Network Working Group                                         H. Audeoud
Internet-Draft                                                 M. Heusse
Intended status: Informational                                       LIG
Expires: September 12, 2019                               March 11, 2019

# Experimental observation of RPL: routing protocol overhead and asymmetric links
**draft-audeoudh-rpl-asymmetric-links-00**

Abstract

This document summarizes our observations of the behavior of RPL on a testbed composed of tens of IEEE 802.15.4 nodes. Our first observation is that the continuous task of estimating the link metric to all candidate neighbors causes a significant background load. This traffic is persistent, even in a stable network where DIO transmissions are eventually widely spaced. Next, this document focuses on the case of the presence of an asymmetric link, due to either a muted or a deaf node. In these circumstances, the standard RPL mechanisms may well generate hundreds of routing messages per node and per hour.

## 1. Introduction

We present three cases in which the RPL protocol [RFC6550] incurs a large number of routing message transmissions even though they correspond to expected situations in LLNs. This memo summarizes our observations on RPL that are part of a broader set of experiments [EXPE].

### 1.1. Convergence and background traffic

The maintenance traffic in RPL converges to a low rate of DIO generation when the topology is stable. Nevertheless, the proactive approach of RPL imposes that the nodes permanently gauge potential new alternative neighbors. This mechanism is not standardized, but it is necessary. Users need to be aware of its existence and its footprint.

### 1.2. Asymmetric links

The quality of radio transmissions depends on the environment and on the radio hardware. In particular, interferences do not have the same impact on all nodes. Also, the transmission conditions are different between devices due to the variability of the amplifier gain and sensitivity.

So a link between two nodes may be asymmetric, and not present the same packet delivery ratios (PDR) in both directions. In extreme cases, a node N may be "deaf" (i.e. other nodes receive N's packets, but N does not receive anything back) or "muted" (i.e. N receives properly, but is not heard).

### 1.3. Experimental setup

Table 1 presents the parameters used in the experiments. The trickle settings match the recommended practice, with Imin more than one order of magnitude greater than the broadcast duration (125 milliseconds in ContikiMAC). We found that this setting effectively avoids DIO collisions in our experiments.

| Parameter | Value |
|---|---|
| Platform | IoT-lab |
| Sensors | IoT-lab's M3 motes (ARM Cortex M3 STM32F103REY) |
| Sensor radio | 802.15.4 @2.4GHz (AT86RF231) |
| OS & RPL implementation | Contiki 3.0 |
| Radio Duty Cycling (RDC) | ContikiMAC |
| RDC Check interval | 125 ms |
| RPL Mode of Operation | Storing |
| Imin | 4 seconds |
| Imax | 1048 seconds |
| DIORedundancyConstant | 10 (standard's default) |
| DAO re-generation period | 15 to 22 minutes |
| Objective function | MRHOF with ETX |
| # UDP traffic intensity per client | 1 request-response every 5 minutes |

Table 1: Main parameters used during the experiments

## 2. Background traffic in a standard scenario

On the IoT-LAB testbed, we start 40 client nodes and a sink during 2 hours. Each client sends one UDP packet every 5 minutes to the sink which replies with another UDP packet. For this experiment, the nodes are distributed in the DODAG at a mean distance of 3.1 hops to the sink (median of 3). There are 9 nodes at a distance of 1 hop, and 5 nodes at a distance greater or equal to 6. Table 2 presents the results numerically.

- o The bulk of multicast messages are DIOs sent by the nodes following the trickle algorithm. As the network is stable, the network-wide multicast DIO generation intensity reduces gradually until all nodes use a DIO interval of Imax (which gives an average

period of approximately 750s in the considered case, approx. 1 message every 20s for our 41-node network).

- An intense and continuous traffic of unicast DIOs. RPL makes no provision for Unicast DIOs; it turns out that these messages are used by Contiki to assess the link quality to the neighbors [FNINES].

So this latter traffic is a re-use of DIO messages for a task which is out of the scope of RPL. The objective of this document is not to discuss the arguments for and against this specific mechanism in Contiki. Nevertheless, such a mechanism is necessary to RPL: "RPL expects an external mechanism to be triggered during the parent selection phase in order to verify link properties and neighbor reachability" [RFC6550]. Also RFC 6719 states that: "A node should compute the path cost for the path through each candidate neighbor reachable through an interface" [RFC6719].

At the application layer, it works well; but the price to pay is very significant, even counting the same overhead for broadcast and unicast messages. There are 4869 RPL message exchanges over the course of the experiment. In our special case of an application traffic which consists in one request response from each end node to the sink, we witness only 5516 one-hop data message transmissions, so that about half of all MAC-layer frames are linked to the routing processes.

| Message | # of occurrence |
|---|---|
| DIS multicast | 26 |
| DIO multicast | 760 |
| DIO unicast | 2497 |
| DAO (unicast, counted on each hop) | 1407 |
| DAO No-Path (unicast, counted on each hop) | 179 |
| Data packet successfully routed end-to-end | 1795 (97%) |
| Data packet emitted (counted on each hop) | 5516 |

Table 2: Number of messages sent during the experiment of background traffic

## 3. In presence of deaf nodes

When a RPL node is not associated with any DODAG, it broadcasts a DIS message. This message resets the trickle timer that schedules the DIO message emissions in its neighborhood, thus generating many DIO broadcasts. Although this mechanism is useful to speed up the insertion of new nodes into the network, it is very harmful when a node N is deaf. Indeed, as N keeps broadcasting DIS packets, its neighbors constantly send back DIOs.

On the IoT-LAB testbed, we start 11 client nodes and a sink during 1 hour. Each client sends one UDP packet every 5 minutes to the sink which replies with another UDP packet. One of the client nodes is deaf: its sensitivity is lowered and it does not receive any message from other nodes, whereas its packets are received by its neighbors.

Table 3 presents the results numerically. DIO packets are constantly broadcast at the average rate of 1 every 3 seconds for the whole network, i.e. approximately 2 broadcast packets per node per minute. This intensity per node is one order of magnitude more than in the previous experiment.

This scenario is not necessarily only a routing protocol problem, but in part a question of neighbor management. One could expect that the MAC layer would eventually detect that a neighbor is unresponsive and blacklist it, for instance. However, here, the detection of the asymmetry could not directly be done by the MAC layer, as DIS and DIO packets are broadcast and thus, not acknowledged. There should be a safeguard mechanism in RPL to break out of the DIS/DIO cycle in these circumstances.

| Message | # of occurrence |
| --- | --- |
| DIS multicast (deaf node excluded) | 5 |
| DIO multicast | 1177 |
| DIO unicast | 315 |
| DAO (unicast, counted on each hop) | 202 |
| DAO No-Path (unicast, counted on each hop) | 40 |
| Data packet successfully routed end-to-end (deaf node excluded) | 233 (99%) |
| Data packet emitted (counted on each hop, deaf node excluded) | 583 |

Table 3: Number of messages sent during the experiment with the deaf node

## 4. In presence of muted nodes

We finally consider the case of a node that loses the ability to reach some neighbors and, in particular, its preferred parent. On the IoT-LAB testbed, we start 11 client nodes and a sink during 1 hour. Each client sends one UDP packet every 5 minutes to the sink which replies with another UDP packet (there is no deaf node here). Obviously, if we completely mute one node from the start, it will not disturb its neighbors much as all its messages would be lost. So we start a node (node 40 of Figure 1) with a regular setting and let it associate itself with the network and communicate with other nodes. Then, after 15 minutes, we reduce its transmission power: it is muted and cannot reach its former neighbors anymore -- except one of them, node 33, so that it is not totally isolated from the rest of the network. After about 10 minutes, the network is completely repaired and nodes are back to transmitting control messages at a normal rate.

```
                 (45)                           (45)
                /  | \                             | \
               /   |  \                            |  \
           (40)  (12)  (18)                     (12)  (18)
             |     | \                             | \
             |     |  \                            |  \
           (33)  (48)  (10)                     (48)  (10)
                /  | \     \                    /  | \     \
               /   |  \     \                  /   |  \     \
           (57)  (55)  (30)  (51)          (57)  (55)  (30)  (51)
                     \                          /    \
                      \                        /      \
                     ( 2)                  (33)      ( 2)
                                             |
                                             |
                                           (40)

              Before node 40 is              After node 40 is
                   muted                         muted
```

Figure 1: DODAG for the muted node experiment

Here, the scenario clearly shows the effect of a local repair of the DODAG. Indeed, the link between node 33 and node 40 should be reversed to reach the DODAG root again. We have a transient loop in the network until node 33 is able to use node 55 as a successor: node 40 uses node 33 as a successor, which in turn uses node 40 as a successor -- but note that the traffic does not loop around, thanks to the data path validation mechanism based on the RPL header.

Table 4 presents the results numerically. This experiment shows that RPL handles this situation correctly: few data packets are lost and the DODAG repair is relatively quick. Nevertheless, a naive user could expect that only a few additional packets would be generated for such a limited topology change as for most routing protocols. But the repair traffic observed in this experiment is merely a consequence of the constant background routing overhead that the proactive approach and trickle imply.

| Message | # of occurrence between 15th and 25th minute | # of occurrence except between 15th and 25th minute | # total |
|---|---|---|---|
| DIS multicast | 0 | 3 | 3 |
| DIO multicast | 37 | 172 | 209 |
| DIO unicast | 58 | 313 | 371 |
| DAO (unicast) | 41 | 258 | 299 |
| DAO No-Path (unicast) | 11 | 40 | 51 |
| Data packet successfully routed end-to-end | 37 (90%) | 220 (99%) | 251 (98%) |
| Data packet emitted (counted on each hop) | 103 | 598 | 701 |

Table 4: Number of messages sent during the experiment with the muted node

## 5. Conclusion

First, we would like to emphasize the robustness of the Contiki implementation of RPL, which also matches the RFC specification to the extent of the points we checked. Our choice of Imin is constrained by the broadcast transmission duration; however, the choice of values Imax and DIORedundancyConstant could cause what one could consider excessive DIO traffic over the long run, but this is not the subject of our observations.

Secondly, the set of RPL-related documents leave aside the necessary mechanism of link metric estimation. This is not unexpected as it can be done without raising interoperability issues. Nevertheless, users need to be aware of this necessity e.g. in the case of power-constrained nodes. Also, as this task is closely related to the routing process, it is tempting and practical to use routing packets to the purpose of link metric estimation, as in Contiki. In this specific case, it is unknown if all implementations tolerate to receive unicast DIO, for instance. A possibility could be to reserve

a packet format for metric estimation or explicitely allow a possible side-use of existing packets.

Pushing this line of thought a bit further, it could be of interest to integrate (or combine) the broadcast DIS/DIO exchange to the link metric estimation.  This would allow the nodes to detect strongly asymmetric links at an early stage, and treat the messages from the corresponding neighbor knowingly.

## 6. IANA Considerations

This document includes no request to IANA.

## 7. Security Consideration

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

## 8. Acknowledgments

The authors would like to thank Dominique Barthel for encouraging us to write this memo, for his guidance and his feedback.

## 9. References

### 9.1. Normative References

[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>.

[RFC6719] Gnawali, O. and P. Levis, "The Minimum Rank with Hysteresis Objective Function", RFC 6719, DOI 10.17487/RFC6719, September 2012, <https://www.rfc-editor.org/info/rfc6719>.

### 9.2. Informative References

[EXPE] Audeoud, H. and M. Heusse, "Experimental Comparison of Routing Protocols for Wireless Sensors Networks: Routing Overhead and Asymmetric Links", in proceedings of the 29th International Teletraffic Congress (ITC 29), Genoa, DOI 10.23919/ITC.2017.8064339, 2017, <http://hal.univ-grenoble-alpes.fr/hal-01609451/document>.

[FNINES] Duquennoy, S., Eriksson, J., and T. Voigt, "Five-Nines Reliable Downward Routing in RPL", arXiv 1710.02324, 2017, <https://arxiv.org/pdf/1710.02324.pdf>.

Authors' Addresses

Henry-Joseph Audeoud
Grenoble Informatics Laboratory

Email: henry-joseph.audeoud@univ-grenoble-alpes.fr

Martin Heusse
Grenoble Informatics Laboratory

Email: martin.heusse@univ-grenoble-alpes.fr