ROLL
Internet-Draft
Intended status: Standards Track
Expires: June 3, 2021

R. Jadhav, Ed.

R. Sahoo
Juniper

Y. Wu
Huawei

November 30, 2020

RPL Observations
draft-ietf-roll-rpl-observations-05

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 June 3, 2021.

Copyright Notice

Table of Contents

1. Motivation

The primary motivation for this draft is to enlist different issues with RPL operation and invoke a discussion within the working group. This draft by itself is not intended for RFC tracks but as a WG discussion track. This draft may in turn result in other work items taken up by the WG which may improvise on the issues mentioned herewith.

2. 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 Root 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].

2.1. 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].

3. 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.

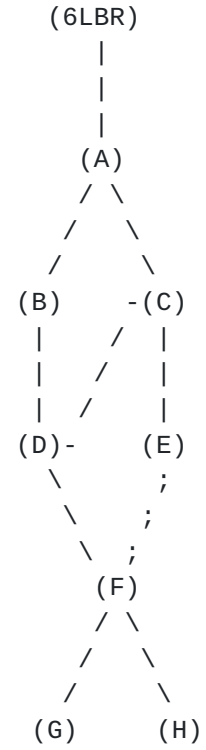

Figure 1: Sample topology

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.

3.1. Deliberations

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

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

4. 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.

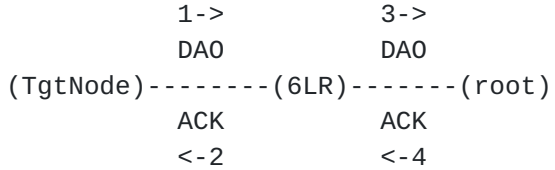

Figure 2: Hop-by-hop DAO-ACK

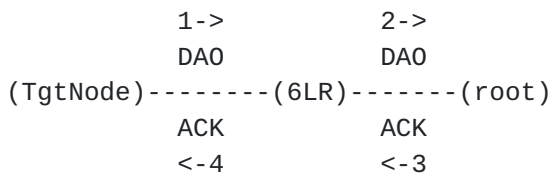

Figure 3: End-to-End DAO-ACK

4.1. 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.

4.2. 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.

4.3. 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.

4.4. 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 options are aggregated?

4.5. 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.

5. Interpreting Trickle Timer

Trickle algorithm defines a mechanism to reset the timer. Trickle timer reset is unlike regular periodic timers wherein the timer is simply reset to start again. Reset of trickle timer implies resetting the trickle back to Imin and starting with a new interval as mentioned in Section 4.2 of [RFC6206].

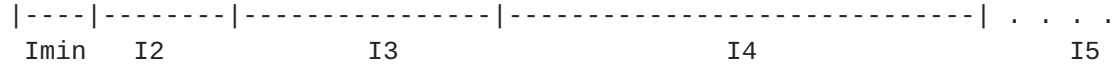

Figure 4: Trickle Timer Operation

The above figure shows an example of trickle intervals. An interval is double that of the previous interval size. Section 4.2. of [RFC6206] states that,

"If Trickle hears a transmission that is "inconsistent" and I is greater than Imin, it resets the Trickle timer. To reset the timer, Trickle sets I to Imin and starts a new interval as in step 2. If I is equal to Imin when Trickle hears an "inconsistent" transmission,

Trickle does nothing. Trickle can also reset its timer in response to external "events"."

Thus if the trickle timer has advanced to subsequent intervals i.e., >= I2, then a reset of trickle timer implies going back to Imin. However, if the trickle timer is currently in Imin and if it hears an inconsistent transmission then it does nothing.

In context to multicast DIS/DIO operation, this implies that if the DIO trickle timer is already at Imin and if the node hears a multicast DIS, then the timer does nothing. It MUST NOT reset the timer again in this case.

An implementation MUST never restart the timer within an interval. For e.g., in the above figure, if the timer is in interval I2, the implementation MUST never restart the timer to the beginning of the current interval i.e., I2. If the timer is in interval T2 and if the reset is to be done then the interval is set back to Imin. If the timer is already in Imin, then the reset should do nothing.

6. 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.

6.1. 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?

7. 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 option. 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.

7.1. 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 options 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?

8. RPL Transit Information in DAO

RPL allows associating a target or set of targets with a Transit Information Option 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 Option for targets.

8.1. Deliberations

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

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

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

9. Upgrades or Extensions to RPL protocol

RPL extensibility is highly desirable and is controlled by protocol elements within the messaging framework. In the pursuit to keep the signalling overhead less, RPL specification has been restricting in its approach to extend its field ranges, thus in some cases putting extensibility at stakes. Consider for example, the mode of operation bits which is three bits in the RPL specification. These bits are already saturated and it may be difficult to add major upgrades without extending these bits.

Addition of new Control Options or new RPL Codes almost certainly results in backward compatibility issues. RFC6550 clearly mentions that a message with an unknown RPL Code MUST be silently discarded. However, no explicit handling is suggested for unknown RPL control option types. In some cases, implementations simply copy-forward an unknown option as it is while in other cases the unknown option is stripped off before forwarding the message.

Deliberations:

(1) What are the extensibility options RPL could implement? How much overhead would it incur?

(2) Most of the extensions are in the form of new control options. Should RPL have a mechanism to only handle such extensions in a backward compatible but in a generic manner?

10. Path Control bits handling

RPL uses Path Control bits in the DAO's Transit Information Option for installing multiple downward routes to the nodes. These multiple routes could be used for reliability, latency or traffic load-balancing within a DAG. The path control bits are usable both in storing and non-storing mode of operation.

RFC6550 Section 9.9 bullet point 9 requires a mandatory setting of Path Control bits in all the unicast DAOs sent by the Target node. However, no existing implementation of RPL supports this. There is no reason for a network which only requires a single path to the root to mandatorily support path control bits.

Deliberations:

(1) Should the mandatory clause for supporting Path Control Bits in RFC6550 Section 9.9 point 9 be removed?

(2) Handling Path Control Bits may be complex. An implementation guideline explaining the use-cases and resource (memory requirements) assumptions would help implementors decide the utility of this technique.

11. Asymmetric Links and RPL

Section 3.1 of [I-D.ietf-intarea-adhoc-wireless-com] explains asymmetric link characteristics and what it takes for a protocol to support asymmetric links. RPL depends on bi-directional links for control even though near-perfect symmetry is not expected. The implication of this is that the upstream and downstream path remains same within a given RPL instance for any pair of nodes. There are following questions sprouting of this design:

(1) Is it possible to detect asymmetric links?

(2) In the presence of asymmetric links what is the impact on the control overhead and is there a way to possibly mitigate or alleviate any negative impact?

[I-D.ietf-roll-aodv-rpl] defines a mechanism to use a pair of instances which are coupled. This allows disjoint upstream and downstream paths between pair of nodes assuming that the link asymmetricity is detected using some outside techniques. The link assumes that the link asymmetricity is already known to the nodes in the form of static configuration. In case of 6tisch networks, the availability of transmission slots information can be used to identify link asymmetricity. The challenge with regards to detecting link asymmetricity arises from scenarios where, for example, the nodes transmit with unequal power levels.

12. Adjacencies probing with RPL

RPL avoids periodic hello messaging as compared to other distance-vector protocols. It uses trickle timer based mechanism to update configuration parameters. This significantly reduces the RPL control

overhead. One of the fallout of this design choice is that, in the absence of regular traffic, the adjacencies could not be tested and repaired if broken.

RPL provides a mechanism in the form of unicast DIS to query a particular node for its DIO. A node receiving a unicast DIS MUST respond with a unicast DIO with Configuration Option. This mechanism could as well be made use of for probing adjacencies and certain implementations such as Contiki uses this. The periodicity of the probing is implementation dependent, but the node is expected to invoke probing only when

(1) There is no data traffic based on which the links could be tested.

(2) There is no L2 feedback. In some case, L2 might provide periodic beacons at link layer and the absence of beacons could be used for link tests.

12.1. Deliberations

(1) Should the probing scheme be standardized? In some cases using multicast based probing may prove advantageous.

(2) In some cases using multicast based probing may prove advantageous. Currently RPL does not have multicast based probing. Multicast DIS/DIO may not be suitable for probing because it could possibly lead to change of states.

13. Control Options eliding mechanism in RPL

RPL configuration changes are rare and thus various configuration options may not change over a long period of time. RPL provides a way for the configuration options to be elided but there are no clear guidelines on how the eliding should be handled. In the absence of such guidelines, it is possible that certain nodes may end up using stale configuration in the event of transient link failures.

14. Managing persistent variables across node reboots

14.1. 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.

14.2. 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.

14.3. 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.

14.3.1. 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.

14.3.2. 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).

14.3.3. 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.

14.4. 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

14.5. Deliberations

(1) Is it possible that RPL removes 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.

14.6. 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.

15. Capabilities and its role in RPL

RPL is a distributed protocol and it requires that the participating nodes agree on basic set of primitives to follow. RPL currently handles this using MOP (Mode of Operation) bits in the DIO. MOP bits inform the nodes the basic mode of operation a node MUST support to join the Instance as a 6LR. The MOP is decided and advertised by the root of the RPL Instance. A node not supporting the given MOP may still join the Instance as a leaf node or 6LN.

RPL further uses DIO Configuration Option to advertise the configuration each node needs to use (for e.g., for trickle timer).

15.1. Handshaking node capabilities

Currently there exist no mechanism to handshake capabilities of the root or 6LRs or 6LNs. If a feature is optional and is supported by 6LRs/6LNs then currently there exists no mechanism to signal it. There are several RPL extension proposals which are possibly optional features. Root needs to know if the 6LR/6LN supports these optional features to enable the extension in that path context. Similarly 6LRs and 6LNs need to know whether the root supports certain extensions that it can make use of.

15.2. How do Capabilities differ from MOP and Configuration Option?

Unlike MOP and Configuration Option which are issued by the root of the Instance, Capabilities can be issued by any node. A 6LN/6LR node can advertise its capabilities such that those can be seen by intermediate 6LRs and the root of the Instance.

15.3. Deliberations

(1) Is it possible for leaf nodes to advertise their set of capabilities, which can be used by root and/or intermediate 6LRs to make run time decisions?

(2) How should these capabilities be carried? Should it be carried in DAO/DIO/DAO-ACK?

(3) Should the definition of capabilities be same in both directions (upstream/downstream)?

16. Backward Compatibility issues with RPL Options

Most of the new work in ROLL requires addition of new control options. Everytime a new control option is added, it is required that all the nodes upgrade to support this option. In many cases, the new specification declares using a Flag day to switch to the new functionality.

New control options may not require mandatory handling on every node but it requires at-least some processing. For e.g., assume that a new control option is added to DIO message. The option does not require any handling on the nodes not supporting it but it requires at-least for these nodes to forward this new control option downstream. Currently the new control option may be stripped off.

It should be possible for the unknown control options to be copied as-is to the downstream/upstream node(s). The specification defining the new control option will decide whether a node should strip-off or copy the unknown control option.

17. RPL under-specification

(a) PathSequence: Is it mandatory to use PathSequence in DAO Transit Information Option? 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 Option 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 3.

18. 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).

19. IANA Considerations

This memo includes no request to IANA.

20. Security Considerations

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

21. References

21.1. Normative References

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

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

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

[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>.

21.2. 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.

[I-D.ietf-intarea-adhoc-wireless-com]
Baccelli, E. and C. Perkins, "Multi-hop Ad Hoc Wireless Communication", draft-ietf-intarea-adhoc-wireless-com-02 (work in progress), July 2016.

[I-D.ietf-roll-aodv-rpl]
Anamalamudi, S., Zhang, M., Perkins, C., Anand, S., and B. Liu, "AODV based RPL Extensions for Supporting Asymmetric P2P Links in Low-Power and Lossy Networks", draft-ietf-roll-aodv-rpl-08 (work in progress), May 2020.

[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)
Marathahalli
Bangalore, Karnataka 560037
India

Email: rahul.ietf@gmail.com

Rabi Narayan Sahoo
Juniper
Whitefield
Bangalore, Karnataka 560037
India

Email: rabinarayans0828@gmail.com

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

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