration, loop-free topology, and self-healing. However, the load imbalance is considered as a significant weakness in this protocol. More specifically, RPL is dealing with non-uniform distribution in large-scale LLNs, which may lead to unequal data traffic distribution. Consequently, the energy of the overloaded nodes will be drained much faster than other nodes. Furthermore, this problem has more harmful impacts if the overloaded node is a bottleneck node (i.e. with the first hop to the root) as shown in Figure 1 for node A and B.

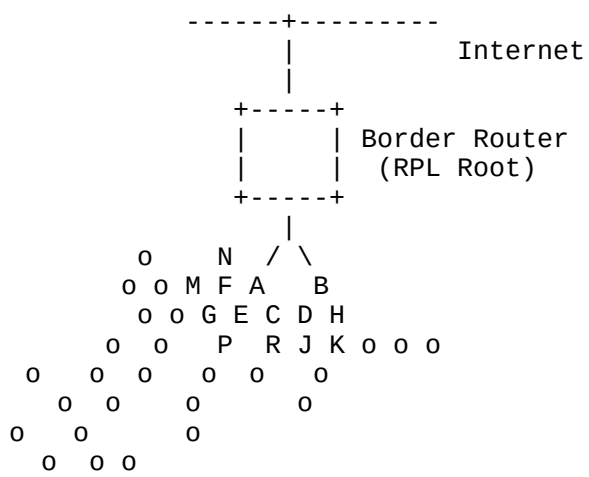

Figure 1: Bottleneck nodes in RPL

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:

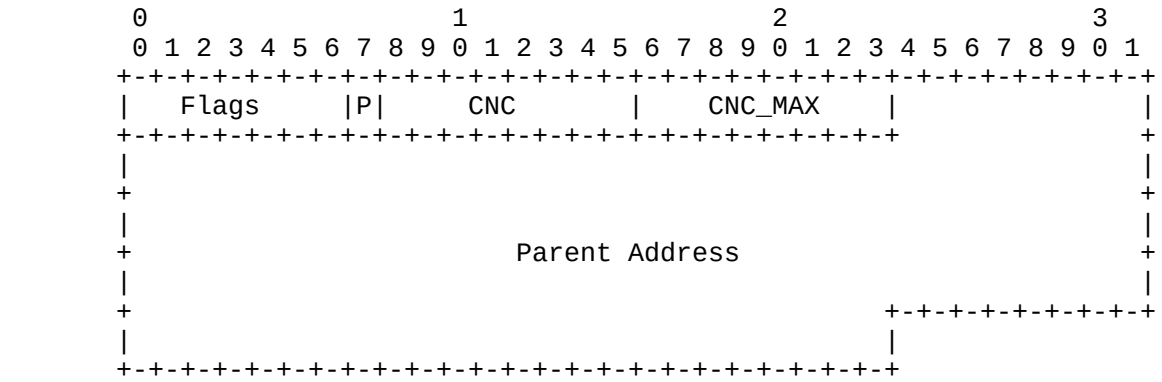

Figure 3: Child Node Count Object Body Format

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