A Critical Evaluation of the “IPv6 Routing Protocol for Low Power and Lossy Networks” (RPL)

Thomas Clausen
Hipercom@LIX
Ecole Polytechnique, France
Thomas@ThomasClausen.org

Ulrich Herberg
Trusted Systems Innovation Group
Fujitsu Laboratories of America, USA
ulrich.herberg@us.fujitsu.com

Matthias Philipp
Hipercom@LIX
Ecole Polytechnique, France
Matthias.Philipp@inria.fr

***Abstract*—With RPL – the “IPv6 Routing Protocol for Low-power Lossy Networks” – emerging as a Proposed Standard “Request For Comment” (RFC) in the Internet Engineering Task Force (IETF) after a ∼2-year development cycle, this paper presents a critical evaluation of the resulting protocol and its applicability and limits. The paper presents a selection of observations of the protocol characteristics, exposes experiences acquired when producing a prototype implementation of RPL, and presents results obtained from testing this protocol – both in a network simulator, and in real-world experiments on a wireless sensor network testbed. The paper aims at providing a better understanding of possible weaknesses and limits of RPL, notably the possible directions that further protocol developments should explore, in order to address these.**

I. Introduction

RPL – the *“Routing Protocol for Low Power and Lossy Networks” (RPL)* [1] – is a proposal for an IPv6 routing protocol for Low-power Lossy Networks (LLNs), by the ROLL Working Group in the Internet Engineering Task Force (IETF). This routing protocol is intended to be *the* IPv6 routing protocol for LLNs and sensor networks, applicable in *all* kinds of deployments and applications of LLNs. The unofficial goal, of the ROLL Working Group, is to prevent fragmentation in the sensor networking market by providing an IP-based routing standard, and solicit broad industrial support behind that standard.

The objective of RPL and ROLL is to target networks which *“comprise up to thousands of nodes”*, where the majority of the nodes have very constrained resources, where the network to a large degree is “managed” by a (single or few) central “supernodes”, and where handling mobility is not an explicit design criteria. Supported traffic patterns include multipoint-to-point, point-to-multipoint and point-to-point traffic. The emphasis among these traffic patterns is to *optimize for* multipoint-to-point traffic, to *reasonably support* point-to-multipoint traffic and to *provide basic features for* point-to-point traffic, in that order.

As of early 2011, RPL has been deemed “ready” by the IETF, for publication as a “Proposed Standard” RFC (Request for Comments). The implication of a protocol being labeled “Proposed Standard” is that it is considered generally stable: well-understood and community reviewed, no known design issues pending, and with some community support. “Proposed Standard” is, however, only the first step on what is called the *Standards Track*[1] – experiences with the protocol, from testing and operational deployments, as well as detailed studies of its characteristics and behaviors, may result in protocol changes or retraction.

It is thus opportune to consider the protocol, at its current level of specification, in order to understand which aspects of it necessitate further investigations, and in order to identify possibly weak points which may restrict the deployment scope of the protocol. This paper has as objective to provide a critical evaluation of RPL, in the spirit of better understanding its characteristics and limits.

A. Paper Outline

The remainder of this paper is organized as follows: section II provides an overview of the functional parts of RPL – the algorithms for constructing the basic forwarding structures, as well as protocol signaling. Sections III-XII each explore a specific aspect of RPL, and provide a critical analysis of the impact of the underlying hypotheses made by the designers of RPL. Where possible, abstract reflections on the protocol are complemented by simulation results and results from experiments in a test-bed with real sensor devices. Section XIII concludes this paper by providing both a summary of the observations made, as well as the authors position regarding the applicability of RPL and the possible directions that protocol development should take, in order that IPv6 routing protocols for LLNs can progress – both on the IETF *Standards Track* and in wide-scale real world deployments.

II. RPL Overview

The basic construct in RPL is a “Destination Oriented Directed Acyclic Graph” (DODAG), depicted in figure 1. In a converged LLN, each RPL router has identified a stable set of parents, each of which is a potential next-hop on a path towards the “root” of the DODAG, as well as a *preferred parent*. Each router, which is part of a DODAG (*i.e.* has selected parents) will emit *DODAG Information Object* (DIO) messages, using link-local multicast, indicating its respective *rank* in the DODAG (*i.e.* distance to the DODAG root according to some metric(s), in the simplest form hop-count). Upon having received a (number of such) DIO messages, a router

[1]The *Standards Track* in the IETF consists of “Proposed Standard”, “Draft Standard”, and “Internet Standard”, in increasing order of maturity.

will calculate its own rank such that it is greater than the rank of each of its parents, select a preferred parent and then itself start emitting DIO messages.

The DODAG formation thus starts at the DODAG root (initially, the only router which is part of a DODAG), and spreads gradually to cover the whole LLN as DIOs are received, parents and preferred parents are selected and further routers participate in the DODAG. The DODAG root also includes, in DIO messages, a *DODAG Configuration Object*, describing common configuration attributes for all RPL routers in that network – including their mode of operation, timer characteristics etc. RPL routers in a DODAG include a verbatim copy of the last received DODAG Configuration Object in their DIO messages, permitting also such configuration parameters propagating through the network.

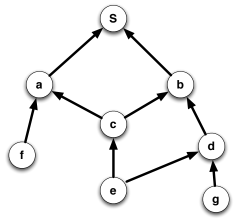

Figure 1. RPL Basic Construct: DODAGs

A Distance Vector protocol, RPL [1] restricts the ability for a router to change rank. A router can freely assume a smaller rank than previously advertised (*i.e.* logically move closer to the root) if it discovers a parent advertising a lower rank, and must then disregard all previous parents of higher ranks. The ability for a router to assume a greater rank (*i.e.* logically move farther from the root) than previously advertised is restricted, to avoid count-to-infinity problems. The root can trigger "global recalculation" of the DODAG by increasing a sequence number, *DODAG version*, in DIO messages.

The DODAG so constructed is used for installing routes: the "preferred parent" of an RPL router can serve as a default route towards the root, or the root can embed in its DIO messages the destination prefixes, included by DIOs generated by RPL routers through the LLN, to which connectivity is provided by the root. Thus, RPL by way of DIO generation provides "upward routes" or "multipoint-to-point routes" from the sensors inside the LLN and towards the root.

"Downward routes" are enabled by having sensors issue *Destination Advertisement Object* (DAO) messages, propagating as unicast via parents towards the DODAG root. These describe which prefixes belong to, and can be reached via, which RPL router. In a network, all RPL routers must operate in either of storing-mode or non-storing-mode, specified by way of a "Mode of Operation" (MOP) flag in the DODAG Configuration Object from the root. Depending on the MOP, DAO messages are forwarded differently towards the root:

- In *non-storing-mode*, an RPL router originates DAO messages, advertising one or more of its parents, and unicast it to the DODAG root. Once the root has received DAOs from an RPL router, and from all routers on the path between it and the root, it can use source routing for reaching advertised destinations inside the LLN.
- In *storing-mode*, each RPL router on the path between the originator of a DAO and the root records a route to the prefixes advertised in the DAO, as well as the next-hop towards these (the router, from which the DAO was received), then forwards the DAO to its preferred parent.

"Point-to-point routes", for communication between devices inside the LLN and where neither of the communicating devices are the DODAG root, are as default supported by having the source sensor transmit via its default route to the DODAG root (*i.e.,* using the upward routes) which will then, depending on the "Mode of Operation" for the DODAG, either add a source-route to the received data for reaching the destination sensor (downward routes in non-storing-mode) or simply use hop-by-hop routing (downward routes in storing-mode). In the case of storing-mode, if the source and the destination for a point-to-point communication share a common ancestor other than the DODAG root, a downward route may be available (and used) before reaching the DODAG root.

A. RPL Message Emission Timing – Trickle Timers

RPL message generation is timer-based, with the root able to configure back-off of message emission intervals using *Trickle* [2], specified in [3]. Trickle, as used in RPL, stipulates that a RPL router transmits a DIO "every so often" – except if receiving a number of DIOs from neighbor routers, enabling the router to determine if its DIO transmission is redundant.

When an RPL router transmits a DIO, there are two possible outcomes: either every neighbor router that hears the message finds that the information contained is consistent with its own state (*i.e.,* the received DODAG version number received corresponds with that which the RPL router has recorded and no better rank is advertised than that which is recorded in the parent set) – or, a recipient RPL router detects that either the sender of the DIO or itself has out-of-date information. If the sender has out-of-date information, then the recipient RPL router schedules transmission of a DIO to update this information. If the recipient RPL router has out-of-date information, then it updates based on the information received in the DIO.

With Trickle, an RPL router will schedule emission of a DIO at some time, t, in the future. When receiving a DIO containing information consistent with its own information, the RPL router will record that "redundant information has been received" by incrementing a redundancy counter, c. At the time t, if c is below some "redundancy threshold", then it transmits its DIO. Otherwise, transmission of a DIO at this time is suppressed, c is reset and a new t is selected to twice as long time in the future – bounded by a pre-configured maximum value for t. If, on the other hand, the RPL router has received an out-of-date DIO from one of its neighbors, t is reset to a pre-configured minimum value and c is set to zero. In both cases, at the expiration of t, the RPL router will verify if c is below the "redundancy threshold" and if so transmit – otherwise, increase t and stay quiet.

III. RPL Data Traffic Flows

RPL makes a-priori assumptions of traffic patterns: sensor-to-root traffic (*multipoint-to-point*) is predominant, root-to-sensor traffic (*point-to-multipoint*) is rare and sensor-to-sensor traffic is somewhat esoteric. While not specifically called out thus in [1], the resulting protocol design reflects these assumptions in that mechanism constructing multipoint-to-point paths is efficient in terms of control traffic generated and state required, point-to-multipoint path construction much less so – and sensor-to-sensor paths subject to potentially significant stretch.

An RPL router selects from among its parents a "preferred parent", to serve as a default route towards the root (and to prefixes advertised by the root). Thus, RPL provides "upward routes" or "multipoint-to-point routes" from the sensors towards the root. An RPL router which wishes to act as a destination for traffic ("downward routes" or "point-to-multipoint") issues DAOs upwards in the DODAG towards the root, describing which prefixes belong to, and can be reached via, that RPL router. Sensor-to-sensor routes are supported by having the source sensor transmit, via its default route, towards the root. In non-storing mode, the data will reach the root, which will send the data packet downward towards the destination sensor. In storing mode, the source and the destination may have a common ancestor other than the DODAG root, which may provide a downward route to the destination.

A. Why This Is A Critical Point

The data traffic characteristics assumed by RPL do not represent a universal distribution of traffic patterns in LLNs:

- There are scenarios where sensor-to-sensor traffic is a more common occurrence, documented *e.g.* in [4].
- There are scenarios, where all traffic is bi-directional, *e.g.* in case sensor devices in the LLN are, in majority, "actively read": a request is issued by the root to a specific sensor, and the sensor value is expected returned.

For the former, all sensor-to-sensor paths include the root, possibly causing congestion on the communications medium near the root, and draining energy from the intermediate RPL routers on an unnecessarily long path. If sensor-to-sensor traffic is common, RPL routers near the root will be particularly solicited as relays, especially in non-storing mode. For the latter, all RPL routers are required to generate DAOs, which generates a considerable control traffic overhead [5].

IV. Fragmentation

Fragmentation of IP packets appears when the size of the IP datagram is larger than the Maximum Transmission Unit (MTU) supported by the link layer. When an IP packet is fragmented, all fragments of that IP packet must be successfully received by a router, in order that the IP packet is successfully received – otherwise, the whole IP packet is lost. Moreover, the additional link-layer frame overhead for each of the fragments increases the capacity required from the medium, and may consume more energy for transmitting a higher number of frames on the network interface.

RPL is an IPv6 routing protocol, designed to operate on constrained link layers, such as 802.15.4 [6], with a maximum MTU of 127 bytes – a deviation from the otherwise specified minimum MTU of 1280 bytes for IPv6 [7]. Reducing the need of fragmentation of packets on such a link layer, compression adaptation layers exist [6], [8], reducing the overhead of the IPv6 header from at least 40 octets to a minimum of 2 octets. With a physical layer packet size of 127 octets, a maximum frame overhead of 25 octets and 21 octets for link layer security [6], 81 octets remain for L2 payload. Further subtracting 2 octets for the compressed IPv6 header leaves 79 octets for L3 data payload.

The second L in LLN indicating *Lossy* [9], higher loss rates than typically seen in IP networks are expected, rendering fragmentation important to avoid.

DIO messages consist of a mandatory base object, facilitating DODAG formation, and additional options for *e.g.* autoconfiguration and network management. The base object contains two unused octets, reserved for future use, resulting in two bytes of unnecessary zeros, sent with each DIO message. The Prefix Information option, used for automatic configuration of address, is even worse: it carries four unused octets to be compatible with IPv6 neighbor discovery.

A. Why This Is A Critical Point

While 79 octets may seem to be sufficient to carry RPL control messages, consider the following: RPL control messages are carried in ICMPv6, and the mandatory ICMPv6 header consumes 4 octets. The DIO base another 24 octets. If link metrics are used, that consumes at least another 8 octets[2]. The DODAG Configuration Object consumes up to a further 16 octets, for a total of 52 octets. Adding a Prefix Information Object for address configuration consumes another 32 octets, for a total of 84 octets – thus exceeding the 79 octets available for L3 data payload and causing fragmentation of such a DIO. As a point of reference, the ContikiRPL [10] implementation includes both the DODAG Configuration option and the Prefix Information option in all DIO message. Any other options, *e.g.* Route Information options indicating prefixes reachable through the root, worsen this situation.

RPL may further increase the probability of fragmentation of also user data traffic: for non-storing-mode, RPL employs source-routing for all downward traffic. [11] specifies the RPL Source Routing header, which imposes a fixed overhead of 8 octets per IP packet leaving 71 octets remaining from the MTU – from which must be deducted a variable number of octets, depending on the length of the route. With fewer octets available for data payload, RPL thus increases the probability for fragmentation of also data packets. This, in particular, for longer paths, *e.g.* in point-to-point traffic between sensors inside the LLN, where data packets transit through the DODAG root and are then source-routed to the destination.

[2]Using a hop count metric; other metrics may require more.

V. DAO Mechanism

RPL specifies two distinct and incompatible "modes of operation" for downward traffic: *storing mode*, where each RPL router is assumed to maintain routes to all destinations in its sub-DODAG, *i.e.* routers that are "deeper down" in the DAG, and *non-storing mode*, where only the root stores routes to destinations in the LLN.

A. Why This Is A Critical Point

In addition to possible fragmentation, as discussed in section IV, the maximum length of the source routing header [11] is limited to 136 octets, including an 8 octet long header. As each IPv6 address has a length of 16 octets, not more than 8 hops from the source to the destination are possible for "raw IPv6". Using address compression [6], the maximum path length may not exceed 64 hops. This excludes scenarios with long "chain-like" topologies, such as traffic lights along a street.

In *storing mode*, each RPL router has to store routes for destinations in its sub-DODAG. This implies that, for RPL routers near the root, the required storage is only bounded by the number of paths to all other destinations in the network. As RPL targets constrained devices with little memory, but also has as ambition to be operating networks consisting of thousands of routers, the storing capacity on these RPL routers may not be sufficient. Indeed, [12] argues that practical experiences suggest that RPL in storing mode should be limited to networks of less than ~30 routers. Aggregation / summarization of addresses may be advanced as a possible argument that this issue is of little significance – section VI will discuss why such an argument does not apply.

In short, the mechanisms in RPL force the choice between requiring all RPL routers to have sufficient memory to store route entries for all destinations (storing-mode) – or, suffer increased risk of fragmentation, and thus loss of data packets, while consuming network capacity by way of source routing through the DODAG root.

In RPL, the "mode of operation" stipulate that either downward routes are not supported (MOP=0), or that they are supported by way of either storing or non-storing mode. In case downward routes are supported, RPL does not provide any mechanism for discriminating between which routes should or should not be maintained. In particular, in order to calculate paths to a given destination, all intermediaries between the DODAG root and that destination must themselves be reachable – effectively rendering downward routes in RPL an "all-or-none" situation. In case a destination is unreachable, all the DODAG root may do is require all destinations to re-issue their DAOs[3], possibly provoking a broadcast-storm-like situation. This, in particular, as [1] does not specify DAO message transmission constraints, in particular specifies no mechanism for adapting DAO emission to the network capacity.

A final point on the DAO mechanism: RPL supports point-to-point traffic only by way of relaying through the DODAG root. In networks where point-to-point traffic is no rare occasion, this causes unduly long paths (with possibly increased energy consumption, increased probability of packet losses) as well as possibly congestion around the DODAG root.

[3]By issuing a DIO with an increased DODAG version number.

VI. Aggregation

As indicated in section V, in storing mode, a RPL router is expected to be able to store routing entries for all destinations in its "sub-DODAG", *i.e.,* routing entries for all destinations in the network where the path to the DODAG root includes that RPL router.

In the Internet, no single router stores explicit routing entries for all destinations – no router has a routing table with 2^{32} entries for IPv4 routing. Rather, IP addresses are assigned hierarchically, such that an IP address does not only uniquely identify a network interface, but also its topological location in the network, as illustrated in figure 2. Colloquially speaking, all addresses with the same prefix are reachable by way of the same router – which can, therefore, advertise only that prefix. Other routers need only record a single routing entry for that prefix, knowing that as the IP packet reaches the router advertising that prefix, more precise routing information is available.

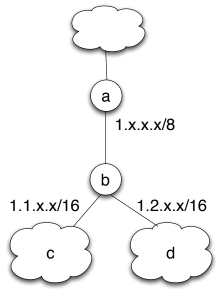

Figure 2. Addressing hierarchies in the Internet

A. Why This Is A Critical Point

In RPL, each RPL router acquires a number of parents, as described in section II, from among which it selects one as its preferred parent and, thus, next-hop on the path to the DODAG root. RPL routers maintain a parent set containing possibly more than a single parent so as to be able to rapidly select an alternative preferred parent, should the previously selected such become unavailable. Thus expected behavior is for an RPL router to be able to change its point of attachment towards the DODAG root. If IP addresses are assigned in a strictly hierarchical fashion, and if scalability of the routing state maintained in storing mode is based on this hierarchy, then this entails that each time a RPL router changes its preferred parent, it must also change its own IP address – as well as cause RPL routers in its "sub-DODAG" to do the same. RPL does not specify signaling for reconfiguring addresses in a sub-DODAG.

A slightly less strict hierarchy can be envisioned, where a router can change its preferred parent without necessarily changing addresses of itself and of its sub-DODAG, provided that its former and new preferred parents both have the same preferred parent, and have addresses hierarchically assigned from that – from the "preferred grandparent". With reference to figure 1, this could be `e` changing its preferred parent from `d` to `c`, provided that both `d` and `c` have `b` as preferred parent. Doing so would impose a restriction on the parent-set selection, admitting only parents which have themselves the same parent – thus, no longer having a DODAG but a simple tree, loosing redundancy in the network connectivity. RPL does not specify rules for admitting only parents with identical grand-parents into the parent set – although such is not prohibited either, if the loss of redundancy from constructing a tree is acceptable.

The DODAG root incrementing the DODAG version number is the mechanism by which RPL enables global reconfiguration of the network, reconstructing the DODAG with (intended) more optimal paths. In case of addressing hierarchies being enforced, so as to enable aggregation, this will either restrict the ability for an optimal DODAG construction, or will trigger global address autoconfiguration so as to ensure addressing hierarchies.

Finally, with IP addresses serving a dual role of an identifier of both an end-point for communication and a topological location in the network, changing the IP address of a device, so as to reflect a change in network topology, also entails interrupting ongoing communication to or through that device. Additional mechanisms (*e.g.* a DNS-like system) mapping "communications identifies" and "IP addresses" is required – a topic investigated, but not resolved, in the Internet[4].

VII. Bidirectionality Hypothesis

Parents (and the preferred parent) are selected based on receipt of DIOs, without verification of the ability for a RPL router to successfully communicate with the parent – *i.e.* without any bidirectionality check of links. However, the basic use of links is for "upward" routes, *i.e.* for the RPL router to use a parent (the preferred parent) as relay towards the DODAG root – in the opposite direction of the one in which the DIO was received.

A. Why This Is A Critical Point

Unidirectional links are no rare occurrence, such as is known from wireless multi-hop networks. If an RPL router receives a DIO on such a unidirectional link, and selects the originator of the DIO as parent, that would be a bad choice: unicast traffic in the upward direction would be lost. If the router had verified the bidirectionality of links, it might have selected a better parent, to which it has a bidirectional link.

VIII. Why NUD is NOT a Solution

[1] suggests using Neighbor Unreachability Detection (NUD) [13] to detect and recover from the situation of unidirectional links between a RPL router and its (preferred) parent(s). When, *e.g.*, a router *a* tries (and fails) to actually use router *b* for forwarding traffic, NUD is supposed engaged to detect and prompt corrective action, *e.g.* by way of selecting an alternative preferred parent.

NUD is based upon observing if a data packet is making forward progress towards the destination, either by way of indicators from upper-layer protocols (such as TCP)[5] or – failing that – by unicast probing by way of transmitting a *unicast* Neighbor Solicitation message and expecting that a solicited Neighbor Advertisement message be returned.

A. Why This Is A Critical Point

An RPL router may receive, *transiently*, a DIO from a router, closer (in terms of rank) to the root than any other router from which a DIO has been received. Some, especially wireless, link layers may exhibit different transmission characteristics between multicast and unicast transmissions[6], leading to a (multicast) DIO being received from farther away than a unicast transmission can reach. DIOs are sent (downward) using link-local multicast, whereas the traffic flowing in the opposite direction (upward) is unicast. Thus, a received (multicast) DIO may not be indicative of useful unicast connectivity – yet, RPL might cause this RPL router to select this attractive router as its preferred parent. This may happen both at initialization or at any time during the LLN lifetime, as RPL allows attachment to a "better parent" at any time.

A DODAG so constructed may appear stable and converged until such time that unicast traffic is to be sent and, thus, NUD invoked. Detecting only at that point that unicast connectivity is not maintained, and causing local (and possibly global) repairs exactly at that time, may lead to traffic not being deliverable. As indicated in section VI, if scalability is dependent on addresses being assigned hierarchically, changing point-of-attachment may entail more than switching preferred parent.

Also, absent all RPL routers consistently advertising their reachability through DAO messages, a protocol requiring bidirectional flows between the communicating devices, such as TCP, will be unable to operate.

Finally, upon having been notified by NUD that the "next hop" is unreachable, an RPL router must discard the preferred parent and select another – hoping that this time, the preferred parent is actually reachable. Also, if NUD indicates "no forward progress" based on an upper-layer protocol, there is no guarantee that the problem stems exclusively from the preferred parent being unreachable. Indeed, it may be a problem farther ahead, possibly outside the LLN, thus changing preferred parent will do nothing to alleviate the situation.

[4]The IETF LISP working group, https://datatracker.ietf.org/wg/lisp/charter/, is chartered to address this issue.

[5]Though not called out in [13], also from lower-layer protocols (such as Link Layer ACKs).

[6]Such is the case for some implementations of IEEE 802.11b, where multicast/broadcast transmissions are sent at much lower bit-rates than are unicast. IEEE 802.11b is, of course, not suggested as a viable interface for LLNs, but serves to illustrate that such asymmetric designs exist.

IX. RPL Implementability and Complexity

RPL is designed to operate on "RPL routers [...] with constraints on processing power, memory, and energy (battery power)" [1]. However, the 163 pages long specification of RPL[7], describes complex mechanisms (*e.g.* the upwards and downward data flows, a security solution, manageability of RPL routers, auxiliary functions for autoconfiguration of RPL routers, etc.), and provides no less than 9 message types, and 10 different message options.

To give one example, the ContikiRPL implementation[8], which provides only storing-mode and no security features, consumes about 50 KByte of memory. Sensor hardware, such as MSP430 sensor platforms, does not contain much more memory than that, *i.e.* there may not be much space left to deploy any application on the RPL router.

A. Why This Is A Critical Point

Since RPL is designed to be *the* routing protocol for LLNs, which covers all the diverse applications requirements listed in [4], [16], [17], [18], it is possible that (i) due to limited memory capacity of the RPL routers, and (ii) due to expensive development cost of the routing protocol implementation, many RPL implementations will only support a partial set of features from the specification, leading to non-interoperable implementations.

X. RPL Underspecification

While [1] is verbose in many parts, as described in section IX, some mechanisms are underspecified.

While for DIOs, the Trickle timer specifies an efficient and easy-to-understand timing for message transmission, the timing of DAO transmission is not explicit. As each DAO may have a limited lifetime, one "best guess" for implementers would be to send DAO periodically, *just before* the life-time of the previous DAO expires. Since DAOs may be lost, another "best guess" would be to send several DAOs shortly one after the other in order to increase probability that at least one DAO is successfully received.

The same underspecification applies for DAO-ACK messages: optionally, on reception of a DAO, an RPL router may acknowledge successful reception by returning a DAO-ACK. Timing of DAO-ACK messages is unspecified by RPL.

A. Why This Is A Critical Point

By not specifying details about message transmission intervals and required actions when receiving DAO and DAO-ACKs, implementations may exhibit a bad performance if not carefully implemented. Some examples are:

1) If DAO messages are not sent in due time before the previous DAO expires (or if the DAO is lost during transmission), the routing entry will expire before it is renewed, leading to a possible data traffic loss.
2) RPL does not specify to use jitter [19] (*i.e.* small random delay for message transmissions). If DAOs are sent periodically, adjacent routers may transmit DAO messages at the same time, leading to link layer collisions.
3) In non-storing mode, the "piece-wise calculation" of routes to a destination from which a DAO has been received, relies on previous reception of DAOs from intermediate routers along the path. If not all of these DAOs from intermediate routers have been received, route calculation is not possible, and DAO-ACKs or data traffic cannot be sent to that destination.

Other examples of underspecification include the local repair mechanism, which may lead to loops and thus data traffic loss, if not carefully implemented: a router discovering that all its parents are unreachable, may – according to the RPL specification – "detach" from the DODAG, *i.e.* increase its own rank to infinity. It may then "poison" its sub-DODAG by advertising its infinite rank in its DIOs. If, however, the router receives a DIO *before* it transmits the "poisoned" DIO, it may attach to its own sub-DODAG, creating a loop. If, instead, it had waited some time before processing DIOs again, chances are it would have succeeded in poisoning its sub-DODAG and thus avoided the loop.

XI. Trickle Convergence

Trickle [3] is used by RPL to schedule transmission of DIO messages, with the objective to minimize the amount of transmitted DIOs while ensuring a low convergence time of the network. The theoretical behavior of Trickle is well understood, and the convergence properties are well studied [2]. Simulations of the mechanism, such as [20], confirm these theoretical studies.

In real-world environments, however, varying link qualities may cause the algorithm to converge less well: frequent message losses entail resets of the Trickle timer and more frequent and unpredicted message emissions. This has been observed in an experimental testbed: 69 RPL routers[9] were positioned in a fixed grid topology. This resulted in DODAGs being constructed with an average of 2.45 children per RPL router and an average rank[10] of 3.58.

Figure 3 shows the number of DIO messages that are emitted (counting each retransmission at intermediate routers) per interval of 10 seconds, in both the test bed and an identical scenario simulated in Ns2. The Ns2 simulation parameters were chosen to match the testbed environment, as far as possible (80 routers, 1265x1265m area, 1800s simulation time, 802.11, two-ray-ground model, 250m transmission range, no mobility, JRPL implementation [21]).

While the number of DIOs, emitted per 10 second interval in the whole network, rapidly drops in the simulation, a constant

[7]Plus additional specifications for routing headers [11], Trickle timer [3], routing metrics [14] and objective function [15].

[8]http://www.sics.se/contiki

[9]MSP430-based wireless sensor routers with IEEE 802.15.4, using [10] IPv6 stack and RPL without downward routes; the parameters of the Trickle timer were set to the implementation defaults (minimum DIO interval: 4 s, DIO interval doublings: 8, redundancy constant: 10).

[10]Hop count.

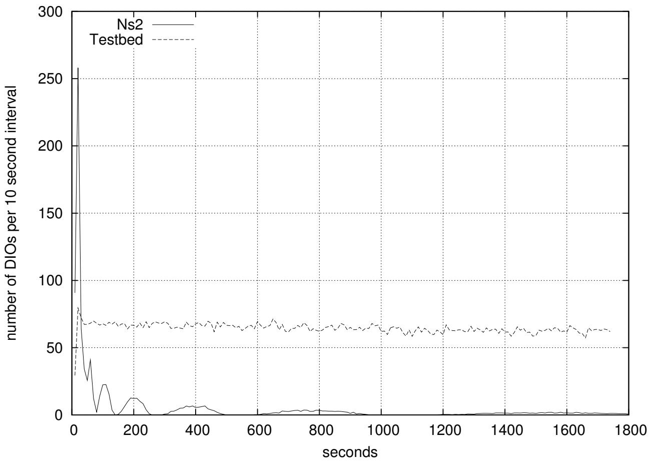

Figure 3. Trickle convergence for simulation and real world experiment

number of about 70 DIOs per 10 second interval was observed in the testbed experiment.

A. Why This Is A Critical Point

The varying link quality in real-world environments results in frequent changes of the best parent, which triggers a reset of the Trickle timer and thus the emission of DIOs. Therefore Trickle does not converge as well for links that are fluctuating in quality as in theory.

The resulting higher control overhead due to frequent DIO emission, leads to higher bandwidth and energy consumption as well as possibly to an increased number of collisions of frames, as observed in [20].

XII. Loops

Section 7.1. describes one way in which routing loops can occur in RPL. [1] states that it "guarantees neither loop free path selection nor tight delay convergence times, but can detect and repair a loop as soon as it is used. RPL uses this loop detection to ensure that packets make forward progress [...] and trigger repairs when necessary". This implies that a loop may only then be detected and fixed when data traffic is sent through the network.

In order to trigger a local repair, RPL relies on the "direction" information (with values "up" or "down"), contained in an IPv6 hop-by-hop option header [22] of a data packet. If an "upward" data packet is received by a RPL router, but the previous hop of the packet is listed with a lower rank in the neighbor set, the RPL router concludes that there must be a routing loop and it may therefore trigger a local repair. For downward traffic in non-storing mode, the root can detect loops if the same router identifier (*i.e.* IP address) appears at least twice in the path towards a destination.

A. Why This Is A Critical Point

The reason for RPL to repair loops only when detected by a data traffic transmission is to reduce control traffic overhead. However, there are two problems in repairing loops only when so triggered: (i) the triggered local repair mechanism delays forward progress of data packets, increasing end-to-end delays, and (ii) the data packet has to be buffered during repair.

(i) may seem as the lesser of the two problems, since in a number of applications, such as data acquisition in smart metering applications, an increased delay may be acceptable. However, for applications such as alarm signals or in home automation (*e.g.* a light switch), increased delay may be undesirable.

As for (ii), RPL is supposed to run on LLN routers with "constraints on [...] memory" [1]; buffering incoming packets during the route repair may not be possible for all incoming data packets, leading to dropped packets. Depending on the transport protocol, these data packets must be retransmitted by the source or are definitely lost.

If carefully implemented with respect to avoiding loops before they occur, the impact of the loop detection in RPL may be minimized. However, it can be observed that with current implementations of RPL, such as the ContikiRPL implementation, loops do occur frequently. During the experiments described in section XI, a snapshot of the DODAG was taken every ten seconds. In 74.14 % of the 4114 snapshots, at least one loop was observed. Further investigation revealed that in all these cases the DODAG was partitioned, and the loop occurred in the sub-DODAG that no longer had a connection to the DODAG root. When the link to the only parent of a router breaks, the router may increase its rank and – when receiving a DIO from a router in its sub-DODAG – attach itself to its own sub-DODAG, thereby creating a loop – as detailed in section X.

While it can be argued that the observed loops are harmless since they occur in a DODAG partition that has no connection to the root anyway, they show that the state of the network is inconsistent. Even worse, when the broken link re-appears, it is possible that in certain situations the loop is only then repaired when data traffic is sent, possibly leading to data loss (as described above). This can occur if the link to the previous parent is reestablished, but the rank of that previous parent has increased in the meantime.

Another problem with the loop repair mechanism arises in non-storing mode when using only downward traffic: while the root can easily detect loops (as described above), it has no direct means to trigger a local repair where the loop occurs. Instead, it can only trigger a global repair by increasing the DODAG version number, leading to a Trickle timer reset and increased control traffic overhead in the network caused by DIO messages, and therefore a possible energy drain of the routers and congestion of the channel.

XIII. Conclusion and Position

Modulo the issues presented regarding *bi-directionality* of links and the possibility of loops, DODAG formation, and so the multipoint-to-point route provisioning mechanism, is elegant and relatively well understood, although the difference in convergence between theory/simulation and real-world sensor network behaviors necessitates further exploration. DIO message generation/processing rules and the Trickle timers [3] are relatively straight-forward to implement, and the state required in each router is minimal and bounded.

The DAO mechanism is what enables *downward* routes, bi-directional traffic flows and sensor-to-sensor flows by way of dog-leg-routing through the root, is less elegant. Problems include "underspecification", *e.g.* of the proper behavior with respect to DAO-ACKs and DAO retransmissions and message generation intervals, as well as two incompatible modes-of-operation: *storing mode*, wherein all LLN routers are expected to have "unbounded" memory (or, at least, enough to store complete routing tables), and *non-storing mode* necessitating source-routing thus possibly more fragmentation and higher probability of IP packets being lost. Both of these appear to be challenging in Low-power *Lossy* Networks with resource-constrained devices – as does addressing scalability concerns by way of address aggregation appear unfeasible in a such self-forming network.

Loops are a real problem in RPL, confirmed experimentally. Even in non-storing mode, where the DODAG root performs source-routing to destinations inside the LLN, loops are a problem: while they can be detected when constructing the source route, the only corrective measure that the DODAG root can take is to trigger global reconstructions of the DODAG – a complete "reboot" of the LLN.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," March 2011, Internet Draft, work in progress, draft-ietf-roll-rpl-19.

[2] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation*, 2004.

[3] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "RFC6206: The Trickle Algorithm," March 2011, Standards Track RFC 6206.

[4] J. Martocci, P. D. Mi, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low Power and Lossy Networks," June 2010, Informational RFC 5867.

[5] T. Clausen and U. Herberg, "A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-Directional Traffic in Low-power and Lossy Networks (LLN)," INRIA, Research Report RR-7637, Jun. 2011. [Online]. Available: http://hal.inria.fr/inria-00598157/en/

[6] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," September 2007, Standards Track RFC 4944.

[7] S. Deering and R. Hinden, "RFC2460: Internet Protocol, Version 6 (IPv6) Specification," December 1998, Standards Track RFC 2460.

[8] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams in Low Power and Lossy Networks (6LoWPAN)," February 2011, Internet Draft, work in progress, draft-ietf-6lowpan-hc-15.

[9] "ROLL Charter," http://datatracker.ietf.org/wg/roll/charter/.

[10] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-Power Wireless IPv6 Routing with ContikiRPL," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (ISPN)*, April 2010.

[11] J. Hui, J. Vasseur, D. Culler, and V. Manral, "An IPv6 Routing Header for Source Routes with RPL," March 2011, Internet Draft, work in progress, draft-ietf-6man-rpl-routing-header-03.

[12] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of rpl and 6lowpan in tinyos," April 2011.

[13] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," September 2007, Standards Track RFC 4861.

[14] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics used for Path Calculation in Low Power and Lossy Networks," March 2011, Internet Draft, work in progress, draft-ietf-roll-routing-metrics-19.

[15] P. Thubert, "RPL Objective Function 0," May 2011, Internet Draft, work in progress, draft-ietf-roll-of0-12.

[16] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," October 2009, Informational RFC 5673.

[17] A. Brandt, J. Buron, and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks," April 2010, Informational RFC 5826.

[18] M. Dohler, T. Watteyne, T. Winter, and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks," May 2009, Informational RFC 5548.

[19] T. Clausen, C. Dearlove, and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)," February 2008, Standards Track RFC 5148.

[20] U. Herberg and T. Clausen, "Study of Multipoint-to-Point and Broadcast Traffic Performance in the 'IPv6 Routing Protocol for Low Power and Lossy Networks' (RPL)," *Journal of Ambient Intelligence and Humanized Computing*, 2011.

[21] U. Herberg, "JRPL web site," http://herberg.name/projects/83-wireless-sensor-networks.

[22] J. Hui and J. Vasseur, "RPL Option for Carrying RPL Information in Data-Plane Datagrams," March 2011, Internet Draft, work in progress, draft-ietf-6man-rpl-option-03.