ROLL
Internet-Draft
Intended status: Standards Track
Expires: October 24, 2018

R. Jadhav, Ed.
R. Sahoo
Y. Wu
Huawei

April 22, 2018

RPL Observations
draft-rahul-roll-rpl-observations-01

Abstract

This document describes RPL protocol design issues, various observations and possible consequences of the design and implementation choices.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 24, 2018.

Table of Contents

1. Introduction 2
1.1. Requirements Language and Terminology 3
2. DTSN increment in storing MOP 3
2.1. Deliberations 4
3. DAO retransmission and use of DAO-ACK in storing MOP 5
3.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK 5
3.2. Problems with hop-by-hop DAO-ACK 6
3.3. Problems with end-to-end DAO-ACK 6
3.4. Deliberations 6
3.5. Implementation Notes 6
4. Handling resource unavailability 7
4.1. Deliberations 7
5. Handling aggregated targets 7
5.1. Deliberations 8
6. RPL Transit Information in DAO 8
6.1. Deliberations 8
7. Managing persistent variables across node reboots 9
7.1. Persistent storage and RPL state information 9
7.2. Lollipop Counters 9
7.3. RPL State variables 10
7.3.1. DODAG Version 10
7.3.2. DTSN field in DIO 11
7.3.3. PathSequence 11
7.4. State variables update frequency 11
7.5. Deliberations 12
7.6. Implementation Notes 12
8. RPL under-specification 12
9. Acknowledgements 13
10. IANA Considerations 13
11. Security Considerations 13
12. References 13
12.1. Normative References 13
12.2. Informative References 14
Appendix A. Additional Stuff 14
Authors' Addresses 14

1. Introduction

RPL [RFC6550] specifies a proactive distance-vector routing scheme designed for LLNs (Low Power and Lossy Networks). RPL enables the network to be formed as a DODAG and supports storing mode and non-storing mode of operations. Non-storing mode allows reduced memory resource usage on the nodes by allowing non-BR nodes to operate without managing a routing table and involves use of source routing

by the 6LBR to direct the traffic along a specific path. In storing mode of operation intermediate routers maintain routing tables.

This work aims to highlight various issues with RPL which makes it difficult to handle certain scenarios. This work will highlight such issues in context to RPL's mode of operations (storing versus non-storing). There are cases where RPL does not provide clear rules and implementations have to make their choices hindering interoperability and performance.

[I-D.clausen-lln-rpl-experiences] provides some interesting points. Some sections in this draft may overlap with some observations in [clausen], but this is been done to further extend some scenarios or observations. It is highly encouraged that readers should also visit [I-D.clausen-lln-rpl-experiences] for other insights. Regardless, this draft is self-sufficient in a way that it does not expect to have read [clausen-draft].

1.1. Requirements Language and Terminology

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

NS-MOP = RPL Non-storing Mode of Operation

S-MOP = RPL Storing Mode of Operation

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

2. DTSN increment in storing MOP

DTSN increment has major impact on the overall RPL control traffic and on the efficiency of downstream route update. DTSN is sent as part of DIO message and signals the downstream nodes to trigger the target advertisement. The 6LR needs to decide when to update the DTSN and usually it should do it in a conservative way. The DTSN update mechanism determines how soon the downward routes are established along the new path. RPL specifications does not provide any clear mechanism on how the DTSN update should happen in case of storing mode.

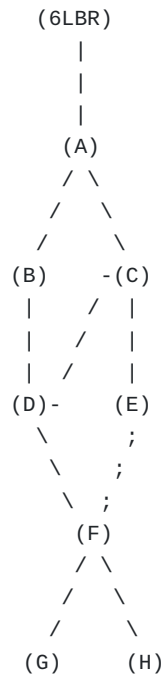

Figure 1: Sample topology

Consider example topology shown in Figure 1, assume that node D switches the parent from node B to C. Ideally the downstream nodes D and its sub-childs should send their target advertisement to the new path via node C. To achieve this result in a efficient way is a challenge. Incrementing DTSN is the only way to trigger the DAO on downstream nodes. But this trigger should be sent not only on the first hop but to all the grand-child nodes. Thus DTSN has to be incremented in the complete sub-DODAG rooted at node D thus resulting in DIO/DAO storm along the sub-DODAG. This is specifically a big issue in high density networks where the metric deteoration might happen transiently even though the signal strength is good.

The primary implementation issue is whether a child node increment its own DTSN when it receives DTSN update from its parent node? This would result in DAO-updates in the sub-DODAG, thus the cost could be very high. If not incremented it may result in serious loss of connectivity for nodes in the sub-DODAG.

2.1. Deliberations

(1) In S-MOP, should the child nodes increment its DIO on seeing that its preferred parent has updated its DTSN?

(2) What are rules for DTSN increment for storing MOP, which multiple implementations can follow thus allowing consistent performance across different implementations?

3. DAO retransmission and use of DAO-ACK in storing MOP

[RFC6550] has an optional DAO-ACK mechanism using which an upstream parent confirms the reception of a DAO from the downstream child. In case of storing mode, the DAO is addressed to the immediate hop upstream parent resulting in DAO-ACK from the parent. There are two implementations possible:

(1) Hop-by-hop ACK: A parent responds with a DAO-ACK immedetialy after receiving the DAO.

(2) End-to-End ACK: A node waits for the upstream parent to send DAO-ACK to respond with a DAO-ACK downstream. The upstream parent may do as many attempts to successfully send this DAO upstream. In other words, the parent node accepts the responsibilty of sending the DAO upstream till the point it is ACKed the moment it responds back with its own ACK to the child.

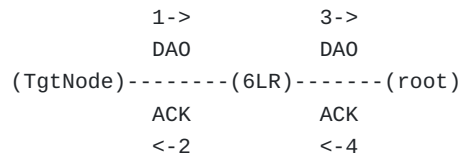

Figure 2: Hop-by-hop DAO-ACK

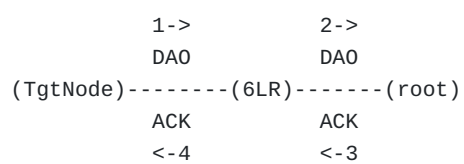

Figure 3: End-to-End DAO-ACK

3.1. Significance of bidirectional Path establishment indication and relevance of DAO-ACK

Lot of application traffic patterns requires that the bidirectional path be established between the target node and the root. A typical example is that COAP request with ACK bit set would require an acknowledgement from the end receiver and thus warrants bidirectional path establishment. It is imperative that the target node first ascertains whether such a bidirectional path is established before initiating such application traffic. In case of non-storing MOP, the

DAO-ACK works perfectly fine to ascertain such bidirectional connectivity since it is an indication that the root which usually is the direct destination of the DAO has received the DAO. But in case of storing MOP, things are more complicated since DAO is sent hop-by-hop and the DAO-ACK semantics are not clear enough as per the current specification. As mentioned in above section, an implementation can choose to implement hop-by-hop ACK or end-to-end ACK.

3.2. Problems with hop-by-hop DAO-ACK

The primary issue with this mode is that target node cannot ascertain bidirection path connectivity on the reception of the DAO-ACK.

3.3. Problems with end-to-end DAO-ACK

In this case, it is possible for the target node to ascertain if the DAO has indeed reached the root since the reception of DAO-ACK on target node confirms this. However there is extra state information that needs to be maintained on the 6LRs on behalf of all the child nodes. Also it is very difficult for the target node to ascertain a timer value to decide whether the DAO transmission has failed to reach the root.

3.4. Deliberations

(1) How should an implementation interpret the DAO-ACK semantics?

(2) What is the best way for the target node to know that the end to end bidirectional path is successfully installed or updated? In NS-MOP, the DAO-ACK provides a clear way to do this. Can the same be achieved for storing-MOP?

(3) What happens if the DAO-ACK with Status!=0 is responded by ancestor node?

(4) How to selectively NACK subset of targets in case target containers are aggregated?

3.5. Implementation Notes

Current RPL open source implementations have both types of DAO-ACK implementations. For e.g. RIOT supports hop-by-hop DAO-ACK. Contiki older versions supported hop-by-hop ACK but the recent version have changed to end-to-end ACK implementation.

The sequence of sending no-path DAO and DAO matters when updating the routing adjacencies on a parent switch. If an implementation chooses to send no-path DAO before DAO then it results in significantly more

overhead for route invalidation. This is because no-path DAO would traverse all the way up to the BR clearing the routes on the way. In case there is a common ancestor post which the old and new path remains same then it is better to send regular DAO first thus limiting the propagation of subsequent no-path DAO till this common ancestor.

4. Handling resource unavailability

The nodes in the constrained networks have to maintain various records such as neighbor cache entries and routing entries on behalf of other targets to facilitate packet forwarding. Because of the constrained nature of the devices the memory available may be very limited and thus the path selection algorithm may have to take into consideration such resource constraints as well.

RPL currently does not have any mechanism to advertise such resource indicator metrics. The primary tables associated with RPL are routing table and the neighbor cache. Even though neighbor cache is not directly linked with RPL protocol, the maintenance of routing adjacencies results in updates to neigbor cache.

4.1. Deliberations

Is it possible to know that an upstream parent/ancestor cannot hold enough routing entries and thus this path should not be used?

Is it possible to know that an upstream parent cannot hold any more neighbor cache entry and thus this upstream parent should not be used?

5. Handling aggregated targets

RPL allows and defines specific procedures so as to aid target aggregation in DAO. Having said that, the specification does not mandate use of aggregated targets nor does it make any comment on whether a receiving node needs to handle it. Target aggregation is an useful tool and especially helps with link layer technologies that does not suffer from low MTUs such as PLC. Even if the implementation does not support aggregating targets, it should atleast mandate reception of aggregated targets in DAO.

RPL has a mechanism currently to ACK the DAO but it does not have a mechanism to ACK the target container. Thus in case of aggregated targets in the DAO, if the subset of the targets fail then it is impossible for the DAO-ACK to signal this to the DAO sender.

5.1. Deliberations

Even if the implementation does not support aggregating targets, should it atleast mandate reception and handling of aggregated targets in DAO?

There is a good scope for compressing aggregated targets which can significantly reduce the RPL control overhead.

How to selectively NACK subset of targets in case target containers are aggregated?

The DEFAULT_DAO_DELAY of 1sec does not help much with aggregation. The upstream parent nodes should wait for more time then the child nodes so as to effectively aggregate. Can we have DEFAULT_DAO_DELAY a function of the level/rank the node is at?

6. RPL Transit Information in DAO

RPL allows associating a target or set of targets with a Transit information container which contains attributes for a path to one or more destinations identified by the set of targets. In case of NS-MOP, the transit Information will contain the all critical Parent Address which allows the common ancestor usually the root to identify the source route header for the target node. The Transit Information also contains other information such as Path Sequence and Path Lifetime which are critical for maintaining route adjacencies.

RPL however does not mandate the use of Transit Information container for targets.

6.1. Deliberations

Is it ok to let implementations decide on the inclusion of Transit Information container?

Is it possible to achieve interop without mandating use of Transit Information Container?

If the Transit Information container is sent, should the handling of PathSequence be mandated?

The DEFAULT_DAO_DELAY of 1sec does not help much with aggregation. The upstream parent nodes should wait for more time then the child nodes so as to effectively aggregate. Can we have DEFAULT_DAO_DELAY a function of the level/rank the node is at?

7. Managing persistent variables across node reboots

7.1. Persistent storage and RPL state information

Devices are required to be functional for several years without manual maintanence. Usually battery power consumption is considered key for operating the devices for several (tens of) years. But apart from battery, flash memory endurance may prove to be a lifetime bottleneck in constrained networks. Endurance is defined as maximum number of erase-write cycles that a NAND/NOR cell can undergo before losing its 'gauranteed' write operation. In some cases (cheaper NAND-MLC/TLC), the endurance can be as less as 2K cycles. Thus for e.g. if a given cell is written 5 times a day, that NAND-flash cell assuming an endurance of 10K cycles may last for less than 6 years.

Wear leveling is a popular technique used in flash memory to minimize the impact of limited cell endurance. Wear leveling works by arranging data so that erasures and re-writes are distributed evenly across the medium. The memory sectors are over-provisioned so that the writes are distributed across multiple sectors. Many IoT platforms do not necessarily consider this over-provisioning and usually provision the memory only to what is required. Some scenarios such as street-lighting may not require the application layer to write any information to the persistent storage and thus the over-provisioning is often ignored. In such cases if the network stack ends up using persistent storage for maintaining its state information then it becomes counter-productive.

In a star topology, the amount of persistent data write done by network protocols is very limited. But ad-hoc networks employing routing protocols such as RPL assume certain state information to be retained across node reboots. In case of IoT devices this storage is mostly floating gate based NAND/NOR based flash memory. The impact of loss of this state information differs depending upon the type (6LN/6LR/6LBR) of the node.

7.2. Lollipop Counters

[RFC6550] Section 7.2. explains sequence counter operation defining lollipop [Perlman83] style counters. Lollipop counters specify mechanism in which even if the counter value wraps, the algorithm would be able to tell whether the received value is the latest or not. This mechanism also helps in "some cases" to recover from node reboot, but is not foolproof.

Consider an e.g. where Node A boots up and initialises the seqcnt to 240 as recommended in [RFC6550]. Node A communicates to Node B using this seqcnt and node B uses this seqcnt to determine whether the

information node A sent in the packet is latest. Now lets assume, the counter value reaches 250 after some operations on Node A, and node B keeps receiving updated seqcnt from node A. Now consider that node A reboots, and since it reinitializes the seqcnt value to 240 and sends the information to node B (who has seqcnt of 250 stored on behalf of node A). As per section 7.2. of [RFC6550], when node B receives this packet it will consider the information to be old (since 240 < 250).

A	B	Output
240	240	A<B, old
240	241	A<B, old
240	::	A<B, old
240	256	A<B, old
240	0	A<B, new
240	1	A>B, new
240	::	A>B, new
240	127	A>B, new

Default values for lollipop counters considered from [RFC6550] Section 7.2.

Table 1: Example lollipop counter operation

Based on this figure, there is dead zone (240 to 0) in which if A operates after reboot then the seqcnt will always be considered smaller. Thus node A needs to maintain the seqcnt in persistent storage and reuse this on reboot.

7.3. RPL State variables

The impact of loss of RPL state information differs depending upon the node type (6LN/6LR/6LBR). Following sections explain different state variables and the impact in case this information is lost on reboot.

7.3.1. DODAG Version

The tuple (RPLInstanceID, DODAGID, DODAGVersionNumber) uniquely identifies a DODAG Version. DODAGVersionNumber is incremented everytime a global repair is initiated for the instance (global or local). A node receiving an older DODAGVersionNumber will ignore the DIO message assuming it to be from old DODAG version. Thus a 6LBR node (and 6LR node in case of local DODAG) needs to maintain the DODAGVersionNumber in the persistent storage, so as to be available

on reboot. In case the 6LBR could not use the latest DODAGVersionNumber the implication are that it won't be able to recover/re-establish the routing table.

7.3.2. DTSN field in DIO

DTSN (Destination advertisement Trigger Sequence Number) is a DIO message field used as part of procedure to maintain Downward routes. A 6LBR/6LR node may increment a DTSN in case it requires the downstream nodes to send DAO and thus update downward routes on the 6LBR/6LR node. In case of RPL NS-MOP, only the 6LBR maintains the downward routes and thus controls this field update. In case of S-MOP, 6LRs additionally keep downward routes and thus control this field update.

In S-MOP, when a 6LR node switches parent it may have to issue a DIO with incremented DTSN to trigger downstream child nodes to send DAO so that the downward routes are established in all parent/ancestor set. Thus in S-MOP, the frequency of DTSN update might be relatively high (given the node density and hysteresis set by objective function to switch parent).

7.3.3. PathSequence

PathSequence is part of RPL Transit Option, and associated with RPL Target option. A node whichs owns a target address can associate a PathSequence in the DAO message to denote freshness of the target information. This is especially useful when a node uses multiple paths or multiple parents to advertise its reachability.

Loss of PathSequence information maintained on the target node can result in routing adjacencies been lost on 6LRs/6LBR/6BBR.

7.4. State variables update frequency

State variable	Update frequency	Impacts node type
DODAGVersionNumber	Low	6LBR, 6LR(local DODAG)
DTSN	High(SM),Low(NSM)	6LBR, 6LR
PathSequence	High(SM),Low(NSM)	6LR, 6LN

Low=<5 per day, High=>5 per day; SM=Storing MOP, NSM=Non-Storing MOP

Table 2: RPL State variables

7.5. Deliberations

(1) Is it possible that RPL reduces the use of persistent storage for maintaining state information?

(2) In most cases, the node reboots will happen very rarely. Thus doing a persistent storage book-keeping for handling node reboot might not make sense. Is it possible to consider signaling (especially after the node reboots) so as to avoid maintaining this persistent state? Is it possible to use one-time on-reboot signalling to recover some state information?

(3) It is necessary that RPL avoids using persistent storage as far as possible. Ideally, extensions to RPL should consider this as a design requirement especially for 6LR and 6LN nodes. DTSN and PathSequence are the primary state variables which have major impact.

7.6. Implementation Notes

An implementation should use a random DAOSequence number on reboot so as to avoid a risk of reusing the same DAOSequence on reboot. Regardless the sequence counter size of 8bits does not provide much gurantees towards choosing a good random number. A parent node will not respond with a DAO-ACK in case it sees a DAO with the same previous DAOSequence.

Write-Before-Use: The state information should be written to the flash before using it in the messaging. If it is done the other way, then the chances are that the node power downs before writing to the persistent storage.

8. RPL under-specification

(a) PathSequence: Is it mandatory to use PathSequence in DAO Transit container? RPL mentions that a 6LR/6LBR hosting the routing entry on behalf of target node should refresh the lifetime on reception of a new Path Sequence. But RPL does not necessarily mandate use of Path Sequence. Most of the open source implementation [RIOT] [CONTIKI] currently do not issue Path Sequence in the DAO message.

(b) Target Container aggregation in DAO: RPL allows multiple targets to be aggregated in a single DAO message and has introduced a notion of DelayDAO using which a 6LR node could delay its DAO to enable such aggregation. But RPL does not have clear text on handling of aggregated DAOs and thus it hinders interoperability.

(c) DTSN Update: RPL does not clearly define in which cases DTSN should be updated in case of storing mode of operation. More details for this are presented in Section 2.

9. Acknowledgements

Many thanks to Pascal Thubert for hallway chats and for helping understand the existing design rationales. Thanks to Michael Richardson for Unstrung RPL implementation rationale. Thanks to ML discussions, in particular (https://www.ietf.org/mail-archive/web/roll/current/msg09443.html).

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

This is an information draft and does add any changes to the existing specifications.

12. References

12.1. Normative References

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

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

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

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

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

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

12.2. Informative References

[I-D.clausen-lln-rpl-experiences]
Clausen, T., Verdiere, A., Yi, J., Herberg, U., and Y. Igarashi, "Observations on RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-clausen-lln-rpl-experiences-11 (work in progress), March 2018.

[Perlman83]
Perlman, R., "Fault-Tolerant Broadcast of Routing Information", North-Holland Computer Networks, Vol.7, December 1983.

Appendix A. Additional Stuff

Authors' Addresses

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

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

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

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

Yuefeng Wu
Huawei
No.101, Software Avenue, Yuhuatai District,
Nanjing, Jiangsu 210012
China

Phone: +86-15251896569
Email: wuyuefeng@huawei.com