Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2017

O. Bergmann
C. Bormann
S. Gerdes
Universitaet Bremen TZI
H. Chen
Huawei Technologies
October 05, 2016

Constrained-Cast: Source-Routed Multicast for RPL
draft-bergmann-bier-ccast-02

Abstract

This specification defines a protocol for forwarding multicast traffic in a constrained node network employing the RPL routing protocol in non-storing mode.

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 April 8, 2017.

Copyright Notice

Table of Contents

1. Introduction

As defined in [RFC6550], RPL Multicast assumes that the RPL network operates in Storing Mode. Multicast DAOs are used to indicate subscription to multicast address to a parent; these DAOs percolate up and create bread-crumbs. This specification, although part of RFC 6550, appears to be incomplete and untested. More importantly, Storing Mode is not in use in constrained node networks outside research operating environments.

The present specification addresses multicast forwarding for RPL networks in the much more common Non-Storing Mode. Non-Storing is based on the root node adding source-routing information to downward packets. Evidently, to make this work, RPL multicast needs to source-route multicast packets. A source route here is a list of identifiers to instruct forwarders to relay the respective IP datagram.

As every forwarder in an IP-based constrained node network has at least one network interface, it is straight-forward to use the address of an outgoing interface as identifiers in this source-route. (Typically, this is a globally unique public address of the node's only network adapter.)

The source-route subsets the whole set of potential forwarders available in the RPL DODAG to those that need to forward in order to reach known multicast listeners.

Including an actual list of outgoing interfaces is rarely applicable, as this is likely to be a large list of 16-byte IPv6 addresses. Even with [RFC6554] style compression, the size of the list becomes prohibitively quickly.

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].

In this specification, the term "byte" is used in its now customary sense as a synonym for "octet".

All multi-byte integers in this protocol are interpreted in network byte order.

2. The BIER Approach

Bit-Indexed Explicit Replication [I-D.ietf-bier-architecture] lists all egress routers in a bitmap included in each multicast packet. This requires creating a mostly contiguous numbering of all egress routers; more importantly, BIER requires the presence of a network map in each forwarders to be able to interpret the bitmap and map it to a set of local outgoing interfaces.

3. The Constrained-Cast Approach

Constrained-Cast employs Bloom Filters [BLOOM] as a compact representation of a match or non-match for elements in a large set: Each element to be included is hashed with multiple hash functions; the result is used to index a bitmap and set the corresponding bit. To check for the presence of an element, the same hash functions are applied to obtain bit positions; if all corresponding bits are set, this is used to indicate a match. (Multiple hash functions are most easily obtained by adding a varying seed value to a single hash algorithm.)

By including a bloom filter in each packet that matches all outgoing interfaces that need to forward the packet, each forwarder can efficiently decide whether (and on which interfaces) to forward the packet.

4. False Positives

Bloom filters are probabilistic. A false positive might be indicating a match where the bits are set by aliasing of the hash values. In case of Constrained-Cast, this causes spurious transmission and wastes some energy and radio bandwidth. However, there is no semantic damage (hosts still filter out unneeded multicasts). The total waste in energy and spectrum can be visualized as the false-positive-rate multiplied by the density of the RPL network. A network can easily live with a significant percentage of false positives. By changing the set of hash functions (i.e., seed) over time, the root can avoid a single node with a false positive to become an unnecessary hotspot for that multicast group.

5. Protocol

The protocol uses DAO-like "MLAO" messages to announce membership to the root as specified in Section 5.1.

For downward messages, the root adds a new routing header that includes a hash function identifier and a seed value; another one of its fields gives the number of hash functions (k) to ask for k instances of application of the hash function, with increasing seed. The format of the new routing header is specified in Section 5.2.

Typical sizes of the bloom filter bitmap that the root inserts into the packet can be 64, 128, or 256 bit, which may lead to acceptable false positive rates if the total number of forwarders in the 10s and 100s. (To do: write more about the math here. Note that this number tallies forwarding routers, not end hosts.)

A potential forwarder that receives a multicast packet adorned with a constrained-cast routing header first checks that the packet is marked with a RPL rank smaller than its own (loop prevention). If yes, it then forwards the packet to all outgoing interfaces that match the bloom filter in the packet.

5.1. Multicast Listener Advertisement Object (MLAO)

The header format of the MLAO is depicted in Figure 1. The basic structure of the MLAO message is similar to the RPL Destination Advertisement Object (DAO). In particular, it starts with RPL ICMP base header with a type value of 155 and the code {IANA TBD1} (MLAO), followed by the Checksum, RPLInstanceID, parameters and flags as in a DAO.

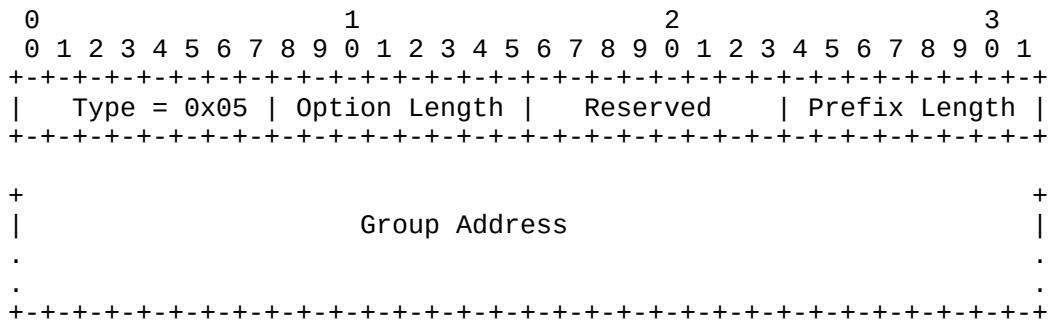


Figure 1: RPL Target Option for MLAO

The group address field indicates the group that the sender of the MLAO is interested in. This field usually contains a 128 bit IPv6 multicast group address. Shorter group identifiers could be used together with a protocol for explicit creation of groups. The MLAO message must have at least one RPL target option to specify the address of the listener that has generated the MLAO. The message is directed to the global unicast address of the DODAG root and travels upwards the routing tree.

Note: It has been suggested to use the RPL Transit Option (Type 0x06) instead as it is used in Non-Storing mode to inform the DODAG root of path attributes. Specifically, this option can be used to limit the subscription by providing a proper Path Lifetime.

5.2. Routing Header

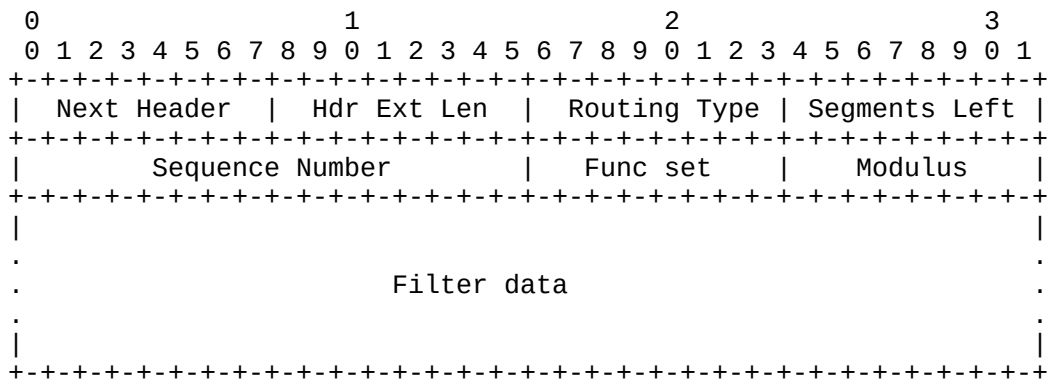


Figure 2: Routing header

Routing Type: {IANA TBD2} 253

Segments Left: This value is always 0, so network nodes that do not support this routing header do not generate ICMP6 error messages.

Sequence Number: 16 bits sequence number. The number space is unique for a sequence of multicast datagrams for a specific group that arrive at the DAG root on their way up. The DAG root increments the number for each datagram it sends down the respective DODAG.

Func set: The set of hash functions used to generate the Filter data value.

Note: As the function set contains a combination of several distinct hash functions, it is currently unclear if 8 bits number space is large enough.

Modulus: The modulus that is used by the hash functions, minus 64 (the minimum filter data size that can be used). The DAG root chooses the modulus (and thus the filter data size) to achieve its objectives for false positive rates (Section 4).

Filter data: A bit field that indicates which nodes should relay this multicast datagram. The length of this field is a multiple of 8 bytes. The actual length is derived from the contents of the field Header Ext Length.

Note: The modulus could be derived from the length of the filter data which is known from the extension header size. On the other hand, keeping a separate record of the modulus means that the DAG root could leave out 8-byte multiples of trailing zero bits if they happen to occur. But then, a modulus that leaves 8-byte sequences of zero bits in the filter is probably too large.

6. Implementation

In 2013, Constrained-Cast was implemented in Contiki. It turns out that forwarders can compute the hash functions once for their outgoing interfaces and then cache them, simply bit-matching their outgoing interface hash bits against the bloom filter in the packet (a match is indicated when all bits in the outgoing interface hash are set in the bloom filter).

The Root computes the tree for each multicast group, computes the bloom filter for it, caches these values, and then simply adds the bloom filter routing header to each downward packet. For adding a new member, the relevant outgoing interfaces are simply added to the bloom filter. For removing a leaving member, however, the bloom

filter needs to be recomputed (which can be sped up logarithmically if desired).

7. Benefits

Constrained-Cast:

- operates in Non-Storing Mode, with the simple addition of a membership information service;
- performs all routing decisions at the root.

Further optimizations might include using a similar kind of bloom filter routing header for unicast forwarding as well (representing, instead of the outgoing interface list, a list of children that forwarding parents need to forward to).

8. IANA Considerations

The following registrations are done following the procedure specified in [RFC6838].

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and "IANA TBD1" with the code selected for TBD1 below and "IANA TBD2" with the value selected for TBD2 below.

8.1. ICMPv6 Parameter Registration

IANA is requested to add the following entry to the Code fields of the RPL Control Codes registry:

Code	Name	Reference
TBD1	MLAO	[RFC-XXXX]

8.2. IPv6 Routing Type Registration

IANA is requested to add the following entries to the IPv6 Routing Types registry:

Value	Name	Reference
TBD2	CCast Routing Header	[RFC-XXXX]

9. Acknowledgments

Thanks to Yasuyuki Tanaka for valuable comments.

This work has been supported by Siemens Corporate Technology.

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, <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>.

10.2. Informative References

[BLOOM] Bloom, B., "Space/time trade-offs in hash coding with allowable errors", ISSN 0001-0782, ACM Press Communications of the ACM vol 13 no 7 pp 422-426, 1970, <http://doi.acm.org/10.1145/362686.362692>.

[I-D.ietf-bier-architecture] Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-04 (work in progress), July 2016.

[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>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

Authors' Addresses

Olaf Bergmann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63904
Email: bergmann@tzi.org

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Stefanie Gerdes
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Hao Chen
Huawei Technologies
12, E. Mozhou Rd
Nanjing 211111
China

Phone: +86-25-5662-7052
Email: philips.chenhao@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

C. Gundogan, Ed.
FU Berlin
D. Barthel
Orange
E. Baccelli
INRIA

July 8, 2016

DIS Modifications
draft-gundogan-roll-dis-modifications-00

Abstract

This document augments [RFC6550] with DIS flags and options that allow a RPL node to better control how neighbor RPL routers respond to its solicitation for DIOs.

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 January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document augments [RFC6550], the RPL routing protocol specification.

1.1. RFC6550 refresher

Per [RFC6550], a RPL node can send a DODAG Information Solicitation (DIS) message to solicit DODAG Information Object (DIO) messages from neighbor RPL routers.

A DIS can be multicast to all the routers in range or it can be unicast to a specific neighbor router.

A DIS may carry a Solicited Information option that specifies the predicates of the DAG(s) the soliciting node is interested in. In the absence of such Solicited Information option, the soliciting node

is deemed interested in receiving DIOs for all the DAGs known by the solicited router(s).

[RFC6550] requires a router to treat the receipt of a multicast DIS as an inconsistency and hence reset its Trickle timers for the matching DAGs. As a result of the general Trickle timer mechanism, future DIOs will be sent at a higher rate. See [RFC6206] for the specification of Trickle timers and the definition of "inconsistency".

[RFC6550] requires a router that receives a unicast DIS to respond by unicasting a DIO for each matching DAG and to not reset the associated Trickle timer. Such a DIO generated in response to a unicast DIS must contain a Configuration option.

This description is summarized in Table 1.

	Unicast DIS	Multicast DIS
no option present	unicast DIO, don't reset Trickle timer	do reset Trickle timer
Solicited Information option present, not matching	do nothing	do nothing
Solicited Information option present, matching	unicast DIO, don't reset Trickle timer	do reset Trickle timer

Table 1: Router behavior on receiving a DIS, as per RFC6550

More precisely, Table 1 describes the behavior of routers for each DAG they belong to. In the general case where multiple RPL instances co-exist in a network, routers will maintain a Trickle timer for the one DAG of each RPL instance they belong to, and nodes may send a DIS with multiple Solicited Information options pertaining to different DAGs or instances. In this more general case, routers will respond for each individual DAG/instance they belong to as per Table 1.

1.2. Undesirable effects

Now, consider a RPL leaf node that desires to join a certain DAG. This node can either wait for its neighbor RPL routers to voluntarily transmit DIOs or it can proactively solicit DIOs using a DIS message. Voluntary DIO transmissions may happen after a very long time if the network is stable and the Trickle timer intervals have reached large

values. Thus, proactively seeking DIOs using a DIS may be the only reasonable option. Since the node does not know which neighbor routers belong to the DAG, it must solicit the DIOs using a multicast DIS (with predicates of the desired DAG specified inside a Solicited Information option). On receiving this DIS, the neighbor routers that belong to the desired DAG will reset their Trickle timers and quickly transmit their DIOs. The downside of resetting Trickle timers is that the routers will keep transmitting frequent DIOs for a considerable duration until the Trickle timers again reach long intervals. These DIO transmissions are unnecessary, consume precious energy and may contribute to congestion in the network.

There are other scenarios where resetting of Trickle timer following the receipt of a multicast DIS is not appropriate. For example, consider a RPL router that desires to free up memory by deleting state for the defunct DAGs it belongs to. Identifying a defunct DAG may require the node to solicit DIOs from its DAG parents using a multicast DIS.

Certain scenarios may require a RPL router to solicit a DIO from a parent by using a unicast DIS. The parent is forced to include a Configuration option within the unicast DIO, although the requesting node might still have this information locally available. Since the information within the Configuration option is described as generally static and unchanging throughout the DODAG, it inflates the unicast DIO unnecessarily by 16 bytes for each request.

1.3. Desired improvments

To deal with the situations described above, there is a need in the industry for DIS flags and options that allow a RPL node to control how neighbor RPL routers respond to its solicitation for DIOs, for example by expressing:

- the routing constraints that routers should meet to be allowed to respond, thereby lowering the number of responders
- whether the responding routers should reset their Trickle timers or not, thereby limiting the cumulated number of transmitted DIOs
- whether the responding routers should respond with a unicast DIO instead of a multicast one, thereby lowering the overhearing cost in the network
- whether the responding routers should omit DIO options that were not requested explicitely and thus reducing the amount of traffic and giving full control over the options of the solicited DIO

- the time interval over which the responding routers should schedule their DIO transmissions, thereby lowering the occurence of collisions.

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 terminology from [RFC6550]. Specifically, the term RPL node refers to a RPL router or a RPL host as defined in [RFC6550].

3. DIS Base Object flags

This document defines three new flags inside the DIS base object:

- the "No Inconsistency" (N) flag: On receiving a multicast DIS with the N flag set, a RPL router MUST NOT reset the Trickle timers for the matching DAGs. In addition, it MUST take specific action, which is to respond by explicitely sending a DIO. This DIO MUST include a Configuration option. This behavior augments [RFC6550], which had provision for such flag. Since this specific, one-shot DIO is not a consequence of the general Trickle timer mechanism, it will be sent right away if no Response Spreading option is present or it will be scheduled according to the Response Spreading option if one is present in the DIS (see Section 4.2).

- the "DIO Type" (T) flag: In case the N flag is set, this T flag specifies what type of DIO is sent in response. It MUST be a unicast DIO if this flag is set and it MUST be a multicast DIO if this flag is reset.

- the "DIO Option Request" (R) flag: On receiving a DIS with the R flag set, the receiver MUST include all options that were requested by the DIS containing one or multiple DIO Option Request options. A responding RPL router MUST NOT include DIO options that were not explicitely requested. Note that this behaviour contradicts with [RFC6550] for the case of including a Configuration option in all DIOs requested by a unicast DIS.

When a unicast DIS is transmitted, both its N and T flags SHOULD be 0, which are the default values per [RFC6550]. On receiving a unicast DIS, the N and T flags MUST be ignored and treated as 00. When the R flag is unset, then a RPL router may include or omit DIO options like specified in [RFC6550]. A rpl router responding to a

DIS with the R flag set MUST only include all requested DIO options in the solicited DIO.

The modified DIS base object is shown in Figure 1.

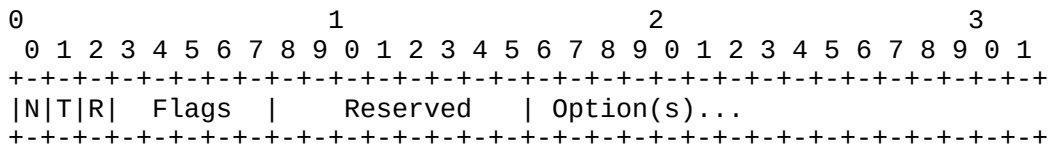


Figure 1: Modified DIS Base Object

4. DIS Options

4.1. Metric Container

In order to lower the number of routers that will respond to a DIS, this document allows routing constraints to be carried by a DIS. Only the router(s) that satisfy these constraints is (are) allowed to respond to the DIS.

These routing constraints are described using a Metric Container option contained in the DIS. Metric Containers are defined in [RFC6550] and [RFC6551]. Metric Containers options were previously only allowed in DIOs. This document augments [RFC6550] by allowing the inclusion of a Metric Container option inside a DIS as well.

A RPL router that receives a DIS with a Metric Container option MUST ignore any Metric object in it, and MUST evaluate the "mandatory" Constraint objects in it by comparing the constraint value to the value of the corresponding routing metric that the router maintains for the matching DAG(s). These routing metric values MUST satisfy all the mandatory constraints in order for the router to consider the solicitation successful for the matching DAG(s). This augments the behavior already present in [RFC6550] with the Solicited Information option.

This option can be used in both unicast and multicast DIS.

4.2. Response Spreading

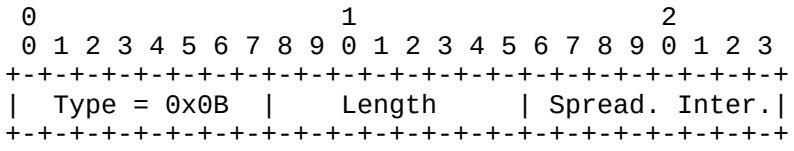


Figure 2: The Response Spreading option

Even with the use of the Solicited Information and the Section 4.1 options, a multicast DIS may still lead to a large number of RPL routers taking immediate action and responding with DIOs. Concurrent transmissions by multiple routers are not desirable since they may lead to poor channel utilization or even to packet loss. Unicast DIOs may be able to avail of link-level retransmissions. However, multicast DIOs usually have no such protection, since they commonly make use of link layer broadcast. To avoid such problems, this document specifies an optional DIO response spreading mechanism.

This document defines a new RPL control message option called Response Spreading option, shown in Figure 2, with a recommended Type value 0x0B (to be confirmed by IANA). A RPL router that explicitely responds with a specific, one-shot DIO to a DIS that includes a Response Spreading option, MUST wait for a time uniformly chosen in the interval [O..2^SpreadingInterval], expressed in ms, before attempting to transmit its DIO. If the DIS does not include a Response Spreading option, the node is free to transmit the DIO as it otherwise would.

A Response Spreading option MAY be included inside a unicast DIS message, but there is no benefit in doing so.

Multiple Response Spreading options SHOULD NOT be used inside a same DIS message.

This mechanism MUST NOT affect the Trickle timer mechanism.

4.3. DIO Option Request

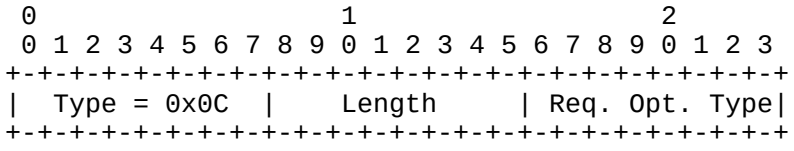


Figure 3: The DIO Option Request option

If a unicast DIS is used to request a DIO, then [RFC6550] mandates that a Configuration option MUST be included in this DIO. The

Configuration option contains generally static information that stays unmodified throughout the DAG. For scenarios where a RPL node is already part of a DAG and hence is holding the information that is propagated with the Configuration option, an inclusion of such leads to an unnecesary inflation of 16 bytes for each solicited DIO.

As per [RFC6550], no process is defined to trigger the inclusion of other DIO options in a solicited DIO.

This document defines a new RPL control message option called DIO Option Request option, shown in Figure 3, with a recommended Type value of 0x0C (to be confirmed by IANA). This new option allows full control over the options of the solicited DIO. The target of a unicast or multicast DIS with the R flag set and with one or more DIO Option Request options included, MUST include these requested options in the solicited DIO. For a DIS with the R flag unset, a RPL router behaves like described in [RFC6550] with regard to DIO options.

5. Full behavior illustration

Figure 4 and Figure 5 illustrate the normative behavior described in Section 3 and Section 4.1.

	Unicast DIS	Multicast DIS	
		N=0	
no option present	unicast DIO, don't reset Trickle timer	do reset Trickle timer	
Solicited Information/Metric Container option present, not matching.	do nothing	do nothing	
Solicited Information/Metric Container option present, matching.	unicast DIO, don't reset Trickle timer	do reset Trickle timer	

Figure 4: Overall DIS behavior, part 1

Notice that Figure 4 is indeed identical to Table 1 when Metric Container options are not used in DIS.

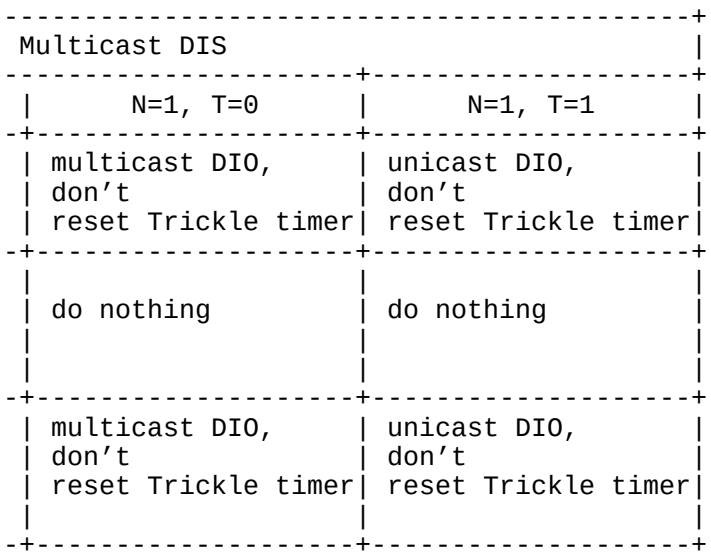


Figure 5: Overall DIS behavior, part 2

For the sake of completeness, let's remind here that a specific, one-shot DIO generated in response to a DIS with the R flag unset MUST contain a Configuration option. If the R flag is set, then this DIO contains only explicitely requested DIO options. This DIO's transmission is delayed according to the Delay Spreading option of the DIS, if one such option is present.

6. IANA Considerations

6.1. DIS Flags

IANA is requested to allocate bits 0, 1 and 2 of the DIS Flag Field to become the "No Inconsistency", "DIO Type", and "DIO Option Request" bits, the functionality of which is described in Section 3 of this document.

Value	Meaning	Reference
0	No Inconsistency	This document
1	DIO Type	This document
2	DIO Option Request	This document

6.2. RPL Control Message Options

IANA is requested to allocate a new code point in the "RPL Control Message Options" registry for the "Response Spreading" option and the "Dio Option Request" option, the behavior of which are described in Section 4.2 and Section 4.3, respectively.

Value	Meaning	Reference
0x0B	Response Spreading	This document
0x0C	DIO Option Request	This document

RPL Control Message Options

7. Security Considerations

TBA

8. Acknowledgements

A lot of text in this document originates from now-expired [I-D.goyal-roll-dis-modifications] co-authored with M. Goyal. The requirements and solutions also draw from now-expired [I-D.dejean-roll-selective-dis] co-authored with N. Dejean. Their contribution is deeply acknowledged.

We also thank (TBA) for their useful feedback and discussion.

9. References

9.1. Normative References

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

9.2. Informative References

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5184] Teraoka, F., Gogo, K., Mitsuya, K., Shibui, R., and K. Mitani, "Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover", RFC 5184, DOI 10.17487/RFC5184, May 2008, <http://www.rfc-editor.org/info/rfc5184>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <http://www.rfc-editor.org/info/rfc5881>.

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

Appendix A. Applications

This section details two example mechanisms that use the DIS flags and options defined in this document. The first mechanism describes how a leaf node may join a desired DAG in an energy efficient manner. The second mechanism details how a node may identify defunct DAGs for which it still maintains state.

A.1. A Leaf Node Joining a DAG

A new leaf node that joins an established LLN runs an iterative algorithm in which it requests (using multicast DIS) DIOs from routers belonging to the desired DAG.

The DIS message has the "No Inconsistency" flag set to prevent resetting of Trickle timer in responding routers, thereby keeping the aggregated number of transmissions low. It also has the "DIO Type" flag set to make responding routers send unicast DIOs back, thereby not triggering full reception in nearby nodes that have state-of-the-art radio receivers with hardware-based address filtering.

The DIS message can include a Response Spreading option prescribing a suitable spreading interval based on the expected density of nearby routers and on the expected Layer 2 technology.

The DIS will likely include a Metric Container listing the routing constraints that the responding routers must satisfy in order to be allowed to respond.

At each iteration, the node multicasts such a DIS and waits for forthcoming DIOs. After a time equal to the spreading interval, the node considers the current iteration to be unsuccessful. The node consequently relaxes the routing constraints somewhat and proceeds to the next iteration.

The cycle repeats until the node receives one or more DIOs or until it has relaxed the constraints to the lowest acceptable values.

This algorithm has been proven in the field to be extremely energy-efficient, especially when routers have a wide communication range.

A.2. Identifying A Defunct DAG

A RPL node may remove a neighbor from its parent set for a DAG for a number of reasons:

- The neighbor is no longer reachable, as determined using a mechanism such as Neighbor Unreachanility Detection (NUD) [RFC4861], Bidirectional Forwarding Detection (BFD) [RFC5881] or L2 triggers [RFC5184]; or

- The neighbor advertises an infinite rank in the DAG; or

- Keeping the neighbor as a parent would required the node to increase its rank beyond L + DAGMaxRankIncrease, where L is the minimum rank the node has had in this DAG; or

- The neighbor advertises membership in a different DAG within the same RPL Instance, where a different DAG is recognised by a different DODAGID or a different DODAGVersionNumber.

Even if the conditions listed above exist, a RPL node may fail to remove a neighbor from its parent set because:

- The node may fail to receive the neighbor’s DIOs advertising an increased rank or the neighbor’s membership in a different DAG;

- The node may not check, and hence may not detect, the neighbor’s unreachability for a long time. For example, the node may not

have any data to send to this neighbor and hence may not encounter any event (such as failure to send data to this neighbor) that would trigger a check for the neighbor's reachability.

In such cases, a node would continue to consider itself attached to a DAG even if all its parents in the DAG are unreachable or have moved to different DAGs. Such a DAG can be characterized as being defunct from the node's perspective. If the node maintains state about a large number of defunct DAGs, such state may prevent a considerable portion of the total memory in the node from being available for more useful purposes.

To alleviate the problem described above, a RPL node may invoke the following procedure to identify a defunct DAG and delete the state it maintains for this DAG. Note that, given the proactive nature of RPL protocol, the lack of data traffic using a DAG can not be considered a reliable indication of the DAG's defunction. Further, the Trickle timer based control of DIO transmissions means the possibility of an indefinite delay in the receipt of a new DIO from a functional DAG parent. Hence, the mechanism described here is based on the use of a DIS message to solicit DIOs about a DAG suspected of defunction. Further, a multicast DIS is used so as to avoid the need to query each parent individually and also to discover other neighbor routers that may serve as the node's new parents in the DAG.

When a RPL node has not received a DIO from any of its parents in a DAG for more than a locally configured time duration:

- The node generates a multicast DIS message with:
  - the "No Inconsistency" flag set so that the responding routers do not reset their Trickle timers.
  - the "DIO Type" flag not set so that the responding routers send multicast DIOs and other nodes in the vicinity do not need to invoke this procedure.
  - a Solicited Information option to identify the DAG in question. This option must have the I and D flags set and the RPLInstanceID/DODAGID fields must be set to values identifying the DAG. The V flag inside the Solicited Information option should not be set so as to allow the neighbors to send DIOs advertising the latest version of the DAG.
  - a Response Spreading option specifying a suitable time interval over which the DIO responses may arrive.

- After sending the DIS, the node waits for the duration specified inside the Response Spreading option to receive the DIOs generated by its neighbors. At the conclusion of the wait duration:
  - If the node has received one or more DIOs advertising newer version(s) of the DAG, it joins the latest version of the DAG, selects a new parent set among the neighbors advertising the latest DAG version and marks the DAG status as functional.
  - Otherwise, if the node has not received a DIO advertising the current version of the DAG from a neighbor in the parent set, it removes that neighbor from the parent set. As a result, if the node has no parent left in the DAG, it marks the DAG as defunct and schedule the deletion of the state it has maintained for the DAG after a locally configured "hold" duration. (This is because, as per RPL specification, when a node no longer has any parents left in a DAG, it is still required to remember the DAG's identity (RPLInstanceID, DODAGID, DODAGVersionNumber), the lowest rank (L) it has had in this DAG and the DAGMaxRankIncrease value for the DAG for a certain time interval to ensure that the node does not join an earlier version of the DAG and does not rejoin the current version of the DAG at a rank higher than L + DAGMaxRankIncrease.)

A.3. Explicit and Implicit DIO Option Requests

Certain information from a DIO is only needed occasionally or for specific events:

- A Configuration option contains information that is static and stays mostly unchanged during the lifetime of the DODAG. Thus, the Configuration option is important when joining the DODAG, but inflates the DIO unnecessarily thereafter.
- A Prefix Information option is also useful when joining a DODAG to perform address autoconfiguration and propagating the prefixes. After that, it is only relevant when changes occure that would affect the prefix information (new prefixes, lifetime updates, ...).
- More DIO options may be added in the future that have similar properties as mentioned above.

The Configuration and Prefix Information options may be omitted from the trickle timer based DIOs, leading to less bytes for each DIO. Once a RPL router decides to join a DODAG, it may solicit a DIO by sending a DIS with the R flag set and two "DIO Option Request"

options included. One for the Configuration option, the other for the Prefix Information option. Upon receiving these options in the next DIO, the RPL router can successfully finish the joining process.

Appendix B. Experimental data

The effectiveness of these flags and options has been measured on real industrial hardware.

Data to be added

Authors' Addresses

Cenk Gundogan (editor)
FU Berlin

Email: cenk.guendogan@fu-berlin.de

Dominique Barthel
Orange
28 Chemin Du Vieux Chene, BP 98
Meylan 38243
France

Email: dominique.barthel@orange.com

Emmanuel Baccelli
INRIA

Phone: +33-169-335-511
Email: Emmanuel.Baccelli@inria.fr
URI: http://www.emmanuelbaccelli.org/

ROLL Working Group
Internet-Draft
Updates: 6553, 6550, 8138 (if approved)
Intended status: Standards Track
Expires: July 19, 2021

M. Robles
UTN-FRM/Aalto
M. Richardson
SSW
P. Thubert
Cisco

January 15, 2021

Using RPI Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane
draft-ietf-roll-useofrplinfo-44

Abstract

This document looks at different data flows through LLN (Low-Power and Lossy Networks) where RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) is used to establish routing. The document enumerates the cases where RFC6553 (RPI Option Type), RFC6554 (Routing Header for Source Routes) and IPv6-in-IPv6 encapsulation is required in data plane. This analysis provides the basis on which to design efficient compression of these headers. This document updates RFC6553 adding a change to the RPI Option Type. Additionally, this document updates RFC6550 defining a flag in the DIO Configuration option to indicate about this change and updates RFC8138 as well to consider the new Option Type when the RPL Option is decompressed.

Table of Contents

1. Introduction

RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [RFC6550] is a routing protocol for constrained networks. [RFC6553] defines the RPL Option carried within the IPv6 Hop-by-Hop Header to carry the RPLInstanceID and quickly identify inconsistencies (loops) in the routing topology. The RPL Option is commonly referred to as the RPL Packet Information (RPI) though the RPI is the routing information that is defined in [RFC6550] and transported in the RPL Option. RFC6554 [RFC6554] defines the "RPL Source Route Header" (RH3), an IPv6 Extension Header to deliver datagrams within a RPL routing domain, particularly in non-storing mode.

These various items are referred to as RPL artifacts, and they are seen on all of the data-plane traffic that occurs in RPL routed networks; they do not in general appear on the RPL control plane

traffic at all which is mostly Hop-by-Hop traffic (one exception being DAO messages in non-storing mode).

It has become clear from attempts to do multi-vendor interoperability, and from a desire to compress as many of the above artifacts as possible that not all implementers agree when artifacts are necessary, or when they can be safely omitted, or removed.

The ROLL WG analyzed how [RFC2460] rules apply to storing and non-storing use of RPL. The result was 24 data plane use cases. They are exhaustively outlined here in order to be completely unambiguous. During the processing of this document, new rules were published as [RFC8200], and this document was updated to reflect the normative changes in that document.

This document updates [RFC6553], changing the value of the Option Type of the RPL Option to make [RFC8200] routers ignore this option when not recognized.

A Routing Header Dispatch for 6LoWPAN (6LoRH)([RFC8138]) defines a mechanism for compressing RPL Option information and Routing Header type 3 (RH3) [RFC6554], as well as an efficient IPv6-in-IPv6 technique.

Most of the use cases described herein require the use of IPv6-in-IPv6 packet encapsulation. When encapsulating and decapsulating packets, [RFC6040] MUST be applied to map the setting of the explicit congestion notification (ECN) field between inner and outer headers. Additionally, [I-D.ietf-intarea-tunnels] is recommended reading to explain the relationship of IP tunnels to existing protocol layers and the challenges in supporting IP tunneling.

Non-constrained uses of RPL are not in scope of this document, and applicability statements for those uses may provide different advice, E.g. [I-D.ietf-anima-autonomic-control-plane].

1.1. Overview

The rest of the document is organized as follows: Section 2 describes the used terminology. Section 3 provides a RPL Overview. Section 4 describes the updates to RFC6553, RFC6550 and RFC 8138. Section 5 provides the reference topology used for the uses cases. Section 6 describes the use cases included. Section 7 describes the storing mode cases and section 8 the non-storing mode cases. Section 9 describes the operational considerations of supporting RPL-unaware-leaves. Section 10 depicts operational considerations for the proposed change on RPI Option Type, section 11 the IANA considerations and then section 12 describes the security aspects.

2. Terminology and Requirements Language

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

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

Consumed: A Routing Header is consumed when the Segments Left field is zero, which indicates that the destination in the IPv6 header is the final destination of the packet and that the hops in the Routing Header have been traversed.

RPL Leaf: An IPv6 host that is attached to a RPL router and obtains connectivity through a RPL Destination Oriented Directed Acyclic Graph (DODAG). As an IPv6 node, a RPL Leaf is expected to ignore a consumed Routing Header and as an IPv6 host, it is expected to ignore a Hop-by-Hop header. It results that a RPL Leaf can correctly receive a packet with RPL artifacts. On the other hand, a RPL Leaf is not expected to generate RPL artifacts or to support IP-in-IP encapsulation. For simplification, this document uses the standalone term leaf to mean a RPL leaf.

RPL Packet Information (RPI): The information defined abstractly in [RFC6550] to be placed in IP packets. The term is commonly used, including in this document, to refer to the RPL Option [RFC6553] that transports that abstract information in an IPv6 Hop-by-Hop Header. [RFC8138] provides an alternate (more compressed) formating for the same abstract information.

RPL-aware-node (RAN): A device which implements RPL. Please note that the device can be found inside the LLN or outside LLN.

RPL-Aware-Leaf(RAL): A RPL-aware-node that is also a RPL Leaf.

RPL-unaware-node: A device which does not implement RPL, thus the device is not-RPL-aware. Please note that the device can be found inside the LLN.

RPL-Unaware-Leaf(RUL): A RPL-unaware-node that is also a RPL Leaf.

6LoWPAN Node (6LN): [RFC6775] defines it as: "A 6LoWPAN node is any host or router participating in a LoWPAN. This term is used when referring to situations in which either a host or router can play the role described.". In this document, a 6LN acts as a leaf.

6LoWPAN Router (6LR): [RFC6775] defines it as:" An intermediate router in the LoWPAN that is able to send and receive Router Advertisements (RAs) and Router Solicitations (RSs) as well as forward and route IPv6 packets. 6LoWPAN routers are present only in route-over topologies."

6LoWPAN Border Router (6LBR): [RFC6775] defines it as:"A border router located at the junction of separate 6LoWPAN networks or between a 6LoWPAN network and another IP network. There may be one or more 6LBRs at the 6LoWPAN network boundary. A 6LBR is the responsible authority for IPv6 prefix propagation for the 6LoWPAN network it is serving. An isolated LoWPAN also contains a 6LBR in the network, which provides the prefix(es) for the isolated network."

Flag Day: A Flag Day is caused when a network is reconfigured in a way that nodes running the older configuration can not communicate with nodes running the new configuration. For instance, when the ARPANET changed from IP version 3 to IP version 4 on January 1, 1983 ([RFC0801]). In the context of this document, a switch from RPI Option Type (0x63) and Option Type (0x23) presents as a disruptive changeover. In order to reduce the amount of time for such a changeover, Section 4.1.3 provides a mechanism to allow nodes to be incrementally upgraded.

Non-Storing Mode (Non-SM): RPL mode of operation in which the RPL-aware-nodes send information to the root about their parents. Thus, the root knows the topology. Because the root knows the topology, the intermediate 6LRs do not maintain routing state and source routing is needed.

Storing Mode (SM): RPL mode of operation in which RPL-aware-nodes (6LRs) maintain routing state (of the children) so that source routing is not needed.

Note: Due to lack of space in some figures (tables) we refer to IPv6-in-IPv6 as IP6-IP6.

3. RPL Overview

RPL defines the RPL Control messages (control plane), a new ICMPv6 [RFC4443] message with Type 155. DIS (DODAG Information Solicitation), DIO (DODAG Information Object) and DAO (Destination Advertisement Object) messages are all RPL Control messages but with different Code values. A RPL Stack is shown in Figure 1.

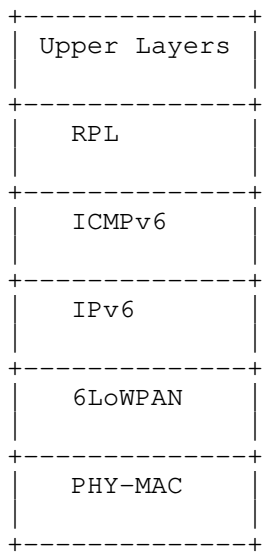


Figure 1: RPL Stack.

RPL supports two modes of Downward internal traffic: in storing mode (SM), it is fully stateful; in non-storing mode (Non-SM), it is fully source routed. A RPL Instance is either fully storing or fully non-storing, i.e. a RPL Instance with a combination of a fully storing and non-storing nodes is not supported with the current specifications at the time of writing this document. External routes are advertised with non-storing-mode messaging even in a storing mode network, see Section 4.1.1

4. Updates to RFC6550, RFC6553 and RFC8138

4.1. Updates to RFC6550

4.1.1. Advertising External Routes with Non-Storing Mode Signaling.

Section 6.7.8. of [RFC6550] introduces the 'E' flag that is set to indicate that the 6LR that generates the DAO redistributes external targets into the RPL network. An external Target is a Target that has been learned through an alternate protocol, for instance a route to a prefix that is outside the RPL domain but reachable via a 6LR. Being outside of the RPL domain, a node that is reached via an external target cannot be guaranteed to ignore the RPL artifacts and cannot be expected to process the [RFC8138] compression correctly. This means that the RPL artifacts should be contained in an IP-in-IP encapsulation that is removed by the 6LR, and that any remaining

compression should be expanded by the 6LR before it forwards a packet outside the RPL domain.

This specification updates [RFC6550] to RECOMMEND that external targets are advertised using Non-Storing Mode DAO messaging even in a Storing-Mode network. This way, external routes are not advertised within the DODAG and all packets to an external target reach the Root like normal Non-Storing Mode traffic. The Non-Storing Mode DAO informs the Root of the address of the 6LR that injects the external route, and the root uses IP-in-IP encapsulation to that 6LR, which terminates the IP-in-IP tunnel and forwards the original packet outside the RPL domain free of RPL artifacts.

In the other direction, for traffic coming from an external target into the LLN, the parent (6LR) that injects the traffic always encapsulates to the root. This whole operation is transparent to intermediate routers that only see traffic between the 6LR and the Root, and only the Root and the 6LRs that inject external routes in the network need to be upgraded to add this function to the network.

A RUL is a special case of external target when the target is actually a host and it is known to support a consumed Routing Header and to ignore a Hop-by-Hop header as prescribed by [RFC8200]. The target may have been learned through an external routing protocol or may have been registered to the 6LR using [RFC8505].

In order to enable IP-in-IP all the way to a 6LN, it is beneficial that the 6LN supports decapsulating IP-in-IP, but that is not assumed by [RFC8504]. If the 6LN is a RUL, the Root that encapsulates a packet SHOULD terminate the tunnel at a parent 6LR unless it is aware that the RUL supports IP-in-IP decapsulation.

A node that is reachable over an external route is not expected to support [RFC8138]. Whether a decapsulation took place or not and even when the 6LR is delivering the packet to a RUL, the 6LR that injected an external route MUST uncompress the packet before forwarding over that external route.

4.1.2. Configuration Options and Mode of Operation

Section 6.7.6 of RFC6550 describes the DODAG Configuration Option as containing a series of Flags in the first octet of the payload.

Anticipating future work to revise RPL relating to how the LLN and DODAG are configured, this document renames the DODAG Configuration Option Flags registry so that it applies to Mode of Operation (MOP) values zero (0) to six (6) only, leaving the flags unassigned for MOP value seven (7).The MOP is described in RFC6550 section 6.3.1.

In addition, this document reserves MOP value 7 for future expansion.

See Sections 11.2 and 11.3.

4.1.3. Indicating the new RPI in the DODAG Configuration option Flag.

In order to avoid a Flag Day caused by lack of interoperation between new RPI Option Type (0x23) and old RPI Option Type (0x63) nodes, this section defines a flag in the DIO Configuration option, to indicate when the new RPI Option Type can be safely used. This means, the flag is going to indicate the value of Option Type that the network will be using for the RPL Option. Thus, when a node joins to a network it will know which value to use. With this, RPL-capable nodes know if it is safe to use 0x23 when creating a new RPL Option. A node that forwards a packet with an RPI MUST NOT modify the Option Type of the RPL Option.

This is done using a DODAG Configuration option flag which will signal "RPI 0x23 enable" and propagate through the network. Section 6.3.1. of [RFC6550] defines a 3-bit Mode of Operation (MOP) in the DIO Base Object. The flag is defined only for MOP value between 0 to 6.

For a MOP value of 7, a node MUST use the RPI 0x23 option.

As stated in [RFC6550] the DODAG Configuration option is present in DIO messages. The DODAG Configuration option distributes configuration information. It is generally static, and does not change within the DODAG. This information is configured at the DODAG root and distributed throughout the DODAG with the DODAG Configuration option. Nodes other than the DODAG root do not modify this information when propagating the DODAG Configuration option.

Currently, the DODAG Configuration option in [RFC6550] states: "the unused bits MUST be initialized to zero by the sender and MUST be ignored by the receiver". If the flag is received with a value zero (which is the default), then new nodes will remain in RFC6553 Compatible Mode; originating traffic with the old-RPI Option Type (0x63) value. If the flag is received with a value of 1, then the value for the RPL Option MUST be set to 0x23.

Bit number three of the flag field in the DODAG Configuration option is to be used as shown in Figure 2 (which is the same as Figure 39 in Section 11 and is shown here for convenience):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 2: DODAG Configuration option Flag to indicate the RPI-flag-day.

In the case of reboot, the node (6LN or 6LR) does not remember the RPI Option Type (i.e., whether or not the flag is set), so the node will not trigger DIO messages until a DIO message is received indicating the RPI value to be used. The node will use the value 0x23 if the network supports this feature.

4.2. Updates to RFC6553: Indicating the new RPI Option Type.

This modification is required in order to be able to send, for example, IPv6 packets from a RPL-Aware-Leaf to a RPL-unaware node through Internet (see Section 7.2.1), without requiring IPv6-in-IPv6 encapsulation.

[RFC6553] (Section 6, Page 7) states as shown in Figure 3, that in the Option Type field of the RPL Option, the two high order bits must be set to '01' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node must discard the packet if it doesn't recognize the Option Type, and the third bit indicates that the Option Data may change in route. The remaining bits serve as the Option Type.

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

Figure 3: Option Type in RPL Option.

This document illustrates that it is not always possible to know for sure at the source that a packet will only travel within the RPL domain or may leave it.

At the time [RFC6553] was published, leaking a Hop-by-Hop header in the outer IPv6 header chain could potentially impact core routers in the internet. So at that time, it was decided to encapsulate any

packet with a RPL Option using IPv6-in-IPv6 in all cases where it was unclear whether the packet would remain within the RPL domain. In the exception case where a packet would still leak, the Option Type would ensure that the first router in the Internet that does not recognize the option would drop the packet and protect the rest of the network.

Even with [RFC8138], where the IPv6-in-IPv6 header is compressed, this approach yields extra bytes in a packet; this means consuming more energy, more bandwidth, incurring higher chances of loss and possibly causing a fragmentation at the 6LoWPAN level. This impacts the daily operation of constrained devices for a case that generally does not happen and would not heavily impact the core anyway.

While intention was and remains that the Hop-by-Hop header with a RPL Option should be confined within the RPL domain, this specification modifies this behavior in order to reduce the dependency on IPv6-in-IPv6 and protect the constrained devices. Section 4 of [RFC8200] clarifies the behaviour of routers in the Internet as follows: "it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so".

When unclear about the travel of a packet, it becomes preferable for a source not to encapsulate, accepting the fact that the packet may leave the RPL domain on its way to its destination. In that event, the packet should reach its destination and should not be discarded by the first node that does not recognize the RPL Option. But with the current value of the Option Type, if a node in the Internet is configured to process the Hop-by-Hop header, and if such node encounters an option with the first two bits set to 01 and conforms to [RFC8200], it will drop the packet. Host systems should do the same, irrespective of the configuration.

Thus, this document updates the Option Type of the RPL Option [RFC6553], naming it RPI Option Type for simplicity, to (Figure 4): the two high order bits MUST be set to '00' and the third bit is equal to '1'. The first two bits indicate that the IPv6 node MUST skip over this option and continue processing the header ([RFC8200] Section 4.2) if it doesn't recognize the Option Type, and the third bit continues to be set to indicate that the Option Data may change en route. The rightmost five bits remain at 0x3(00011). This ensures that a packet that leaves the RPL domain of an LLN (or that leaves the LLN entirely) will not be discarded when it contains the RPL Option.

With the new Option Type, if an IPv6 (intermediate) node (RPL-not-capable) receives a packet with a RPL Option, it should ignore the

Hop-by-Hop RPL Option (skip over this option and continue processing the header). This is relevant, as it was mentioned previously, in the case that there is a flow from RAL to Internet (see Section 7.2.1).

This is a significant update to [RFC6553].

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

Figure 4: Revised Option Type in RPL Option. (*)represents this document

Without the signaling described below, this change would otherwise create a lack of interoperation (flag day) for existing networks which are currently using 0x63 as the RPI Option Type value. A move to 0x23 will not be understood by those networks. It is suggested that RPL implementations accept both 0x63 and 0x23 when processing the header.

When forwarding packets, implementations SHOULD use the same value of RPI Type as was received. This is required because the RPI Option Type does not change en route ([RFC8200] - Section 4.2). It allows the network to be incrementally upgraded and allows the DODAG root to know which parts of the network have been upgraded.

When originating new packets, implementations should have an option to determine which value to originate with, this option is controlled by the DIO Configuration option (Section Section 4.1.3).

The change of RPI Option Type from 0x63 to 0x23, makes all [RFC8200] Section 4.2 compliant nodes tolerant of the RPL artifacts. There is no longer a need to remove the artifacts when sending traffic to the Internet. This change clarifies when to use IPv6-in-IPv6 headers, and how to address them: The Hop-by-Hop Options header containing the RPI MUST always be added when 6LRs originate packets (without IPv6-in-IPv6 headers), and IPv6-in-IPv6 headers MUST always be added when a 6LR finds that it needs to insert a Hop-by-Hop Options header containing the RPL Option. The IPv6-in-IPv6 header is to be addressed to the RPL root when on the way up, and to the end-host when on the way down.

In the non-storing case, dealing with not-RPL aware leaf nodes is much easier as the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node.

The 6LBR can recognize not-RPL aware leaf nodes because it will receive a DAO about that node from the 6LR immediately above that not-RPL aware node.

The non-storing mode case does not require the type change from 0x63 to 0x23, as the root can always create the right packet. The type change does not adversely affect the non-storing case.(see Section 4.1.3)

4.3. Updates to RFC8138: Indicating the way to decompress with the new RPI Option Type.

This modification is required in order to be able to decompress the RPL Option with the new Option Type of 0x23.

RPI-6LoRH header provides a compressed form for the RPL RPI; see [RFC8138], Section 6. A node that is decompressing this header MUST decompress using the RPI Option Type that is currently active: that is, a choice between 0x23 (new) and 0x63 (old). The node will know which to use based upon the presence of the flag in the DODAG Configuration option defined in Section 4.1.3. E.g. If the network is in 0x23 mode (by DIO option), then it should be decompressed to 0x23.

[RFC8138] section 7 documents how to compress the IPv6-in-IPv6 header.

There are potential significant advantages to having a single code path that always processes IPv6-in-IPv6 headers with no conditional branches.

In Storing Mode, the scenarios where the flow goes from RAL to RUL and RUL to RUL include compression of the IPv6-in-IPv6 and RPI headers. The use of the IPv6-in-IPv6 header is MANDATORY in this case, and it SHOULD be compressed with [RFC8138] section 7. Figure 5 illustrates the case in Storing mode where the packet is received from the Internet, then the root encapsulates the packet to insert the RPI. In that example, the leaf is not known to support RFC 8138, and the packet is encapsulated to the 6LR that is the parent and last hop to the final destination.

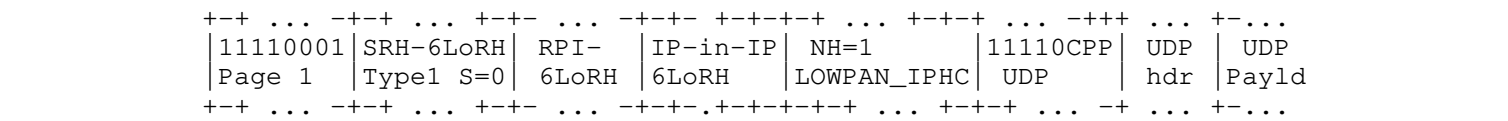


Figure 5: RPI Inserted by the Root in Storing Mode

In Figure 5, the source of the IPv6-in-IPv6 encapsulation is the Root, so it is elided in the IP-in-IP 6LoRH. The destination is the parent 6LR of the destination of the inner packet so it cannot be elided. It is placed as the single entry in an SRH-6LoRH as the first 6LoRH. There is a single entry so the SRH-6LoRH Size is 0. In that example, the type is 1 so the 6LR address is compressed to 2 bytes. It results that the total length of the SRH-6LoRH is 4 bytes. Follows the RPI-6LoRH and then the IP-in-IP 6LoRH. When the IP-in-IP 6LoRH is removed, all the router headers that precede it are also removed. The Paging Dispatch [RFC8025] may also be removed if there was no previous Page change to a Page other than 0 or 1, since the LOWPAN_IPHC is encoded in the same fashion in the default Page 0 and in Page 1. The resulting packet to the destination is the inner packet compressed with [RFC6282].

5. Sample/reference topology

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

Figure 6 shows the reference RPL Topology for this document. The letters above the nodes are there so that they may be referenced in subsequent sections. In the figure, 6LR represents a full router node. The 6LN is a RPL aware router, or host (as a leaf). Additionally, for simplification purposes, it is supposed that the 6LBR has direct access to Internet and is the root of the DODAG, thus the 6BBR is not present in the figure.

The 6LN leaves (RAL) marked as (F, H and I) are RPL nodes with no children hosts.

The leaves marked as RUL (G and J) are devices that do not speak RPL at all (not-RPL-aware), but use Router-Advertisements, 6LowPAN DAR/DAC and 6LoWPAN ND only to participate in the network [RFC8505]. In the document these leaves (G and J) are also referred to as a RUL.

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

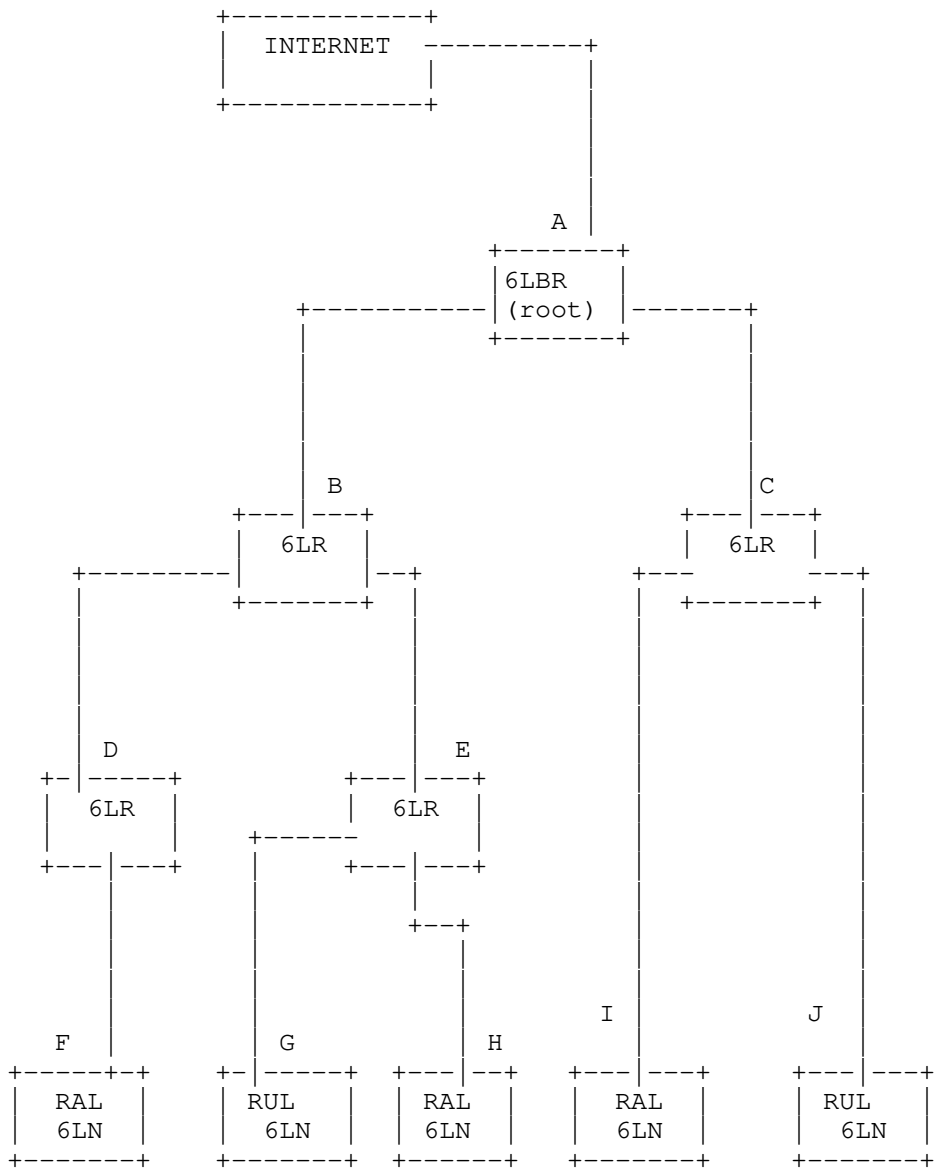


Figure 6: A reference RPL Topology.

6. Use cases

In the data plane a combination of RFC6553, RFC6554 and IPv6-in-IPv6 encapsulation are going to be analyzed for a number of representative traffic flows.

The use cases describe the communication in the following cases: - Between RPL-aware-nodes with the root (6LBR) - Between RPL-aware-nodes with the Internet - Between RUL nodes within the LLN (e.g. see Section 7.1.4) - Inside of the LLN when the final destination address resides outside of the LLN (e.g. see Section 7.2.3).

The use cases are as follows:

Interaction between Leaf and Root:

RAL to root

root to RAL

RUL to root

root to RUL

Interaction between Leaf and Internet:

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

Interaction between leaves:

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

This document is consistent with the rule that a Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed. This is a fundamental precept of the IPv6 architecture as outlined in [RFC8200].

As the rank information in the RPI artifact is changed at each hop, it will typically be zero when it arrives at the DODAG root. The DODAG root MUST force it to zero when passing the packet out to the Internet. The Internet will therefore not see any SenderRank information.

Despite being legal to leave the RPI artifact in place, an intermediate router that needs to add an extension header (e.g. RH3 or RPL Option) MUST still encapsulate the packet in an (additional) outer IP header. The new header is placed after this new outer IP header.

A corollary is that an intermediate router can remove an RH3 or RPL Option only if it is placed in an encapsulating IPv6 Header that is addressed TO this intermediate router. When doing the above, the whole encapsulating header must be removed. (A replacement may be added). This sometimes can result in outer IP headers being addressed to the next hop router using link-local address.

Both the RPL Option and the RH3 headers may be modified in very specific ways by routers on the path of the packet without the need to add and remove an encapsulating header. Both headers were designed with this modification in mind, and both the RPL RH3 and the RPL Option are marked mutable but recoverable: so an IPsec AH security header can be applied across these headers, but it can not secure the values which mutate.

The RPI MUST be present in every single RPL data packet.

Prior to [RFC8138], there was significant interest in creating an exception to this rule and removing the RPI for downward flows in non-storing mode. This exception covered a very small number of cases, and caused significant interoperability challenges while adding significant interest in the code and tests. The ability to compress the RPI down to three bytes or less removes much of the pressure to optimize this any further [I-D.ietf-anima-autonomic-control-plane].

Throughout the following subsections, the examples are described in more details in the first subsections, and more concisely in the later ones.

The uses cases are delineated based on the following IPV6 and RPL mandates:

The RPI has to be in every packet that traverses the LLN.

- Because of the above requirement, packets from the Internet have to be encapsulated.

- A Header cannot be inserted or removed on the fly inside an IPv6 packet that is being routed.

- Extension headers may not be added or removed except by the sender or the receiver.

- RPI and RH3 headers may be modified by routers on the path of the packet without the need to add and remove an encapsulating header.

- an RH3 or RPL Option can only be removed by an intermediate router if it is placed in an encapsulating IPv6 Header, which is addressed to the intermediate router.

- Non-storing mode requires downstream encapsulation by root for RH3.

The uses cases are delineated based on the following assumptions:

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

- Each IPv6 node (including Internet routers) obeys [RFC8200], so that 0x23 RPI Option Type can be safely inserted.

- All 6LRs obey [RFC8200].

- The RPI is ignored at the IPv6 dst node (RUL).

- In the uses cases, we assume that the RAL supports IP-in-IP encapsulation.

- In the uses cases, we don’t assume that the RUL supports IP-in-IP encapsulation.

- For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags.

- The description for RALs applies to RAN in general.

- Non-constrained uses of RPL are not in scope of this document.

- Compression is based on [RFC8138].

- The flow label [RFC6437] is not needed in RPL.

7. Storing mode

In storing mode (SM) (fully stateful), the sender can determine if the destination is inside the LLN by looking if the destination address is matched by the DIO's Prefix Information Option (PIO) option.

The following table (Figure 7) itemizes which headers are needed in each of the following scenarios. It indicates whether an IPv6-in-IPv6 header must be added and what destination it must be addressed to: (1) the final destination (the RAL node that is the target (tgt)), (2) the "root", or (3) the 6LR parent of a RUL.

In cases where no IPv6-in-IPv6 header is needed, the column states "No", and the destination is N/A (Not Applicable). If the IPv6-in-IPv6 header is needed, the column shows "must".

In all cases, the RPI is needed, since it identifies inconsistencies (loops) in the routing topology. In general, the RH3 is not needed because it is not used in storing mode. However, there is one scenario (from the root to the RUL in SM) where the RH3 can be used to point at the RUL (Figure 11).

The leaf can be a router 6LR or a host, both indicated as 6LN. The root refers to the 6LBR (see Figure 6).

Interaction between	Use Case	IPv6-in-IPv6	IPv6-in-IPv6 dst
Leaf - Root	RAL to root	No	N/A
	root to RAL	No	N/A
	root to RUL	must	6LR
	RUL to root	must	root
Leaf - Internet	RAL to Int	may	root
	Int to RAL	must	RAL (tgt)
	RUL to Int	must	root
	Int to RUL	must	6LR
Leaf - Leaf	RAL to RAL	No	N/A
	RAL to RUL	No(up)	N/A
		must(down)	6LR
	RUL to RAL	must(up)	root
		must(down)	RAL
	RUL to RUL	must(up)	root
		must(down)	6LR

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

7.1. Storing Mode: Interaction between Leaf and Root

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

RAL to root

root to RAL

RUL to root

root to RUL

7.1.1. SM: Example of Flow from RAL to Root

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

In this case the flow comprises:

RAL (6LN) --> 6LR_i --> root(6LBR)

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

The RAL (Node F) inserts the RPI, and sends the packet to 6LR (Node D) which decrements the rank in the RPI and sends the packet up. When the packet arrives at 6LBR (Node A), the RPI is removed and the packet is processed.

No IPv6-in-IPv6 header is required.

The RPI can be removed by the 6LBR because the packet is addressed to the 6LBR. The RAL must know that it is communicating with the 6LBR to make use of this scenario. The RAL can know the address of the 6LBR because it knows the address of the root via the DODAGID in the DIO messages.

The Figure 8 summarizes what headers are needed for this use case.

Header	RAL src	6LR_i	6LBR dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 8: SM: Summary of the use of headers from RAL to root

7.1.2. SM: Example of Flow from Root to RAL

In this case the flow comprises:

root (6LBR) --> 6LR_i --> RAL (6LN)

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

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

No IPv6-in-IPv6 header is required.

The Figure 9 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	RAL dst
Added headers	RPI	--	--
Modified headers	--	RPI	--
Removed headers	--	--	RPI
Untouched headers	--	--	--

Figure 9: SM: Summary of the use of headers from root to RAL

7.1.3. SM: Example of Flow from Root to RUL

In this case the flow comprises:

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

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

6LR_i (Node B) represents the intermediate routers from the source (6LBR) to the destination (RUL), 1 <= i <= n, where n is the total

number of routers (6LR) that the packet goes through from the 6LBR (Node A) to the RUL (Node G).

The 6LBR will encapsulate the packet in an IPv6-in-IPv6 header, and prepend an RPI. The IPv6-in-IPv6 header is addressed to the 6LR parent of the RUL (6LR_n). The 6LR parent of the RUL removes the header and sends the packet to the RUL.

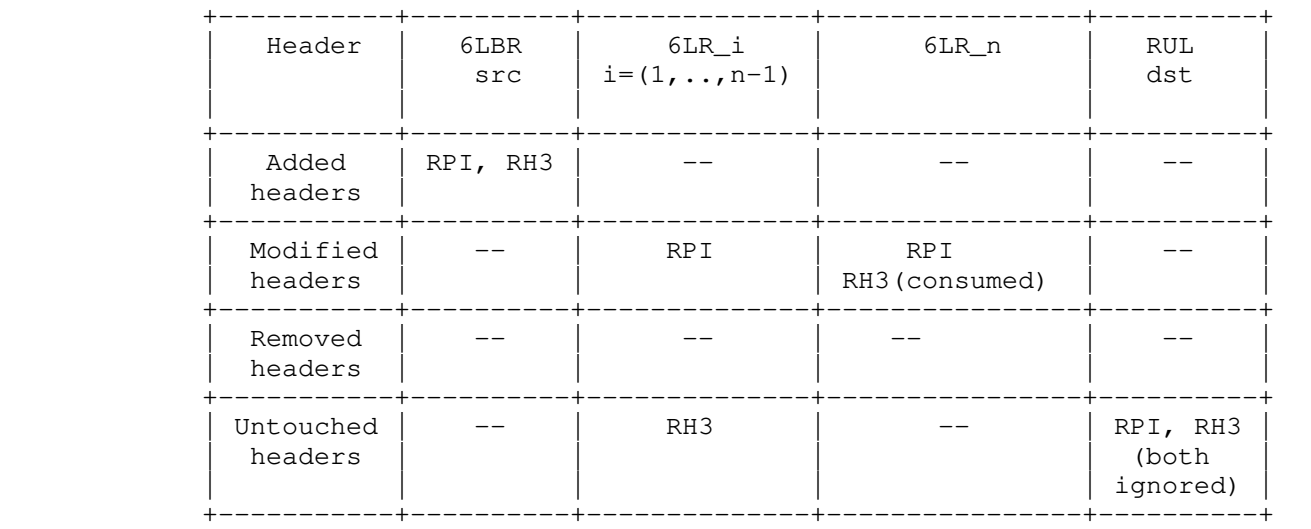
The Figure 10 summarizes what headers are needed for this use case.

Header	6LBR src	6LR_i	6LR_n	RUL dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 10: SM: Summary of the use of headers from root to RUL

IP-in-IP encapsulation may be avoided for Root to RUL communication. In SM, it can be replaced by a loose RH3 header that indicates the RUL, in which case the packet is routed to the 6LR as a normal SM operation, then the 6LR forwards to the RUL based on the RH3, and the RUL ignores both the consumed RH3 and the RPI, as in Non-Storing Mode.

The Figure 11 summarizes what headers are needed for this scenario.



Header	6LBR src	6LR_i i=(1,..,n-1)	6LR_n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI	RPI RH3(consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	RH3	--	RPI, RH3 (both ignored)

Figure 11: SM: Summary of the use of headers from root to RUL without encapsulation

7.1.4. SM: Example of Flow from RUL to Root

In this case the flow comprises:

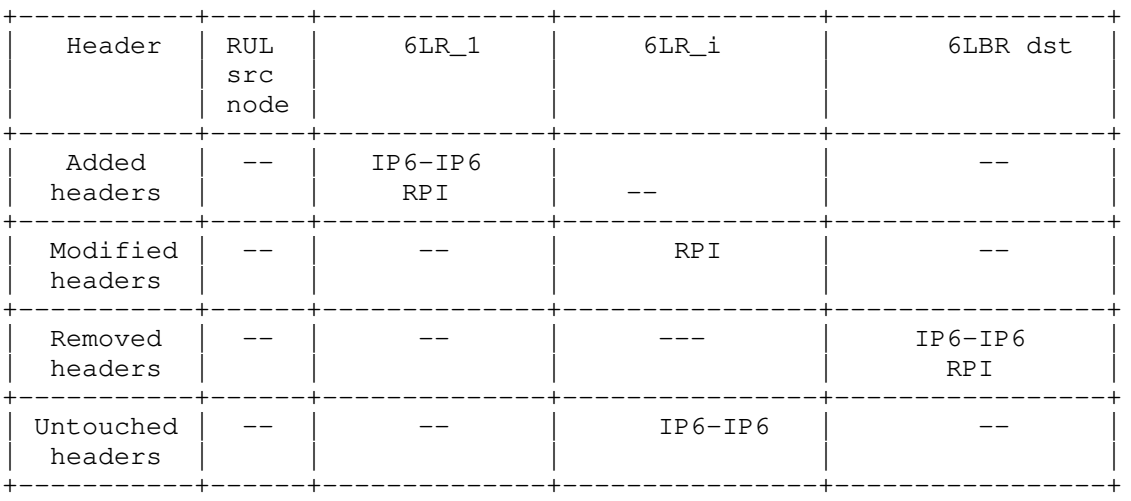
RUL (IPv6 src node) --> 6LR_1 --> 6LR_i --> root (6LBR)

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

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

When the packet arrives from the RUL (Node G) to 6LR_1 (Node E), the 6LR_1 will encapsulate the packet in an IPv6-in-IPv6 header with an RPI. The IPv6-in-IPv6 header is addressed to the root (Node A). The root removes the header and processes the packet.

The Figure 12 shows the table that summarizes what headers are needed for this use case where the IPv6-in-IPv6 header is addressed to the root (Node A).



Header	RUL src node	6LR_1	6LR_i	6LBR dst
Added headers	--	IP6-IP6 RPI	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	---	IP6-IP6 RPI
Untouched headers	--	--	IP6-IP6	--

Figure 12: SM: Summary of the use of headers from RUL to root.

7.2. SM: Interaction between Leaf and Internet.

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

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

7.2.1. SM: Example of Flow from RAL to Internet

In this case the flow comprises:

RAL (6LN) --> 6LR_i --> root (6LBR) --> Internet

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

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

RPL information from RFC 6553 may go out to Internet as it will be ignored by nodes which have not been configured to be RPI aware. No IPv6-in-IPv6 header is required.

On the other hand, the RAL may insert the RPI encapsulated in a IPv6-in-IPv6 header to the root. Thus, the root removes the RPI and send the packet to the Internet.

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

The Figure 13 summarizes what headers are needed for this use case when there is no encapsulation. Note that the RPI is modified by 6LBR to set the SenderRank to zero in case that it is not already zero. The Figure 14 summarizes what headers are needed when encapsulation to the root takes place.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

Figure 13: SM: Summary of the use of headers from RAL to Internet with no encapsulation

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	IP6-IP6 RPI	--	--	--
Modified headers	--	RPI	--	--
Removed headers	--	--	IP6-IP6 RPI	--
Untouched headers	--	IP6-IP6	--	--

Figure 14: SM: Summary of the use of headers from RAL to Internet with encapsulation to the root (6LBR).

7.2.2. SM: Example of Flow from Internet to RAL

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RAL (6LN)

For example, a communication flow could be: Internet --> Node A root(6LBR) --> Node B (6LR_1) --> Node D (6LR_n) --> Node F (RAL)

When the packet arrives from Internet to 6LBR the RPI is added in a outer IPv6-in-IPv6 header (with the IPv6-in-IPv6 destination address set to the RAL) and sent to 6LR, which modifies the rank in the RPI. When the packet arrives at the RAL, the packet is decapsulated, which removes the RPI before the packet is processed.

The Figure 15 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	RAL dst
Added headers	--	IP6-IP6(RPI)	--	--
Modified headers	--	--	RPI	--
Removed headers	--	--	--	IP6-IP6(RPI)
Untouched headers	--	--	--	--

Figure 15: SM: Summary of the use of headers from Internet to RAL.

7.2.3. SM: Example of Flow from RUL to Internet

In this case the flow comprises:

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

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

The node 6LR_1 (i=1) will add an IPv6-in-IPv6(RPI) header addressed to the root such that the root can remove the RPI before passing upwards. In the intermediate 6LR, the rank in the RPI is modified.

The originating node will ideally leave the IPv6 flow label as zero so that the packet can be better compressed through the LLN. The 6LBR will set the flow label of the packet to a non-zero value when sending to the Internet, for details check [RFC6437].

The Figure 16 shows the table that summarizes what headers are needed for this use case.

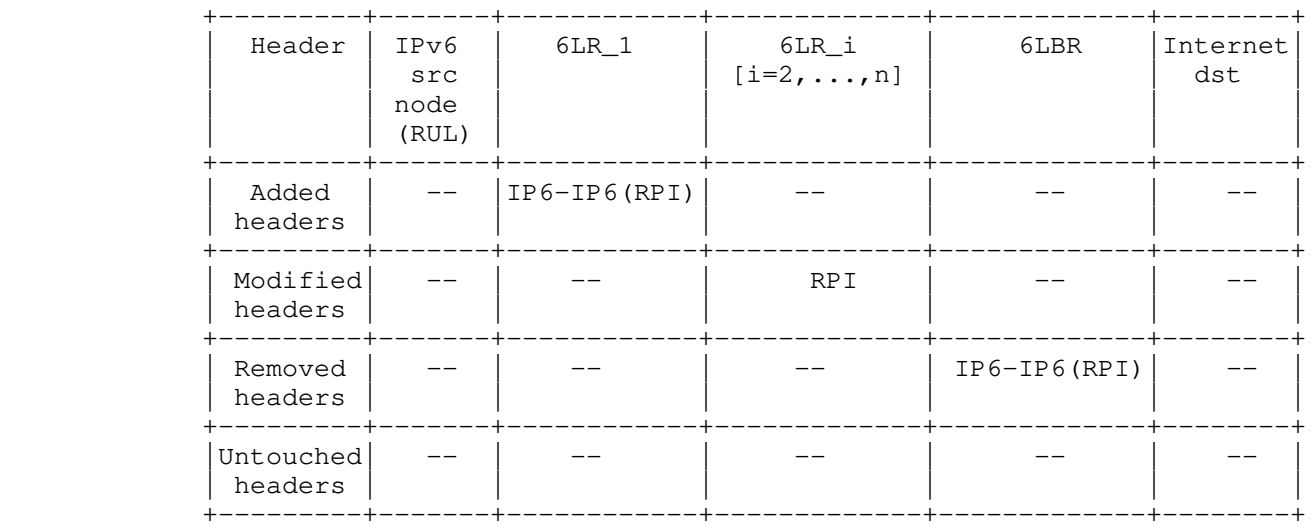


Figure 16: SM: Summary of the use of headers from RUL to Internet.

7.2.4. SM: Example of Flow from Internet to RUL.

In this case the flow comprises:

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

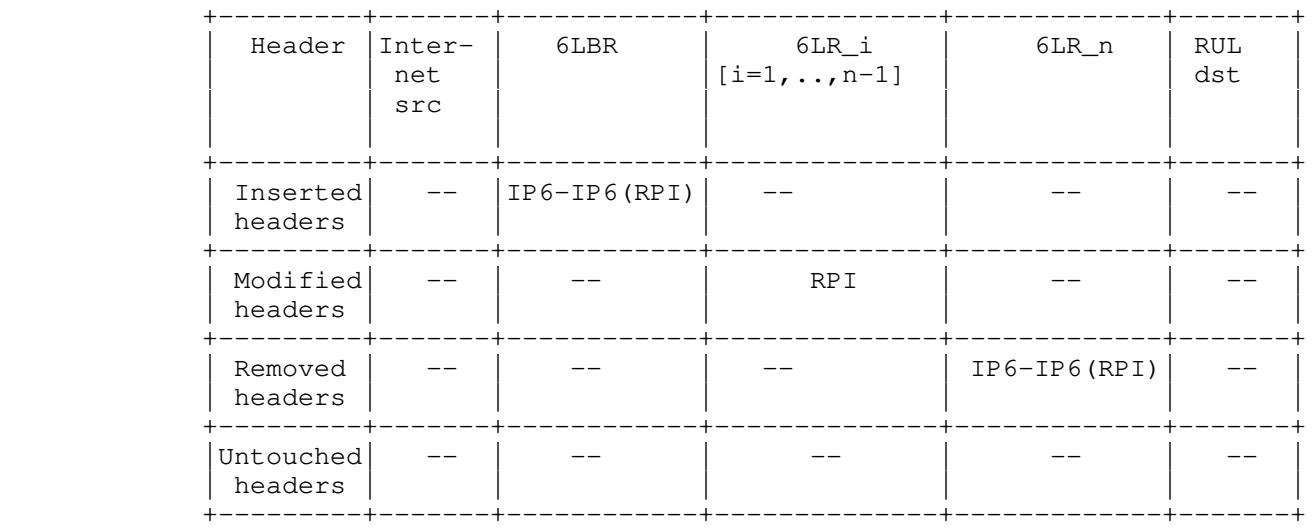
For example, a communication flow could be: Internet --> Node A root(6LBR) --> Node B (6LR_i)--> Node E (6LR_n) --> Node G (RUL)

The 6LBR will have to add an RPI within an IPv6-in-IPv6 header. The IPv6-in-IPv6 is addressed to the 6LR parent of the RUL.

Further details about this are mentioned in [I-D.ietf-roll-unaware-leaves], which specifies RPL routing for a 6LN acting as a plain host and not being aware of RPL.

The 6LBR may set the flow label on the inner IPv6-in-IPv6 header to zero in order to aid in compression [RFC8138][RFC6437].

The Figure 17 shows the table that summarizes what headers are needed for this use case.



Header	Inter-net src	6LBR	6LR_i [i=1,..,n-1]	6LR_n	RUL dst
Inserted headers	--	IP6-IP6(RPI)	--	--	--
Modified headers	--	--	RPI	--	--
Removed headers	--	--	--	IP6-IP6(RPI)	--
Untouched headers	--	--	--	--	--

Figure 17: SM: Summary of the use of headers from Internet to RUL.

7.3. SM: Interaction between Leaf and Leaf

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

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

7.3.1. SM: Example of Flow from RAL to RAL

In [RFC6550] RPL allows a simple one-hop optimization for both storing and non-storing networks. A node may send a packet destined to a one-hop neighbor directly to that node. See section 9 in [RFC6550].

When the nodes are not directly connected, then in storing mode, the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LR_x) --> 6LR_id --> RAL dst (6LN)

For example, a communication flow could be: Node F (RAL src)--> Node D (6LR_ia)--> Node B (6LR_x) --> Node E (6LR_id) --> Node H (RAL dst)

6LR_ia (Node D) represents the intermediate routers from source to the common parent (6LR_x) (Node B), 1 <= ia <= n, where n is the total number of routers (6LR) that the packet goes through from RAL (Node F) to the common parent 6LR_x (Node B).

6LR_id (Node E) represents the intermediate routers from the common parent (6LR_x) (Node B) to destination RAL (Node H), 1 <= id <= m, where m is the total number of routers (6LR) that the packet goes through from the common parent (6LR_x) to destination RAL (Node H).

It is assumed that the two nodes are in the same RPL domain (that they share the same DODAG root). At the common parent (Node B), the direction flag ('O' flag) of the RPI is changed (from decreasing ranks to increasing ranks).

While the 6LR nodes will update the RPI, no node needs to add or remove the RPI, so no IPv6-in-IPv6 headers are necessary.

The Figure 18 summarizes what headers are needed for this use case.

Header	RAL src	6LR_ia	6LR_x (common parent)	6LR_id	RAL dst
Added headers	RPI	--	--	--	--
Modified headers	--	RPI	RPI	RPI	--
Removed headers	--	--	--	--	RPI
Untouched headers	--	--	--	--	--

Figure 18: SM: Summary of the Use of Headers from RAL to RAL

7.3.2. SM: Example of Flow from RAL to RUL

In this case the flow comprises:

RAL src (6LN) --> 6LR_ia --> common parent (6LBR - The root-) --> 6LR_id --> RUL (IPv6 dst node)

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

6LR_ia represents the intermediate routers from source (RAL) to the common parent (the Root), 1 <= ia <= n, where n is the total number of routers (6LR) that the packet goes through from RAL to the Root.

6LR_id (Node E) represents the intermediate routers from the Root (Node B) to destination RUL (Node G). In this case, 1 <= id <= m, where m is the total number of routers (6LR) that the packet goes through from the Root down to the destination RUL.

In this case, the packet from the RAL goes to 6LBR because the route to the RUL is not injected into the RPL-SM. Thus, the RAL inserts an RPI (RPI1) addressed to the root(6LBR). The root does not remove the RPI1 (the root cannot remove an RPI if there is no encapsulation). The root inserts an IPv6-IPv6 encapsulation with an RPI2 and sends it to the 6LR parent of the RUL, which removes the encapsulation and RPI2 before passing the packet to the RUL.

The Figure 19 summarizes what headers are needed for this use case.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	RPI1	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 19: SM: Summary of the Use of Headers from RAL to RUL

7.3.3. SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_ia --> 6LBR --> 6LR_id --> RAL dst (6LN)

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

6LR_ia (Node E) represents the intermediate routers from source (RUL) (Node G) to the root (Node A). In this case, 1 <= ia <= n, where n is the total number of routers (6LR) that the packet goes through from source to the root.

6LR_id represents the intermediate routers from the root (Node A) to destination RAL (Node F). In this case, 1 <= id <= m, where m is the total number of routers (6LR) that the packet goes through from the root to the destination RAL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI1) encapsulated in a IPv6-in-IPv6 header to the root. The root removes the outer header including the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the destination RAL (Node F).

The Figure 20 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--	--
Modified headers	--	--	RPI1	--	RPI2	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)
Untouched headers	--	--	--	--	--	--

Figure 20: SM: Summary of the use of headers from RUL to RAL.

7.3.4. SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node)--> 6LR_1--> 6LR_ia --> 6LBR --> 6LR_id --> RUL (IPv6 dst node)

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

Internal nodes 6LR_ia (e.g: Node E or Node B) is the intermediate router from the RUL source (Node G) to the root (6LBR) (Node A). In this case, 1 <= ia <= n, where n is the total number of routers (6LR) that the packet goes through from the RUL to the root. 6LR_1 refers when ia=1.

6LR_id (Node C) represents the intermediate routers from the root (Node A) to the destination RUL dst node (Node J). In this case, 1 <= id <= m, where m is the total number of routers (6LR) that the packet goes through from the root to destination RUL.

The 6LR_1 (Node E) receives the packet from the RUL (Node G) and inserts the RPI (RPI), encapsulated in an IPv6-in-IPv6 header directed to the root. The root removes the outer header including

the RPI (RPI1) and inserts a new RPI (RPI2) addressed to the 6LR father of the RUL.

The Figure 21 shows the table that summarizes what headers are needed for this use case.

Header	RUL src	6LR_1	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst
Added Headers	--	IP6-IP6(RPI1)	--	IP6-IP6 (RPI2)	--	--	--
Modified headers	--	--	RPI1	--	RPI2	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 21: SM: Summary of the use of headers from RUL to RUL

8. Non Storing mode

In Non Storing Mode (Non-SM) (fully source routed), the 6LBR (DODAG root) has complete knowledge about the connectivity of all DODAG nodes, and all traffic flows through the root node. Thus, there is no need for all nodes to know about the existence of RPL-unaware nodes. Only the 6LBR needs to act if compensation is necessary for not-RPL aware receivers.

The table (Figure 22) summarizes what headers are needed in the following scenarios, and indicates when the RPI, RH3 and IPv6-in-IPv6 header are to be inserted. The last column depicts the target destination of the IPv6-in-IPv6 header: 6LN (indicated by "RAL"), 6LR (parent of a RUL) or the root. In cases where no IPv6-in-IPv6 header is needed, the column indicates "No". There is no expectation on RPL that RPI can be omitted, because it is needed for routing, quality of service and compression. This specification expects that an RPI is always present. The term "may(up)" means that the IPv6-in-IPv6 header may be necessary in the upwards direction. The term "must(up)" means that the IPv6-in-IPv6 header must be present in the upwards direction. The term "must(down)" means that the IPv6-in-IPv6 header must be present in the downward direction.

The leaf can be a router 6LR or a host, both indicated as 6LN (Figure 6). In the table (Figure 22) the (1) indicates a 6tisch case [RFC8180], where the RPI may still be needed for the RPLInstanceID to be available for priority/channel selection at each hop.

Interaction between	Use Case	RPI	RH3	IPv6-in-IPv6	IP-in-IP dst
Leaf - Root	RAL to root	Yes	No	No	No
	root to RAL	Yes	Yes	No	No
	root to RUL	Yes (1)	Yes	No	6LR
	RUL to root	Yes	No	must	root
Leaf - Internet	RAL to Int	Yes	No	may(up)	root
	Int to RAL	Yes	Yes	must	RAL
	RUL to Int	Yes	No	must	root
	Int to RUL	Yes	Yes	must	6LR
Leaf - Leaf	RAL to RAL	Yes	Yes	may(up)	root
				must(down)	RAL
	RAL to RUL	Yes	Yes	may(up)	root
				must(down)	6LR
	RUL to RAL	Yes	Yes	must(up)	root
				must(down)	RAL
	RUL to RUL	Yes	Yes	must(up)	root
				must(down)	6LR

Figure 22: Table that shows headers needed in Non-Storing mode: RPI, RH3, IPv6-in-IPv6 encapsulation.

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

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

RAL to root

root to RAL

RUL to root

root to RUL

8.1.1. Non-SM: Example of Flow from RAL to root

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

RAL (6LN) --> 6LR_i --> root(6LBR)

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

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

This situation is the same case as storing mode.

The Figure 23 summarizes what headers are needed for this use case.

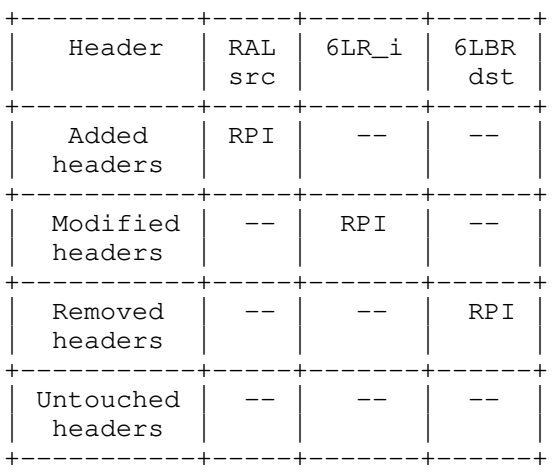


Figure 23: Non-SM: Summary of the use of headers from RAL to root

8.1.2. Non-SM: Example of Flow from root to RAL

In this case the flow comprises:

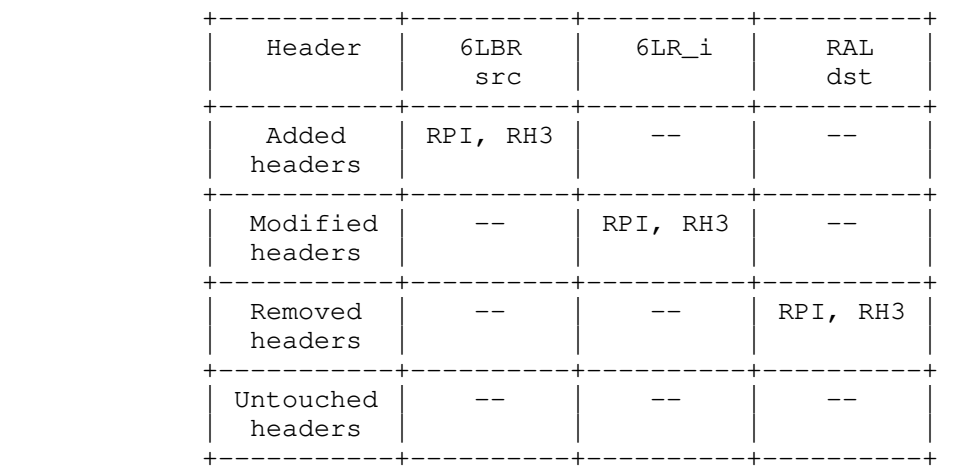
root (6LBR) --> 6LR_i --> RAL (6LN)

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

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

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

The Figure 24 summarizes what headers are needed for this use case.



Header	6LBR src	6LR_i	RAL dst
Added headers	RPI, RH3	--	--
Modified headers	--	RPI, RH3	--
Removed headers	--	--	RPI, RH3
Untouched headers	--	--	--

Figure 24: Non-SM: Summary of the use of headers from root to RAL

8.1.3. Non-SM: Example of Flow from root to RUL

In this case the flow comprises:

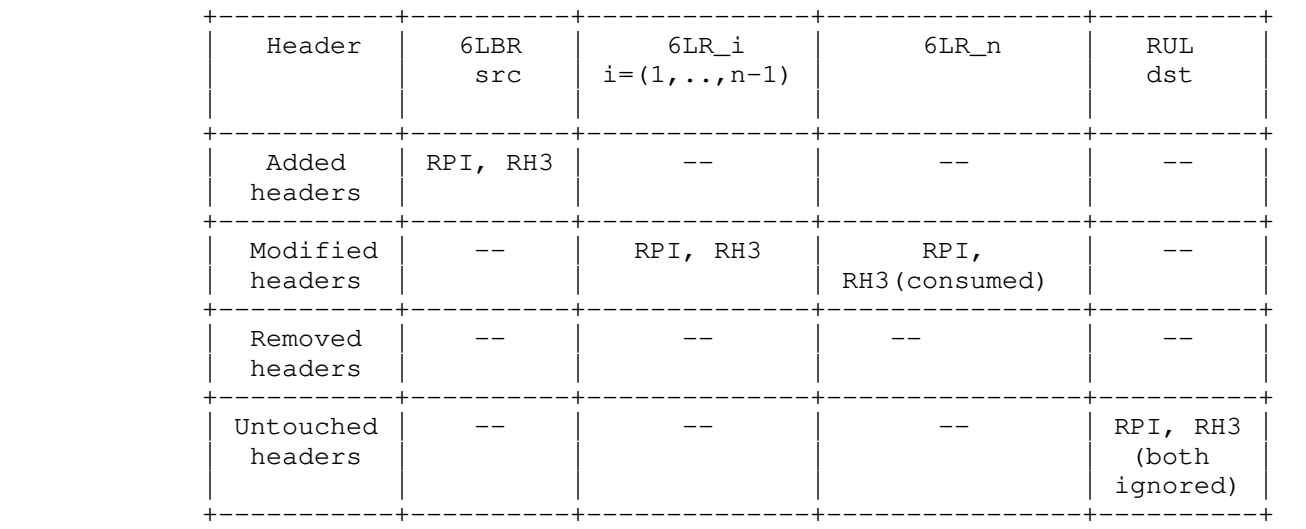
root (6LBR) --> 6LR_i --> RUL (IPv6 dst node)

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

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

In the 6LBR, the RH3 is added; it is then modified at each intermediate 6LR (6LR_1 and so on), and it is fully consumed in the last 6LR (6LR_n) but is left in place. When the RPI is added, the RUL, which does not understand the RPI, will ignore it (per [RFC8200]); thus, encapsulation is not necessary.

The Figure 25 depicts the table that summarizes what headers are needed for this use case.



Header	6LBR src	6LR_i i=(1,..,n-1)	6LR_n	RUL dst
Added headers	RPI, RH3	--	--	--
Modified headers	--	RPI, RH3	RPI, RH3(consumed)	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI, RH3 (both ignored)

Figure 25: Non-SM: Summary of the use of headers from root to RUL

8.1.4. Non-SM: Example of Flow from RUL to root

In this case the flow comprises:

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

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

6LR_i represents the intermediate routers from source to destination. In this case, 1 <= i <= n, where n is the total number of routers (6LR) that the packet goes through from source (RUL) to destination (6LBR). For example, 6LR_1 (i=1) is the router that receives the packets from the RUL.

In this case, the RPI is added by the first 6LR (6LR_1) (Node E), encapsulated in an IPv6-in-IPv6 header, and modified in the subsequent 6LRs in the flow. The RPI and the entire packet are consumed by the root.

The Figure 26 shows the table that summarizes what headers are needed for this use case.

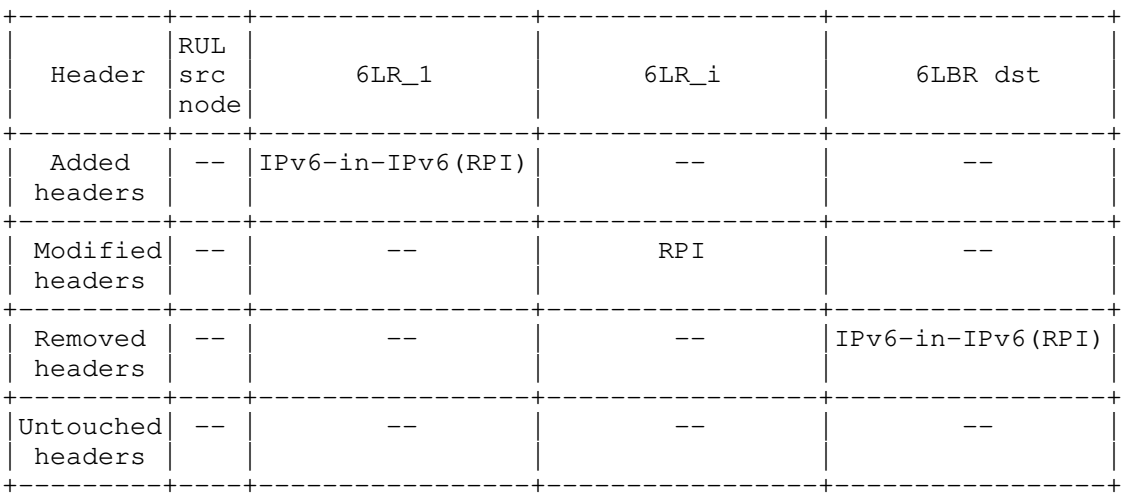


Figure 26: Non-SM: Summary of the use of headers from RUL to root

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

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

RAL to Internet

Internet to RAL

RUL to Internet

Internet to RUL

8.2.1. Non-SM: Example of Flow from RAL to Internet

In this case the flow comprises:

RAL (6LN) src --> 6LR_i --> root (6LBR) --> Internet dst

For example, a communication flow could be: Node F (RAL) --> Node D --> Node B --> Node A --> Internet. Having the RAL information about the RPL domain, the packet may be encapsulated to the root when the destination is not in the RPL domain of the RAL.

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

In this case, the encapsulation from the RAL to the root is optional. The simplest case is when the RPI gets to the Internet (as the Figure 27 shows it), knowing that the Internet is going to ignore it.

The IPv6 flow label should be set to zero to aid in compression [RFC8138], and the 6LBR will set it to a non-zero value when sending towards the Internet [RFC6437].

The Figure 27 summarizes what headers are needed for this use case when no encapsulation is used. The Figure 28 summarizes what headers are needed for this use case when encapsulation to the root is used.

Header	RAL src	6LR_i	6LBR	Internet dst
Added headers	RPI	--	--	--
Modified headers	--	RPI	RPI	--
Removed headers	--	--	--	--
Untouched headers	--	--	--	RPI (Ignored)

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

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

Figure 28: Non-SM: Summary of the use of headers from RAL to Internet with encapsulation to the root

8.2.2. Non-SM: Example of Flow from Internet to RAL

In this case the flow comprises:

Internet --> root (6LBR) --> 6LR_i --> RAL dst (6LN)

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

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

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

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

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

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

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

In this case the flow comprises:

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

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

6LR_i represents the intermediate routers from source to destination, 1 <= i <= n, where n is the total number of routers (6LRs) that the packet goes through from the source (RUL) to the 6LBR, e.g., 6LR_1 (i=1).

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

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

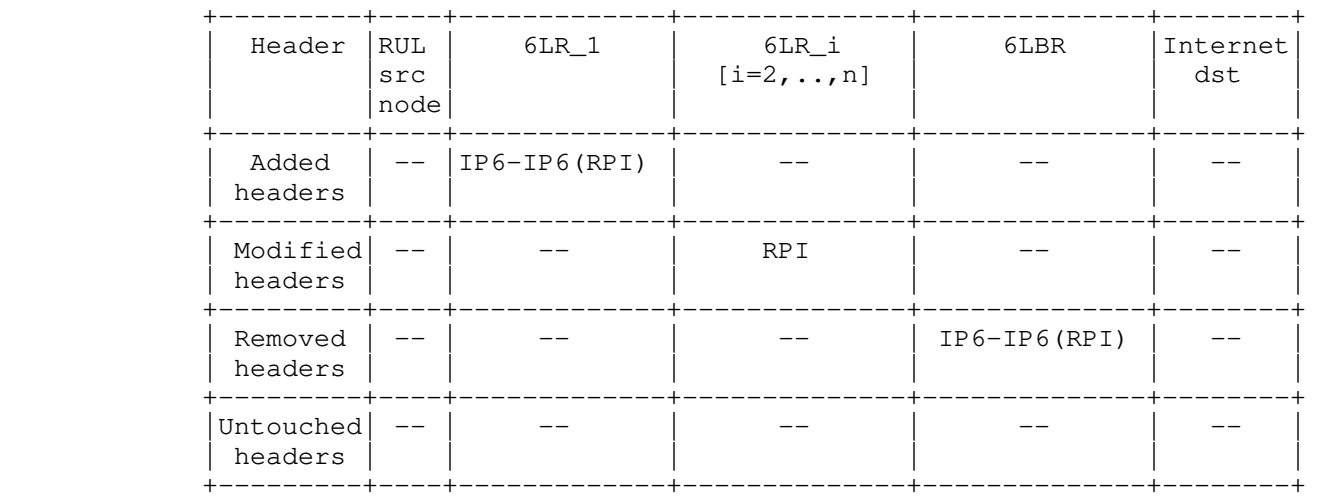


Figure 30: Non-SM: Summary of the use of headers from RUL to Internet

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

In this case the flow comprises:

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

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

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

The 6LBR must add an RH3 header inside an IPv6-in-IPv6 header. The 6LBR will know the path, and will recognize that the final node is not a RPL capable node as it will have received the connectivity DAO from the nearest 6LR. The 6LBR can therefore make the IPv6-in-IPv6 header destination be the last 6LR. The 6LBR will set to zero the flow label upon entry in order to aid compression [RFC8138].

The Figure 31 shows the table that summarizes what headers are needed for this use case.

Header	Internet src	6LBR	6LR_i	6LR_n	RUL dst
Added headers	--	IP6-IP6(RH3,RPI)	--	--	--
Modified headers	--	--	IP6-IP6 (RH3,RPI)	--	--
Removed headers	--	--	--	IP6-IP6 (RH3,RPI)	--
Untouched headers	--	--	--	--	--

Figure 31: Non-SM: Summary of the use of headers from Internet to RUL.

8.3. Non-SM: Interaction between leaves

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

RAL to RAL

RAL to RUL

RUL to RAL

RUL to RUL

8.3.1. Non-SM: Example of Flow from RAL to RAL

In this case the flow comprises:

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

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

6LR_ia represents the intermediate routers from source to the root, 1 <= ia <= n, where n is the total number of routers (6LR) that the packet goes through from RAL to the root.

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

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

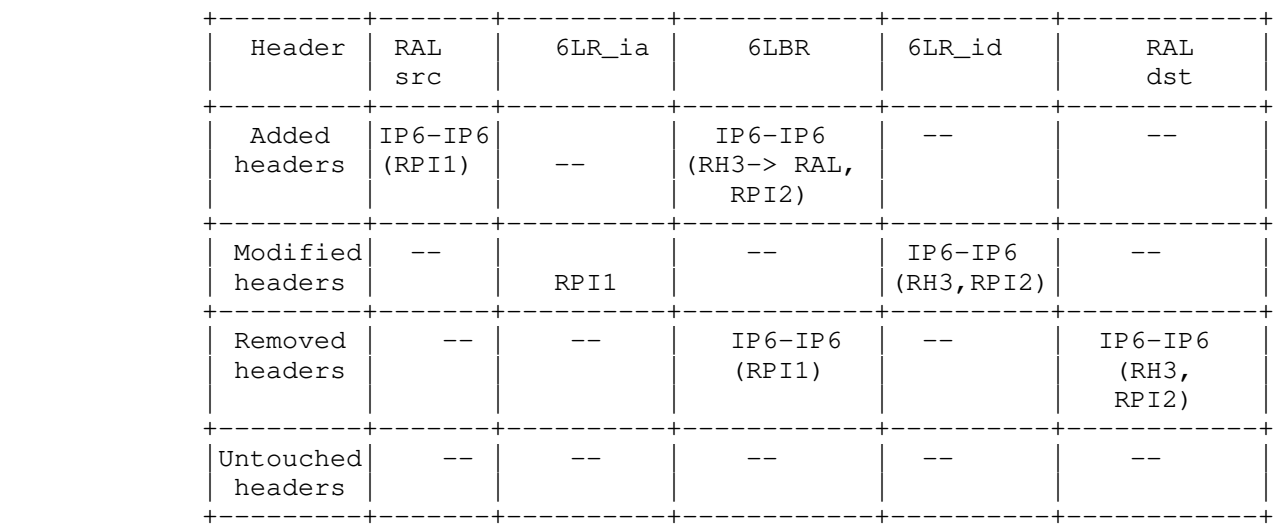
The originating node may put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root, so that the 6LBR can remove that header. If it does not, then the RPI1 is forwarded down from the root in the inner header to no avail.

The 6LBR will need to insert an RH3 header, which requires that it add an IPv6-in-IPv6 header. It removes the RPI(RPI1), as it was contained in an IPv6-in-IPv6 header addressed to it. Otherwise, there may be an RPI buried inside the inner IP header, which should get ignored. The root inserts an RPI (RPI2) alongside the RH3.

Networks that use the RPL P2P extension [RFC6997] are essentially non-storing DODAGs and fall into this scenario or scenario Section 8.1.2, with the originating node acting as 6LBR.

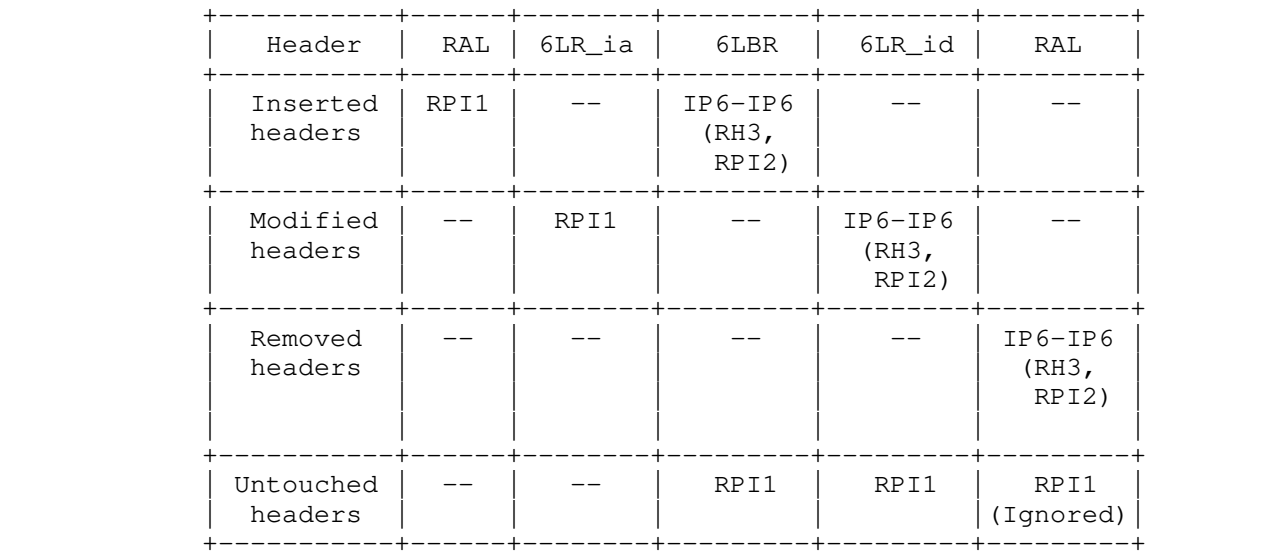
The Figure 32 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place.

The Figure 33 shows the table that summarizes what headers are needed for this use case when there is no encapsulation to the root. Note that in the Modified headers row, going up in each 6LR_ia only the RPI1 is changed. Going down, in each 6LR_id the IPv6 header is swapped with the RH3 so both are changed alongside with the RPI2.



Header	RAL src	6LR_ia	6LBR	6LR_id	RAL dst
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3-> RAL, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3,RPI2)	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--

Figure 32: Non-SM: Summary of the Use of Headers from RAL to RAL with encapsulation to the root.



Header	RAL	6LR_ia	6LBR	6LR_id	RAL
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	RPI1	RPI1	RPI1 (Ignored)

Figure 33: Non-SM: Summary of the Use of Headers from RAL to RAL without encapsulation to the root.

8.3.2. Non-SM: Example of Flow from RAL to RUL

In this case the flow comprises:

RAL --> 6LR_ia --> root (6LBR) --> 6LR_id --> RUL (IPv6 dst node)

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

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

6LR_id represents the intermediate routers from the root to the destination, 1 <= id <= m, where m is the total number of the intermediate routers (6LRs).

As in the previous case, the RAL (6LN) may insert an RPI (RPI1) header which must be in an IPv6-in-IPv6 header addressed to the root so that the 6LBR can remove this RPI. The 6LBR will then insert an RH3 inside a new IPv6-in-IPv6 header addressed to the last 6LR_id (6LR_id = m) alongside the insertion of RPI2.

If the originating node does not put the RPI (RPI1) into an IPv6-in-IPv6 header addressed to the root. Then, the RPI1 is forwarded down from the root in the inner header to no avail.

The Figure 34 shows the table that summarizes what headers are needed for this use case when encapsulation to the root takes place. The Figure 35 shows the table that summarizes what headers are needed for this use case when no encapsulation to the root takes place.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--

Figure 34: Non-SM: Summary of the use of headers from RAL to RUL with encapsulation to the root.

Header	RAL src node	6LR_ia	6LBR	6LR_id	6LR_n	RUL dst node
Inserted headers	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	RPI1	RPI1	RPI1	RPI1 (Ignored)

Figure 35: Non-SM: Summary of the use of headers from RAL to RUL without encapsulation to the root.

8.3.3. Non-SM: Example of Flow from RUL to RAL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id --> RAL dst (6LN)

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

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

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

In this scenario the RPI (RPI1) is added by the first 6LR (6LR_1) inside an IPv6-in-IPv6 header addressed to the root. The 6LBR will remove this RPI, and add its own IPv6-in-IPv6 header containing an RH3 header and an RPI (RPI2).

The Figure 36 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	RAL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)
Untouched headers	--	--	--	--	--	--

Figure 36: Non-SM: Summary of the use of headers from RUL to RAL.

8.3.4. Non-SM: Example of Flow from RUL to RUL

In this case the flow comprises:

RUL (IPv6 src node) --> 6LR_1 --> 6LR_ia --> root (6LBR) --> 6LR_id --> RUL (IPv6 dst node)

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

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

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

This scenario is the combination of the previous two cases.

The Figure 37 shows the table that summarizes what headers are needed for this use case.

Header	RUL src node	6LR_1	6LR_ia	6LBR	6LR_id	6LR_m	RUL dst node
Added headers	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--	--	--
Modified headers	--	--	RPI1	--	IP6-IP6 (RH3, RPI2)	--	--
Removed headers	--	--	--	IP6-IP6 (RPI1)	--	IP6-IP6 (RH3, RPI2)	--
Untouched headers	--	--	--	--	--	--	--

Figure 37: Non-SM: Summary of the use of headers from RUL to RUL

9. Operational Considerations of supporting RUL-leaves

Roughly half of the situations described in this document involve leaf ("host") nodes that do not speak RPL. These nodes fall into two further categories: ones that drop a packet that have RPI or RH3 headers, and ones that continue to process a packet that has RPI and/or RH3 headers.

[RFC8200] provides for new rules that suggest that nodes that have not been configured (explicitly) to examine Hop-by-Hop headers, should ignore those headers, and continue processing the packet. Despite this, and despite the switch from 0x63 to 0x23, there may be nodes that are pre-RFC8200, or simply intolerant. Those nodes will drop packets that continue to have RPL artifacts in them. In general, such nodes can not be easily supported in RPL LLNs.

There are some specific cases where it is possible to remove the RPL artifacts prior to forwarding the packet to the leaf host. The critical thing is that the artifacts have been inserted by the RPL root inside an IPv6-in-IPv6 header, and that the header has been addressed to the 6LR immediately prior to the leaf node. In that case, in the process of removing the IPv6-in-IPv6 header, the artifacts can also be removed.

The above case occurs whenever traffic originates from the outside the LLN (the "Internet" cases above), and non-storing mode is used. In non-storing mode, the RPL root knows the exact topology (as it must create the RH3 header) and therefore knows which 6LR is prior to the leaf. For example, in Figure 6, Node E is the 6LR prior to leaf Node G, or Node C is the 6LR prior to leaf Node J.

Traffic originating from the RPL root (such as when the data collection system is co-located on the RPL root), does not require an IPv6-in-IPv6 header (in storing or non-storing mode), as the packet is originating at the root, and the root can insert the RPI and RH3 headers directly into the packet, as it is formed. Such a packet is slightly smaller, but only can be sent to nodes (whether RPL aware or not), that will tolerate the RPL artifacts.

An operator that finds itself with a high amount of traffic from the RPL root to RPL-not-aware-leaves, will have to do IPv6-in-IPv6 encapsulation if the leaf is not tolerant of the RPL artifacts. Such an operator could otherwise omit this unnecessary header if it was certain of the properties of the leaf.

As storing mode can not know the final path of the traffic, intolerant (that drop packets with RPL artifacts) leaf nodes can not be supported.

10. Operational considerations of introducing 0x23

This section describes the operational considerations of introducing the new RPI Option Type of 0x23.

During bootstrapping the node gets the DIO with the information of RPI Option Type, indicating the new RPI in the DODAG Configuration option Flag. The DODAG root is in charge to configure the current network to the new value, through DIO messages and when all the nodes are set with the new value. The DODAG should change to a new DODAG version. In case of rebooting, the node does not remember the RPI Option Type. Thus, the DIO is sent with a flag indicating the new RPI Option Type.

The DODAG Configuration option is contained in a RPL DIO message, which contains a unique DTSN counter. The leaf nodes respond to this message with DAO messages containing the same DTSN. This is a normal part of RPL routing; the RPL root therefore knows when the updated DODAG Configuration option has been seen by all nodes.

Before the migration happens, all the RPL-aware nodes should support both values . The migration procedure is triggered when the DIO is sent with the flag indicating the new RPI Option Type. Namely, it remains at 0x63 until it is sure that the network is capable of 0x23, then it abruptly changes to 0x23. The 0x23 RPI Option allows to send packets to not-RPL nodes. The not-RPL nodes should ignore the option and continue processing the packets.

As mentioned previously, indicating the new RPI in the DODAG Configuration option flag is a way to avoid the flag day (abrupt changeover) in a network using 0x63 as the RPI Option Type value. It is suggested that RPL implementations accept both 0x63 and 0x23 RPI Option type values when processing the header to enable interoperability.

11. IANA Considerations

11.1. Option Type in RPL Option

This document updates the registration made in [RFC6553] Destination Options and Hop-by-Hop Options registry from 0x63 to 0x23 as shown in Figure 38.

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

Figure 38: Option Type in RPL Option.(*)represents this document

DODAG Configuration option is updated as follows (Figure 39):

Bit number	Description	Reference
3	RPI 0x23 enable	This document

Figure 39: DODAG Configuration option Flag to indicate the RPI-flag-day.

11.2. Change to the DODAG Configuration Options Flags registry

This document requests IANA to change the name of the "DODAG Configuration Option Flags" registry to "DODAG Configuration Option Flags for MOP 0..6".

This document requests to be mentioned as a reference for this change.

11.3. Change MOP value 7 to Reserved

This document requests the changing the registration status of value 7 in the Mode of Operation registry from Unassigned to Reserved. This change is in support of future work.

This document requests to be mentioned as a reference for this entry in the registry.

12. Security Considerations

The security considerations covered in [RFC6553] and [RFC6554] apply when the packets are in the RPL Domain.

The IPv6-in-IPv6 mechanism described in this document is much more limited than the general mechanism described in [RFC2473]. The willingness of each node in the LLN to decapsulate packets and forward them could be exploited by nodes to disguise the origin of an attack.

While a typical LLN may be a very poor origin for attack traffic (as the networks tend to be very slow, and the nodes often have very low duty cycles), given enough nodes, LLNs could still have a significant impact, particularly if the attack is targeting another LLN. Additionally, some uses of RPL involve large backbone ISP scale equipment [I-D.ietf-anima-autonomic-control-plane], which may be equipped with multiple 100Gb/s interfaces.

Blocking or careful filtering of IPv6-in-IPv6 traffic entering the LLN as described above will make sure that any attack that is mounted must originate from compromised nodes within the LLN. The use of BCP38 [BCP38] filtering at the RPL root on egress traffic will both alert the operator to the existence of the attack, as well as drop the attack traffic. As the RPL network is typically numbered from a single prefix, which is itself assigned by RPL, BCP38 filtering involves a single prefix comparison and should be trivial to automatically configure.

There are some scenarios where IPv6-in-IPv6 traffic should be allowed to pass through the RPL root, such as the IPv6-in-IPv6 mediated communications between a new Pledge and the Join Registrar/ Coordinator (JRC) when using [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-6tisch-dtsecurity-zerotouch-join]. This is the case for the RPL root to do careful filtering: it occurs only when the Join Coordinator is not co-located inside the RPL root.

With the above precautions, an attack using IPv6-in-IPv6 tunnels can only be by a node within the LLN on another node within the LLN. Such an attack could, of course, be done directly. An attack of this kind is meaningful only if the source addresses are either fake or if the point is to amplify return traffic. Such an attack, could also be done without the use of IPv6-in-IPv6 headers using forged source addresses. If the attack requires bi-directional communication, then IPv6-in-IPv6 provides no advantages.

Whenever IPv6-in-IPv6 headers are being proposed, there is a concern about creating security issues. In the Security Considerations

section of [RFC2473], it was suggested that tunnel entry and exit points can be secured by securing the IPv6 path between them. This recommendation is not practical for RPL networks. [RFC5406] goes into some detail on what additional details would be needed in order to "Use IPsec". Use of ESP would prevent [RFC8138] compression (compression must occur before encryption), and [RFC8138] compression is lossy in a way that prevents use of AH. These are minor issues. The major issue is how to establish trust enough such that IKEv2 could be used. This would require a system of certificates to be present in every single node, including any Internet nodes that might need to communicate with the LLN. Thus, using IPsec requires a global PKI in the general case.

More significantly, the use of IPsec tunnels to protect the IPv6-in-IPv6 headers would in the general case scale with the square of the number of nodes. This is a lot of resource for a constrained nodes on a constrained network. In the end, the IPsec tunnels would be providing only BCP38-like origin authentication! That is, IPsec provides a transitive guarantee to the tunnel exit point that the tunnel entry point did BCP38 on traffic going in. Just doing origin filtering per BCP 38 at the entry and exit of the LLN provides a similar level of security without all the scaling and trust problems related to IPv6 tunnels as discussed in RFC 2473. IPsec is not recommended.

An LLN with hostile nodes within it would not be protected against impersonation with the LLN by entry/exit filtering.

The RH3 header usage described here can be abused in equivalent ways. An external attacker may form a packet with an RH3 that is not fully consumed and encapsulate it to hide the RH3 from intermediate nodes and disguise the origin of traffic. As such, the attacker's RH3 header will not be seen by the network until it reaches the destination, which will decapsulate it. As indicated in section 4.2 of [RFC6554], RPL routers are responsible for ensuring that an SRH is only used between RPL routers. As such, if there is an RH3 that is not fully consumed in the encapsulated packet, the node that decapsulates it MUST ensure that the outer packet was originated in the RPL domain and drop the packet otherwise.

Also, as indicated by section 2 of [RFC6554], RPL Border Routers "do not allow datagrams carrying an SRH header to enter or exit a RPL routing domain". This sentence must be understood as concerning non-fully-consumed packets. A consumed (inert) RH3 header could be present in a packet that flows from one LLN, crosses the Internet, and enters another LLN. As per the discussion in this document, such headers do not need to be removed. However, there is no case described in this document where an RH3 is inserted in a non-storing

network on traffic that is leaving the LLN, but this document should not preclude such a future innovation.

In short, a packet that crosses the border of the RPL domain MAY carry and RH3, and if so, that RH3 MUST be fully consumed.

The RPI, if permitted to enter the LLN, could be used by an attacker to change the priority of a packet by selecting a different RPLInstanceID, perhaps one with a higher energy cost, for instance. It could also be that not all nodes are reachable in an LLN using the default RPLInstanceID, but a change of RPLInstanceID would permit an attacker to bypass such filtering. Like the RH3, an RPI is to be inserted by the RPL root on traffic entering the LLN by first inserting an IPv6-in-IPv6 header. The attacker's RPI therefore will not be seen by the network. Upon reaching the destination node the RPI has no further meaning and is just skipped; the presence of a second RPI will have no meaning to the end node as the packet has already been identified as being at it's final destination.

For traffic leaving a RUL, if the RUL adds an opaque RPI then the 6LR as a RPL border router SHOULD rewrite the RPI to indicate the selected Instance and set the flags. This is done in order to avoid: 1) The leaf is an external router that passes a packet that it did not generate and that carries an unrelated RPI and 2) The leaf is an attacker or presents misconfiguration and tries to inject traffic in a protected instance. Also, this applies in the case where the leaf is aware of the RPL instance and passes a correct RPI; the 6LR needs a configuration that allows that leaf to inject in that instance.

The RH3 and RPIs could be abused by an attacker inside of the network to route packets on non-obvious ways, perhaps eluding observation. This usage appears consistent with a normal operation of [RFC6997] and can not be restricted at all. This is a feature, not a bug.

[RFC7416] deals with many other threats to LLNs not directly related to the use of IPv6-in-IPv6 headers, and this document does not change that analysis.

Nodes within the LLN can use the IPv6-in-IPv6 mechanism to mount an attack on another part of the LLN, while disguising the origin of the attack. The mechanism can even be abused to make it appear that the attack is coming from outside the LLN, and unless countered, this could be used to mount a Distributed Denial Of Service attack upon nodes elsewhere in the Internet. See [DDOS-KREBS] for an example of such attacks already seen in the real world.

If an attack comes from inside of LLN, it can be alleviated with SAVI (Source Address Validation Improvement) using [RFC8505] with

[I-D.ietf-6lo-ap-nd]. The attacker will not be able to source traffic with an address that is not registered, and the registration process checks for topological correctness. Notice that there is an L2 authentication in most of the cases. If an attack comes from outside LLN IPv6-in- IPv6 can be used to hide inner routing headers, but by construction, the RH3 can typically only address nodes within the LLN. That is, an RH3 with a CmprI less than 8 , should be considered an attack (see RFC6554, section 3).

Nodes outside of the LLN will need to pass IPv6-in-IPv6 traffic through the RPL root to perform this attack. To counter, the RPL root SHOULD either restrict ingress of IPv6-in-IPv6 packets (the simpler solution), or it SHOULD walk the IP header extension chain until it can inspect the upper-layer-payload as described in [RFC7045]. In particular, the RPL root SHOULD do [BCP38] processing on the source addresses of all IP headers that it examines in both directions.

Note: there are some situations where a prefix will spread across multiple LLNs via mechanisms such as the one described in [I-D.ietf-6lo-backbone-router]. In this case the BCP38 filtering needs to take this into account, either by exchanging detailed routing information on each LLN, or by moving the BCP38 filtering further towards the Internet, so that the details of the multiple LLNs do not matter.

13. Acknowledgments

This work is done thanks to the grant given by the StandICT.eu project.

A special BIG thanks to C. M. Heard for the help with the Section 4. Much of the redaction in that section is based on his comments.

Additionally, the authors would like to acknowledge the review, feedback, and comments of (alphabetical order): Dominique Barthel, Robert Cragie, Simon Duquennoy, Ralph Droms, Cenk Guendogan, Rahul Jadhav, Benjamin Kaduk, Matthias Kovatsch, Gustavo Mercado, Subramanian Moonesamy, Marcela Orbiscay, Charlie Perkins, Cristian Perez, Alvaro Retana, Peter van der Stok, Xavier Vilajosana, Eric Vyncke and Thomas Watteyne.

14. References

14.1. Normative References

[BCP38] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/bcp38>.

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

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

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

[RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <https://www.rfc-editor.org/info/rfc6553>.

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

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <https://www.rfc-editor.org/info/rfc7045>.

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

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

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

14.2. Informative References

[DDOS-KREBS]
Goodin, D., "Record-breaking DDoS reportedly delivered by >145k hacked cameras", September 2016, <http://arstechnica.com/security/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/>.

[I-D.ietf-6lo-ap-nd]
Thubert, P., Sarikaya, B., Sethi, M., and R. Struik, "Address Protected Neighbor Discovery for Low-power and Lossy Networks", draft-ietf-6lo-ap-nd-23 (work in progress), April 2020.

[I-D.ietf-6lo-backbone-router]
Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-20 (work in progress), March 2020.

[I-D.ietf-6tisch-dtsecurity-zerotouch-join]
Richardson, M., "6tisch Zero-Touch Secure Join protocol", draft-ietf-6tisch-dtsecurity-zerotouch-join-04 (work in progress), July 2019.

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-30 (work in progress), October 2020.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-45 (work in progress), November 2020.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.ietf-roll-unaware-leaves]
Thubert, P. and M. Richardson, "Routing for RPL Leaves", draft-ietf-roll-unaware-leaves-29 (work in progress), January 2021.

[RFC0801] Postel, J., "NCP/TCP transition plan", RFC 801, DOI 10.17487/RFC0801, November 1981, <https://www.rfc-editor.org/info/rfc801>.

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

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

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

[RFC5406] Bellovin, S., "Guidelines for Specifying the Use of IPsec Version 2", BCP 146, RFC 5406, DOI 10.17487/RFC5406, February 2009, <https://www.rfc-editor.org/info/rfc5406>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

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

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

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

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

[RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <https://www.rfc-editor.org/info/rfc8180>.

[RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <https://www.rfc-editor.org/info/rfc8504>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

Authors' Addresses

Maria Ines Robles
Universidad Tecno. Nac.(UTN)-FRM, Argentina/ Aalto University Finland

Email: mariainesrobles@gmail.com

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/mcr/

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

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

ROLL WG
INTERNET-DRAFT
Intended Status: Informational
Expires: August 8, 2016

R. Jadhav
R. Sahoo
Z. Cao
Huawei Tech
H. Deng
China Mobile

February 25, 2016

No-Path DAO Problem Statement
draft-jadhav-roll-no-path-dao-ps-00

Abstract

This document describes the problems associated with the use of No-Path DAO messaging in RPL.

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

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.

[RFC6550] 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. Current No-Path DAO messaging

[RFC6550] 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:

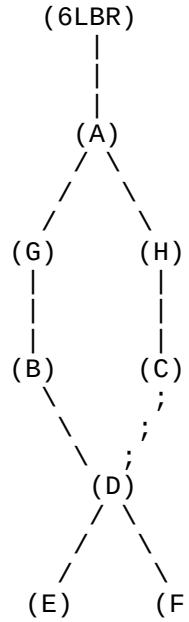


Figure 1: Sample Topology

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).

1.2. 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.

1.3. 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.

1.4. 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].

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].

2. Problems with current No-Path DAO messaging

We will confront the following problems when using the 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. [RFC6550]

assumes communication link with the previous parent for No-Path DAO messaging.

[RFC6550] 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: Support of removal of entries to the dependent nodes of the switching node.

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 disruption of downstream reachability to the node while sending NP-DAO.

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. Security Considerations

This draft is a problem statement, and therefore, does not introduce any new security risks.

5. IANA Considerations

Not applicable to this document.

6. References

6.1. 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.

[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.

[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.

6.2. Informative References

[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, May 2009.

Authors' Addresses

Rahul Arvind Jadhav
Huawei Tech,
Kundalahalli Village,
Bangalore, India

EMail: rahul.jadhav@huawei.com

Rabi Narayan Sahoo
Huawei Tech,
Kundalahalli Village,
Bangalore, India

EMail: rabinarayans@huawei.com

Zhen Cao
Huawei Tech,
Beijing, China

EMail: zhen.cao@huawei.com

Hui Deng
China Mobile,
Beijing, China

EMail: denghui@chinamobile.com

ROLL
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2017

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

November 15, 2016

Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs)
draft-satish-roll-aodv-rpl-03

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 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 May 19, 2017.

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].

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 DIO object of RREQ option MUST be always an odd number.

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

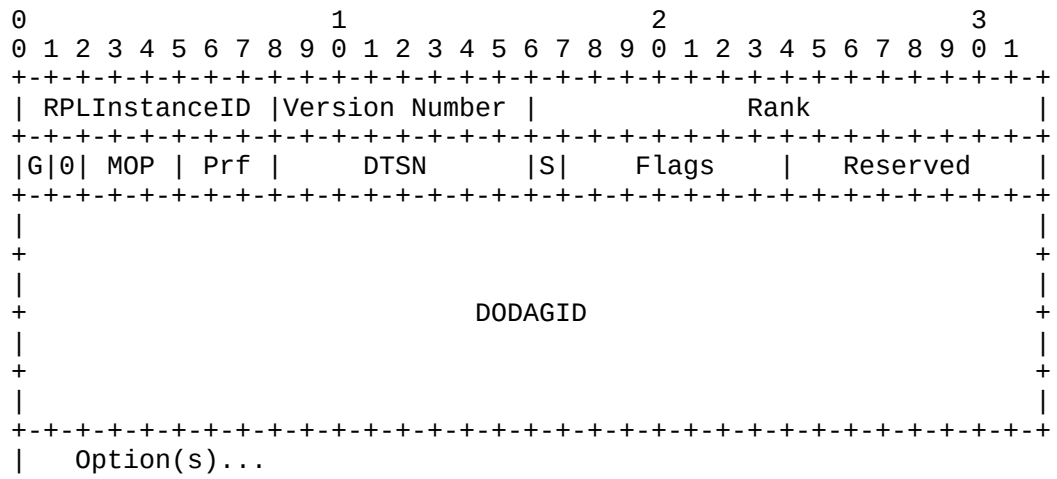


Figure 1: DIO modification to support asymmetric route discovery

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 this figure:
S := OrigNode; R := Intermediate nodes; D := TargNode

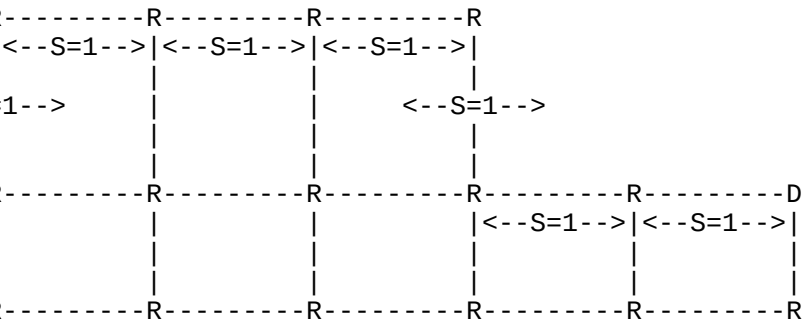


Figure 2: AODV-RPL with Symmetric Paired Instances

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 not specified in this document.

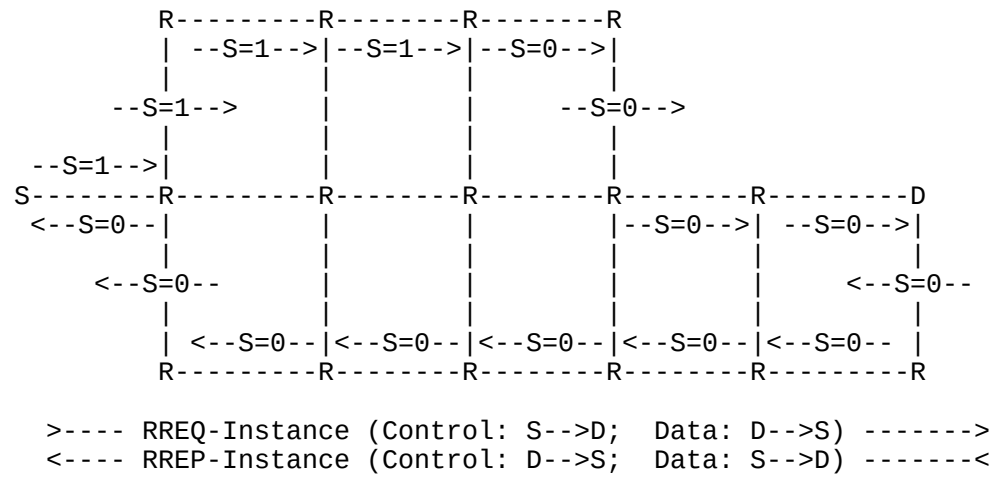


Figure 3: AODV-RPL with Asymmetric Paired Instances

5. RREQ Message

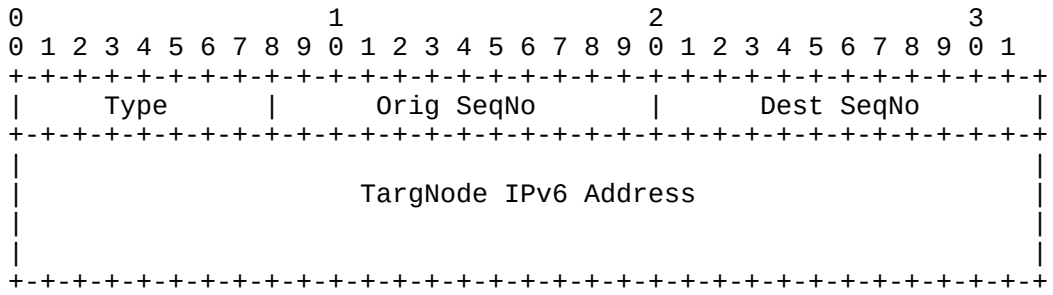


Figure 4: DIO RREQ option format for AODV-RPL MoP

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. This address MUST be in the RREQ option (see Figure 4) of AODV-RPL.

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:

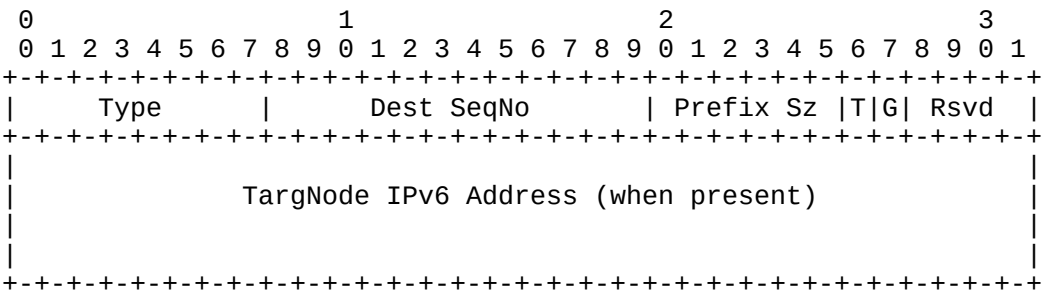


Figure 5: DIO RREP option format for AODV-RPL MoP

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 odd 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 InstanceID of RREP-Instance is even number then the TargNode IPv6 Address is elided in RREP option. When the InstanceID of RREP-Instance is an odd number with "T" bit set to "1" then TargNode IPv6 Address is transmitted in 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 application data to TargNode along the path as discovered through RREP messages. Similarly, 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. References

11.1. 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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6998>.

11.2. Informative References

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

Authors' Addresses

Satish Anamalamudi
Huawei Technologies
No. 156 Beiqing Rd. Haidian District
Beijing 100095
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: rashid.sangi@huawei.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: April 17, 2017

P. van der Stok
consultant
AR. Sangi
Huawei Technologies

October 14, 2016

MPL Forwarder Select (MPLFS)
draft-vanderstok-roll-mpl-forw-select-01

Abstract

This document describes a Forwarder Selection (MPLFS) protocol for the Multicast Protocol for Low-Power and lossy Networks (MPL) to reduce the density of forwarders such that the number of forwarded messages is reduced. The protocol uses Trickle to distribute link-local information about the identity of the neighbours of the nodes that have MPL-enabled interfaces. In the end-state all nodes are connected to a minimum number, N_DUPLICATE, of forwarders, where N_DUPLICATE is application dependent, and there is a path between any two forwarders.

Note

Discussion and suggestions for improvement are requested, and should be sent to roll@ietf.org.

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 April 17, 2017.

Table of Contents

1. Introduction

The Multicast Protocol for Low-Power and Lossy Networks (MPL) [RFC7731] is designed for small devices interconnected by a lossy wireless network such as IEEE 802.15.4. A seed sends a multicast message with a realm-local scope, admin-local scope or higher as specified in [RFC4291].

Forwarders forward these messages with an increasing interval size. When the density of forwarders is high, the message may be forwarded by a high number of forwarders that conflict on the link. With extreme forwarder densities and small Trickle intervals, just sending one multicast message may lead to an overload of the communication medium.

The number of forwarded messages can be reduced by selecting a minimal set of forwarders. However, for large networks, manually selecting the forwarders is much work, and changing network conditions and configurations make the manual selection an unwanted burden to the network management.

This document specifies a protocol that selects the forwarders such that each MPL-enabled device is connected to N_DUPLICATE forwarders, where N_DUPLICATE > 0 can be set. The parameter N_DUPLICATE determines how much path redundancy there is for each MPL message. The value of N_DUPLICATE should be at least 1, because a value of 0 has as result that no forwarder exists in the network during the protocol execution. Moreover, the protocol is distributed and dynamic in nature to face a continuously changing topology.

The protocol is inspired by the work described for NeighbourHood Discovery (NHDP) [RFC6130] and Simple Multicast Forwarding (SMF) [RFC6621]. In contrast to the "HELLO" messages described in [RFC6130], this protocol uses the Trickle protocol [RFC6206] to multicast link-local messages, containing a CBOR payload [RFC7049].

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 [RFC2119].

Readers of this specification should be familiar with all the terms and concepts discussed in [RFC7731]. The following terms are defined in this document:

synchronization time The moment that a node can change its state at messages reception.

The following list contains the abbreviations used in this document.

XXXX: TODO, and others to follow.

2. Protocol overview

Nodes participating in MPLFS exchange messages with a format that is described in Section 6. A participating node communicates to all its neighbours with link-local multicast messages as described in Section 4.

Failing links provide a lot of instability. Only messages sent over stable links are accepted. Section 4 describes a mechanism to refuse messages from unstable links.

Each node maintains a set of 1-hop neighbours where each neighbour contains information about its own 1-hop neighbours. On the basis of the contents of the set, the node can decide to become a Forwarder or not, as explained in Section 5.

The protocol never ends, with a minimum frequency of exchanging maintenance messages specified by an interval size of I_MAX_SELECT. When the set of links is stable, the protocol stabilizes such that there is a path between any two forwarders, and every MPL-enabled node is connected to at least N_DUPLICATE MPL forwarders (when existing), where N_DUPLICATE > 0. N_DUPLICATE can be set dependent on the application requirements. With N_DUPLICATE = 2, it is expected that a multicast message arrives at an intended recipient with very high probability.

Nodes have a state that determines whether they are forwarder or not. The state of a node can only be changed by the node itself. To avoid race conditions, (e.g. two nodes simultaneously decide to be no forwarder, while only one is intended) the node with the highest address of all 1-hop neighbours is the only one allowed to change state. Unlike [RFC5614], that considers 3-tuple (Router Priority, MDR Level and Router ID) to allow self state change, this approach only takes into account the node address. Consequently, only k-hop neighbours, with k > 2, can change state simultaneously, and the 1-hop and 2-hop neighbours of a given node can change state one by one.

3. Data sets

Each node, n_0, maintains a state with two values: Fixed Forwarder (FF) and No Forwarder (NF). Each node also maintains a set, S1_0, containing information about n_0’s 1-hop neighbours and n_0 itself. Each entry, n_i, in S1_0 has the following attributes:

address of n_i: the address can be the 64 bit IPv6 address or the short 16 bit address.

average-rssi-in: the average rssi of the messages received by n_0 from n_i.

average-rssi-out: the average rssi of the messages received by n_i from n_0.

nr_FF: the number of neighbours, n_ij, of n_i (including n_i) with state = FF.

nr_Under: the number of neighbours, n_ij, of n_i with nr_FF < N_DUPLICATE.

nr_Above: the number of neighbours, n_ij, of n_i with nr_FF > N_DUPLICATE.

size: the size of S1_i, the set of 1-hop neighbours of n_i.

state: the state of n_i.

4. Neighbor distribution

A participating node multicasts link-local so-called "neighbour messages" with the Trickle protocol. It uses the multicast address LINK_LOCAL_ALL_NODES as destination. The message sent by n_0 contains the contents of S1_0. The contents of a "neighbour message" from n_i received by n_0 is called M_i. The rssi value associated with the reception of the "neighbour message" is called new_rssi. The message M_i contains information about the set S1_i with the following attributes for all nodes in S1_i:

- address
- average-rssi-in
- nr_FF
- nr_Under
- nr_Above
- size
- state

On reception of M_i from n_i for the first time, the receiving node adds n_i to S1_0, and sets average-rssi-in of n_i in S1_0 to new_rssi. For all following messages from n_i, the average-rssi-in for n_i is calculated in the following way: average-rssi-in := (average-rssi-in*WEIGHT_AVERAGE + new_rssi)/(WEIGHT_AVERAGE+1).

The neighbour nodes of M_i are called n_ij. For the n_ij with an address that is equal to the address of n_0: the value of average-rssi-out of n_0 is set equal to the value of average-rssi-in of n_ij.

The contents of n_0 is updated with the contents of M_i. Updating includes the following actions:

- Add n_i to S1_0, if n_i not present in S1_0.
- Set size of n_i equal to the number of entries in M_i.

- When n_ij.address = n_j.address, copy the values of nr_Under, nr_Above, nr_FF, and state of n_ij to n_j.

When the average-rssi-in and average-rssi-out values of n_i have been averaged over more than WEIGHT_AVERAGE messages, and the averaged RSSI values are smaller than MAXIMUM_RSSI, n_i is called "valid".

5. Selection Algorithm

The protocol aims at allocating forwarders in the densest part of the network. A dense network is characterized by a high number of neighbours. Therefore, the protocol attempts to assign status FF to the nodes with the highest number of neighbours that have less than N_DUPLICATE neighbours with state = FF (nr_FF < N_DUPLICATE).

It is required that a path exists between every two forwarders to prevent network partitioning. Therefore, a node can become forwarder iff one of its neighbours is a forwarder. The consequence of this rule is that one so-called "source-forwarder" must be selected by the network administrator. A likely choice for the "source-forwarder" is the border router.

At the start of the selection protocol the node, n_0, sets its state to No Forwarder (NF). It sets the Trickle timer to its minimum interval, I_MIN_SELECT, and starts multicasting M_0 to its neighbours. Every time entries are added to, or removed from, S1_0, the Trickle interval timer is set to I_MIN_SELECT.

The executing node, n_0, calculates the following parameter values:

- max-under is the maximum of the nr_Under attribute of all valid n_i in S1_0.
- max_address_u is the maximum of the addresses of valid n_i with nr_Under = max-under.
- max_address_a is the maximum of the addresses of all valid n_i.
- connected is true iff nr_FF of all neighbouring forwarders is equal to nr_FF of n_0.

The information about the state and the nr_Under value of the neighbours comes in asynchronously. Time is needed before the state in a node correctly reflects the state changes of the network. A node can change its state when during the reception of messages of all neighbours, the value of nr_Under has not changed.

To calculate its new state, n_0 does the following:

When the state is NF, a neighbour with state = FF exists, and address = max-address_u:
set state to FF.

When the state is FF, nr_Above = size S1_0, connected is true, and address = max_address_a:
set state to NF.

6. CBOR payload

The payload format is /application/cbor [RFC7049]. The contents of the message is a CBOR array (Major type 4) of CBOR arrays composed of neighbour address, rssi value, size of S1_i, forwarder state, nr_FF, nr_Under, and nr_Above. Assuming two neighbours, in diagnostic JSON the payload looks like:

```
[
[address_1, average-rssi-in_1, size_1, state_1,
    nr_FF_1, nr_Under_1, nr_Above_1],
[address_2, average-rssi-in_2, size_2, state_2,
    nr_FF_2, nr_Under_2, nr_Above_2]
]
```

Figure 1: CBOR payload

7. Default parameter values

The following text recommends default values for the MPLFS protocols.

I_MIN_SELECT = 0,2; minimum Trickle timer interval.

I_MAX_SELECT = 10; maximum Trickle timer interval.

WEIGHT_AVERAGE = 10; number of values to average rssi.

MAXIMUM_RSSI = 3; maximum acceptable average rssi value.

N_DUPLICATE = 2; requested number of MPL forwarder neighbours for every MPL enabled node.

8. Acknowledgements

We are very grateful to

9. Changelog

Changes from version 00 to version 01

- Definition of S1_0 improved
- Algorithm changed and simulated
- Moment of state change specified

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, <http://www.rfc-editor.org/info/rfc2119>.

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

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7731] Hui, J. and R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)", RFC 7731, DOI 10.17487/RFC7731, February 2016, <http://www.rfc-editor.org/info/rfc7731>.

10.2. Informative References

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5614] Ogier, R. and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding", RFC 5614, DOI 10.17487/RFC5614, August 2009, <http://www.rfc-editor.org/info/rfc5614>.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, DOI 10.17487/RFC6130, April 2011, <http://www.rfc-editor.org/info/rfc6130>.

[RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <http://www.rfc-editor.org/info/rfc6621>.

Authors' Addresses

Peter van der Stok
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI: www.vanderstok.org

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

Email: rashid.sangi@huawei.com