Some Considerations on Routing in Particular and Lossy Environments

Thomas Clausen, Ulrich Herberg
Laboratoire d'Informatique (LIX) – Ecole Polytechnique, France
Thomas@ThomasClausen.org, Ulrich@Herberg.name

***Abstract*—This paper presents a selection of observations and experiences acquired when producing a prototype implementation of RPL as well as an evaluation of the applicability of this protocol for various specific "real-world" deployments.**

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). The basic construct in RPL is a DODAG – a destination oriented directed acyclic graph, rooted in a "controller".

Traffic inside the LLN flows along this DODAG, either upward (towards the "controller") or downward. In RPL, upward routes, having the controller as destination (either by way of explicitly addressing the destination, or by using the controller as "gateway"), are provided by the DODAG construction mechanism: each LLN router selects a set of parents, on a path towards the controller, as well as a *preferred parent*. Once a router is part of a DODAG (*i.e.* has selected parents) will emit *DODAG Information Object* (DIO) messages, using link-local multicasting, indicating its respective *rank* in the DODAG (*i.e.* its position – distance according to some metric(s), in the simplest form hop-count – with respect to the root).

Routes for any destination inside the LLN, other than the controller, are provided by these destinations generating *Destination Advertisement Objects* (DAOs).

II. RPL Data Traffic Flows

RPL makes a-priori assumptions of traffic patterns: sensor-to-controller traffic (*multipoint-to-point*) is predominant, controller-to-sensor traffic (*point-to-multipoint*) is rare and sensor-to-sensor traffic is somewhat esoteric.

An LLN router in RPL will select a "preferred parent", to serve as a default route towards the controller (or prefixes advertised by the controller). Thus, RPL provides "upward routes" or "multipoint-to-point routes" from the sensors towards the controller.

An LLN router in RPL, which wishes to act as a destination for traffic ("downward routes" or "point-to-multipoint"), will issue DAOs upwards in the DODAG towards the controller, describing which prefixes belong to, and can be reached via, that LLN router.

Sensor-to-sensor routes are supported by having the source sensor transmit, via its default route, to the controller, which will send the data packet downward towards the destination sensor.

A. Why This Is A Critical Point

The data traffic characteristics assumed by RPL does not represent a universal distribution of traffic patterns in LLNs, for example:

- There are scenarios in which sensor-to-sensor traffic is assumed a more common occurrence, such as [2].
- There are scenarios, in which all traffic is bi-directional, such as for example in case sensor devices in the LLN are, in majority, "actively read": a request is issued by the controller to a specific sensor, and the sensor value is expected returned.

For the former, all sensor-to-sensor routes transit the root, possibly causing congestion in the wireless spectrum near the root, as well as draining energy from the intermediate routers on an unnecessarily long path.

For the latter, all LLN routers are required to generate DAOs, which generates a considerable control traffic overhead.

III. DAO Mechanism

Two distinct "modes of operation" for the downward mechanism are specified in RPL:

(i) In *storing mode*, each router is assumed to maintain routes to all LLN routers in its sub-DODAG, *i.e.* routers that are "deeper down" in the DAG. DAOs propagate from the routers towards the controller, where each intermediate router adds its reverse routing stack to the DAO message (aggregating routes where possible).

(ii) In *non-storing mode*, only the controller stores routes to all LLN routers in the network. Each LLN router unicasts DAOs to the controller, which then calculates routes to all destinations by "piecing together" the information collected from DAO messages (which contain the destination prefix of the LLN router and addresses of the parents through which it is reachable). In non-storing mode, downward traffic is sent by way of source.

A. Why This Is A Critical Point

In *non-storing mode*, source-routes increase the L3 header length – with small MTUs, this may lead to increased fragmentation, thus successful delivery of an IP packet depends on successful delivery of possibly more fragments – a single

fragment lost renders otherwise successfully delivered fragments of the IP packet lost. LLNs are, generally, characterized by higher loss-rates and smaller MTUs [3]. In addition to possible fragmentation, the maximum length of the source routing header [4] is limited to 255 octets at maximum. As each IPv6 address has a length of 16 octets, not more than 15 hops from the source to the destination are possible. Even with address compression, such as specified in [5], the maximum path length may not exceed 127 hops. This excludes scenarios with long "chain-like" topologies, such as traffic lights along a street.

In *storing mode*, each LLN router has to store routes for its sub-DODAG. This implies that, for LLN routers near the controller, the required storage is only bounded by the number of paths to all other LLN routers in the network. As RPL targets constrained devices with little memory, but in networks consisting of thousands of routers, the storing capacity on these LLN routers may not be sufficient.

IV. Bidirectionality Hypothesis

Parents (and the preferred parent) are selected based on receipt of DIOs, without verification of the ability for a LLN 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 LLN router to use a parent (the preferred parent) as relay towards the DODAG controller – 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 LLN 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.

V. Why NUD is NOT a Solution

[1] suggests using Neighbor Unreachability Detection (NUD) [6] to detect and recover from the situation of unidirectional links between a LLN router and its (preferred) parent(s). When *a* tries (and fails) to actually use *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)[1] 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.

[1]Though not called out in [6], also from lower-layer protocols (such as Link Layer ACKs)

A. Why This Is A Critical Point

A LLN router may receive, *transiently*, a DIO from a router, much closer (in terms of rank) to the controller 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[2], 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 LLN 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.

Also, absent all LLN 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, a LLN router must discard the preferred parent and select another preferred parent – 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 father ahead, possibly outside the LLN, thus changing preferred parent will do nothing to alleviate the situation.

VI. RPL Implementability and Complexity

RPL is designed to operate on "LLN routers [...] with constraints on processing power, memory, and energy (battery power)" [1]. However, the 159 pages long specification[3], describes complex mechanisms (*e.g.* the upwards and downward data flows, a security solution, manageability of LLN routers, auxiliary functions for autoconfiguration of LLN routers, etc.), and provides no less than 9 message types, and 10 different message options.

To give one example, the ContikiRPL implementation[4], which does not provide non-storing mode or any security features, consumes about 50 KByte of memory. Sensor hardware,

[2]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.

[3]Plus additional specifications for routing headers [4], Trickle timer [7], routing metrics [8] and objective function [9].

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

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 LLN 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 [2], [10], [11], [12], it is possible that (i) due to limited memory capacity of the LLN 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.

As RPL is targeted for a "Standards Track" publication, interoperable implementations are desired. Already during the IPSO Interoperability Workshop in 2010 was it observed that several implementations were not interoperable, as they only implemented one mode of operation or the other (*i.e.* storing or non-storing mode).

VII. RPL Underspecification

While [1] is verbose in many parts, as described in section VI, 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 LLN router may acknowledge successful reception by sending back a DAO-ACK. Again, timing of the DAO-ACK messages is not specified 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 [13] (*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.

VIII. Position

While RPL provides support for all of *multipoint-to-point*, *point-to-multipoint* and sensor-to-sensor traffic, its strength clearly is in providing connectivity for *multipoint-to-point* flows. Modulo the issues presented regarding *bi-directionality* of links and the possibility of loops, the DODAG formation mechanism is elegant, efficient, and relatively well understood. The DIO message generation/processing rules and the trickle timers [7], necessary for this DODAG formation, are relatively straight-forward to implement as well – are actually not very complex. The state required in each router is also minimal and bounded – down to a single entry in the routing table (the preferred parent). Such pure *multipoint-to-point* traffic flows are not rare either: *data-acquisition* networks, where sensors on their own impulse communicate their readings to a controller are important, *e.g.* in various environmental monitoring or *data acquisition* scenarios.

As elegant as the support for pure *multipoint-to-point* traffic flows (*i.e. pure upward routes*) by way of DODAG construction is, the support for other traffic flows appears less so: the DAO mechanism, supporting *downward* routes, is what enables bi-directional traffic flows (*e.g.* for *active reading* of sensors) and sensor-to-sensor flows by way of dog-leg-routing through the controller. This mechanism appears as a complicated addition to the elegant DODAG mechanism. It is, actually, also underspecified. Examples of such "underspecification" include the proper behavior with respect to DAO-ACKs and DAO retransmissions. The strength of the DODAG mechanism is that, by way of trickle timers, DIO emissions automatically taper off as the network becomes stable. For DAOs, a "best guess" – as this is not specified – is for these to be periodic. Similarly, RPL specifies two incompatible modes for such *downward* routing: *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.

An example of an *active reading* deployment is "smart metering", where a utility company wishes to interrogate

individual consumption of its clients: each reading is initiated by the controller requesting the meter reading and the meter replying with the current consumption. Typically, a utility company will wish to read "several thousand meters" – but that over the course of a day (or more). Thus, all downward routes need not be *immediately* available – nor permanently maintained – but need be discoverable when needed.

Protocols so discovering routes on demand exist, such as AODV [14] and a proposed simplification for LLNs entitled LOAD [15]; neither of which requires source-routing, however both may require more state than a single routing table entry in each router. It should be noted that the RPL *storing-mode*, which also eliminates the need for source-routing, likewise does not have a bounded state. In an *active reading* deployment, however, this state requirement may, in the case of a protocol such as LOAD or AODV, be reasonably managed by the controller by way of spacing readings appropriately so as to keep the number of active concurrent routes (and, so, the state required in each router) below a threshold supported by each LLN router.

While evaluating the complexity of a protocol based on the complexity of its specification isn't entirely fair, it is still indicative that while the RPL specification counts 159 pages, and depends on several other specifications (trickle, metrics, 6lowpan routing header, etc.), LOAD counts 17 pages and AODV (published as RFC) 37 pages. One possible solution to the problem of complexity could be to "modularize" RPL, *i.e.*, extract different modules (such as for upward data traffic, storing mode, non-storing mode) into separate specifications with a common framework that allows implementations to select which modules to implement, and to define a mechanism to assure that two interoperating implementations provide the same modules, or at least stipulate which modules a given implementation contains.

A legitimate question, from the position of *active reading*, is therefore what the extra complexity of implementing RPL brings. A further legitimate question to ask is what resulting protocol would have emerged, had the design-basis been *active reading* rather than *data acquisition*.

A consequence of the above could be to suggest that a possible LLN routing protocol solution would be modular, consisting of:

- the DODAG formation mechanism from RPL, specifically the DIO and Trickle components;
- an on-demand route discovery mechanism, for example derived from LOAD or AODV.

With the ability to combine these, as needed, for a given deployment/scenario, it might provide a both elegant, efficient and simple routing solution.

REFERENCES

[1] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," February 2011, Internet Draft, work in progress, draft-ietf-roll-rpl-18.

[2] 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.

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

[4] J. Hui, J. Vasseur, D. Culler, and V. Manral, "An IPv6 Routing Header for Source Routes with RPL," October 2010, Internet Draft, work in progress, draft-ietf-6man-rpl-routing-header-01.

[5] 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.

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

[7] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," January 2011, Internet Draft, work in progress, draft-ietf-roll-trickle-08.

[8] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics used for Path Calculation in Low Power and Lossy Networks," January 2011, Internet Draft, work in progress, draft-ietf-roll-routing-metrics-17.

[9] P. Thubert, "RPL Objective Function 0," January 2011, Internet Draft, work in progress, draft-ietf-roll-of0-05.

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

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

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

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

[14] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," July 2003, Experimental RFC 3561.

[15] K. Kim, S. D. Park, G. Montenegro, S. Yoo, and N. Kushalnagar, "6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)," June 2007, Internet Draft, work in progress, draft-daniel-6lowpan-load-adhoc-routing-03.