Is RPL Ready for Actuation? A Comparative Evaluation in a Smart City Scenario

Timofei Istomin[1], Csaba Kiraly[2], and Gian Pietro Picco[1]

[1] DISI – University of Trento, 38123 Povo, Trento, Italy
{timofei.istomin,gianpietro.picco}@unitn.it
[2] Bruno Kessler Foundation, 38123 Povo, Trento, Italy
kiraly@fbk.eu

Abstract. Low-power wireless actuation is attracting interest in many domains, yet it is significantly less investigated than its sensing counterpart, especially in large-scale scenarios. As a consequence, guidelines about which protocol, among the few existing ones, is best suited to a given scenario are generally lacking.
In this paper, we investigate the relative performance of simple dissemination-based solutions against the standard, state-of-the-art RPL protocol. These choices of protocols are motivated concretely by our involvement in the deployment of a large-scale infrastructure for smart city applications, which directly informs our evaluation, where we use the actual network topology.
Our findings, albeit in a specific scenario, suggest that RPL still leaves much to be desired w.r.t. actuation. Two out of the three RPL implementations we considered exhibited unacceptable performance when used out-of-the-box. Even after some tuning and debugging, simple, dissemination-based solutions perform surprisingly better under several conditions. These findings motivate further research on the topic of large-scale low-power wireless actuation.

1 Introduction

The growing importance of cyber-physical systems, where the target environment is augmented with small devices able to sense and actuate according to the application logic, has brought low-power wireless networks to the forefront as an enabling technology. Nevertheless, although wireless *sensing* has been a popular research topic in the last decade, wireless *actuation* has received considerably less attention. As a result, not only there are fewer proposals in this latter realm, but also noticeably less common knowledge about the protocol tradeoffs, especially when applied to a real scenario.

Goal and motivation. The work we present here stems from this observation, and was prompted by a concrete necessity. Our research team was sought after for collaboration by a company deploying in Trento, Italy, a large-scale wireless infrastructure of 860+ IEEE 802.15.4 nodes, for monitoring and control of public lighting and other "smart city" applications. Our task was to improve the current network stack which is based on simple *flooding*, by identifying an existing solution providing better performance, to be used in the final deployment. *"Beating flooding: that's going to be a piece of cake"* we thought cockily—a thought probably shared by many readers. This paper shows instead that the winner is not so clear. As our study is based on the deployment topology and

scale for a smart city—a scenario at the forefront of today's technological trends—our findings raise questions about the state of the art of low-power wireless actuation.

Protocols under study. We chose RPL [13] as the main candidate because it is a standard and provides interoperability with mainstream Internet technology. Moreover, it is designed to support both the many-to-one traffic typical of sensing and the unicast or multicast one-to-many necessary to large-scale actuation. Several implementations of the standard exist, which bear significant differences [9]. We focused on TinyRPL [16] and ContikiRPL [15], arguably the most popular implementations available. We include also ORPL [3], which aims to improve the scalability of downward routing of RPL.

The baseline for our comparison is the flooding protocol currently operational in our reference smart city deployment. It implements a simple scheme, in which nodes repeat incoming messages once, after a small random delay, using link-level broadcast. A history of seen messages and time-to-live (TTL) are used to filter duplicates and avoid loops. We also included Trickle in our comparison as another representative of *dissemination* protocols. In fact, protocol complexity was an issue for the company, therefore, Trickle constitutes an alternative to flooding less radical than RPL.

Scenario and methodology. We cast our comparison in the real-world scenario above, leveraging the first-hand information we can obtain from it. The planned deployment comprises 864 nodes on lampposts, divided into 13 clusters whose size is 25 to 134 nodes, each with a dedicated gateway to the Internet. Peculiar network topologies determined by the urban structure and radio interference properties make this scenario different from the indoor testbeds typically employed to evaluate protocol performance. Since nodes on lampposts are mains-connected, a duty-cycling MAC is not necessary.

Nevertheless, we do not have access to the actual infrastructure deployment, and cannot perform protocol experiments directly on it. Simulation is essentially the only option to perform our comparison. The use of simulation has well-known drawbacks, e.g., the approximations made w.r.t. the radio channel. In our study, in the absence of radio models or experimental traces expressly targeting a smart city environment, we resort to the MRM model provided by the Cooja emulator and commonly used by the literature. However, we also aim to reproduce the interference and background noise present in diverse urban environments, based on noise measurements we acquired in several locations, described in Section 2 along with simulation settings.

Related work. Most experimental studies of RPL explore its data collection performance and topology stability [5, 7, 9, 12]; only a few deal with one-to-many traffic required for actuation. Authors of [8] study downward routing of TinyRPL in a 30 node indoor testbed and report results matching our observations for small clusters under low noise. The design of RPL downward routing is criticized in [1], although experiments are limited to many-to-one routing in an indoor testbed. The one-to-many routing of ORPL is shown to outperform RPL in an indoor environment [3], using ContikiMAC.

Other protocols like WirelessHART, ISA100.11a [11] and LWB [4], although in principle relevant to our study, were excluded due to lack of support by simulators.

Findings. Section 3 presents the results of our study, which is geared towards answering a very simple question: *"Is RPL ready for actuation?"* Based on the results, Section 4 formulates an answer, which is not a positive one. Finally, we end the paper with brief concluding remarks, including opportunities for future work on the topic.

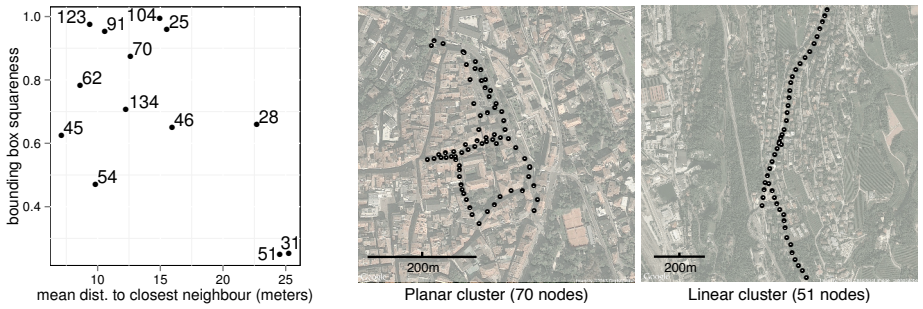

Fig. 1. Summary statistics about the geometry of the topology of all the 13 clusters (left) and topology of two representative clusters (right). Note the different scale of the maps.

2 Simulation Settings

We base our study on Cooja [10], which supports both Contiki [15] based implementations and others (e.g., TinyRPL) thanks to its hardware emulation feature.

Topologies. The left side of Fig. 1 shows a comparison of cluster geometries, characterized by three metrics: *i)* number of nodes in the cluster (point label); *ii)* distance to the closest neighbor, averaged over all nodes (x-axis); *iii)* aspect ratio of a bounding box aligned with the largest span, indicating how "linear" a cluster is (y-axis). The right side of Fig. 1 shows the topology of two representative clusters.

Signal propagation model. We base our simulation on Cooja's multi-path ray tracing model (MRM). It models radio hardware properties, background noise and interference through signal-to-interference-and-noise ratio (SINR), the capture and multi-path effects; however, its simplistic obstacle model is not sufficient to define the complex architecture of a city. We configure MRM based on the popular CC2420 radio chip.

Modeling noise. The background radio noise (including ambient and man-generated effects) directly influences signal-to-interference-and-noise ratio (SINR), thus also radio reception range and protocol performance. The noise floor in a dense urban environment can be relatively high and with high short-term variations.

This is in contrast with works assuming a noise-free environment (e.g., [14]), and with the conditions found in the testbeds commonly used in experiments. To verify this statement we performed measurements on all IEEE 802.15.4 channels in the Indriya [2] and TWIST [6] tesbeds as well as in several places of Trento and Moscow, including suburbs, densely inhabited areas and the university campus. The testbed measurements show a mean noise floor of -90 to -98 dBm and a standard deviation of 2–4 dBm, depending on the node and channel. In the cities, the mean noise floor is usually -85 to -95 dBm (occasionally up to -75 dBm), and the standard deviation is 0–10 dBm.

We use the notation $MRM(N_{avg}, N_{sd})$ to indicate an MRM model with noise floor N_{avg} and standard deviation N_{sd}. We also consider the *theoretical radio reception range*, an estimate calculated using Friis transmission equation based on N_{avg}, the parameters of the radio subsystem, and a transmission power of 0 dBm.

Table 1. Layer 3 parameters.

Protocol	L3 parameters	L2
Flooding	rebroadcast delay: 8–80ms	CSMA 1
Trickle	I_{min}=1/32s, I_{max}=1/2s, K=1	
ContikiRPL	routing table size: 70, routing metric: ETX	
TinyRPL		CSMA 2
ORPL	routing metric: EDC	RDC

Table 2. Layer 2 parameters.

Parameter	CSMA 1	CSMA 2	RDC
CCA backoff	128ms–1.2s	0.3–10ms	125–500ms
backoff increase	exponential	none	linear
no-ACK retry delay	as backoff	103ms	as backoff
duty-cycle interval	—		125ms
no-ACK TX attempts	5		
neighbour table size	20		60

Protocol settings. All the protocols under study are highly customizable through parameters such as buffer sizes, timeouts, retries and hop count. Wherever possible, we used the default values, as these are likely to be first choice in a deployment and the ones tested the most. We did, however, include tuned and modified versions of ContikiRPL and ORPL, since their initial results showed clear discrepancies w.r.t. expectations. The most important protocol parameters used in our study are summarized in Tables 1 and 2.

Application setup and performance metrics. We test protocol performance in sending commands from the gateway to other nodes in the cluster, as in the current infrastructure. Messages have a 6 B payload, enough to fit a command code and 1–2 parameters. As actuation commands are issued sparingly, we focus on the reliability and timeliness of delivering isolated commands, rather than scalability in terms of traffic load.

In each experiment, after a warm-up time needed to stabilize logical topology, the gateway sends $B = 2000$ isolated commands, each destined to a node chosen with uniform random selection, with an inter-command interval (ICI) of 5 s. For statistical relevance, simulations are run 5 times per set of parameters; the plots report the average value along with error bars denoting the minimum and maximum values. Reliability and timeliness are quantified by measuring the *packet delivery ratio (PDR)* and average *delivery delay* for each destination and further averaging over all nodes of the cluster.

We also consider the *network utilization per actuation command*, expressed in bytes sent over the radio. To compute it we sum up the data and control traffic transmitted after the warm up and normalize w.r.t. the number of actuation commands sent.

3 Results

We compare the selected protocols in different radio propagation environments and network topologies. Space limitations force us to show the effect of noise floor only for the two distinctive clusters presented in Fig. 1: a 70-node "planar" one and a 51-node "linear" one. The noise variance was set to $N_{sd} = 1$ dBm. We then focus on two interesting noise configurations, and analyze the impact of topology and scale, showing results for all the clusters.

Debugging ContikiRPL and ORPL. In our first trials, ContikiRPL showed lower performance than expected. Log inspection identified routing table management as the culprit. When a node rejoins through another DODAG branch, the next-hop entry at the branching point is not updated until it expires. Traffic along the stale path causes routing errors, triggering unnecessary DODAG reconstructions through version increase. A

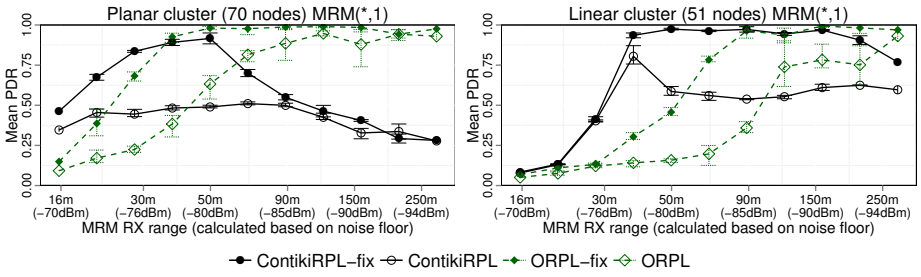

Fig. 2. PDR achieved by ContikiRPL, ORPL, and their debugged variants.

vicious circle is formed: version increase causes churn, churn brings routing errors and version increase. Our fix to this issue has been merged into the Contiki code base.

The default ORPL configuration also performed poorly. A custom one, with bitmap filters and a 125 ms ContikiMAC period showed better performance, though degrading over time. The so-called false positive mechanism stalled nodes; we disabled it, as it is anyway useless with bitmap filters. We also disabled the bitmap ageing mechanism and modified the input filters in the reception path to solve occasional memory corruptions.

The effect of our modifications is evident in Fig. 2, showing PDR as a function of radio range (noise floor) on our reference clusters. Our modified implementations match or outperform the original ones on all clusters and for all the metrics, often with remarkable performance gains. Therefore, hereafter they are the only ones we report.

Impact of noise floor. Fig. 3 shows the protocol performance as a function of noise floor, similarly to Fig. 2 but this time with all metrics and for all studied protocols.

From a reliability perspective, only Trickle performs well in all cases where the graph is still connected. This is expected, since it is the only protocol enjoying unlimited retransmissions; in case of an isolated message, sooner or later Trickle delivers.

On the planar cluster, ContikiRPL-fix handles high noise better than flooding. In this situation, the radio range is so small that nodes have only few neighbors with weak links; the L2 retransmissions of ContikiRPL-fix are more effective than the multi-path properties of flooding. Nevertheless, below -75 dBm, as the improved range and link quality makes link-local broadcasts more efficient, flooding takes the lead. On the linear cluster, ContikiRPL-fix provides good results below -77 dBm, against the -80 dBm of flooding. However, ContikiRPL-fix never reaches $PDR = 100\%$, even in medium-noise scenarios where its competitors do, and shows poor performance at low noise, especially on the planar cluster. The increased radio range makes the network denser, overfilling the neighbor tables; further, when the next-hop node is not found among the neighbors, a global DODAG repair is triggered. ORPL-fix delivers less than ContikiRPL-fix at high noise, but it does not suffer from higher density, and performs in line with dissemination protocols when noise is -80 dBm in the planar cluster, and -85 dBm in the linear one. TinyRPL follows in the ranking, consistently on both clusters. It is not affected by high network density since it does not rely on the neighbor table to resolve the next-hop link-local address for a given target. Instead, the address is obtained directly as the last two octets of the link-local IPv6 address of the neighbor.

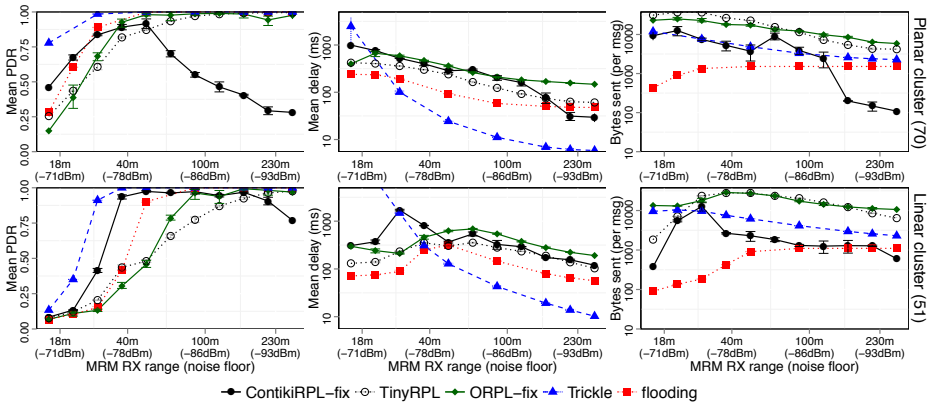

Fig. 3. Effect of noise floor on protocol performance: PDR, delay, and network utilization (columns) on the two selected clusters (rows).

Regarding the *average delay* of packet delivery, differences were so significant that we had to resort to a logarithmic scale. Trickle achieves best performance in low-noise situations. At high noise, it is the only protocol with high PDR values, which makes its seemingly larger average delay non-comparable to other protocols. Flooding and TinyRPL follow with a significant increase in delivery times. ContikiRPL-fix and ORPL-fix close the ranking, with delays between 500 and 800 ms. For ORPL-fix this higher delay is partly justified by the use of the duty-cycling MAC.

For evaluating *network utilization*, a logarithmic scale was again required. For flooding, the metric is simply proportional to the cluster-level average PDR, since each node that receives a packet repeats it exactly once. Trickle is heavier than flooding due to its retransmissions. Although flooding involves the whole network when delivering a single packet, its network utilization is still less than RPL variants under most conditions, except for very low noise situations. Indeed, RPL cannot amortize the topology maintenance cost under the (realistic) traffic properties used in our tests.

All clusters: trends and effects of topology. After exploring the influence of the noise floor, we demonstrate the effects of topology characteristics under typical low and high noise scenarios, $MRM(-90, 2)$ and $MRM(-85, 3)$ respectively. Fig. 4 shows the results for all clusters. Note that we use the number of nodes to identify them.

Both ContikiRPL-fix and TinyRPL show scalability issues as the number of nodes grows. Moreover, the selected radio model values fall outside the "comfort area" of ContikiRPL-fix, triggering its node density issues, for which it provides the worst PDR of all protocols. ORPL-fix, on the other hand, reaches high PDR, thanks to the noise level below -85 dBm. Other trends are the high PDR and almost flat delay recorded for flooding and Trickle regardless the number of nodes. Network utilization instead increases almost linearly with the number of nodes for all protocols.

There are, however, outliers from the above trends; some are correlated to topology characteristics, as in the case of the three clusters marked by asterisks which have "special" topologies. Clusters 31 and 51 are sparse and long, while cluster 28 is U-

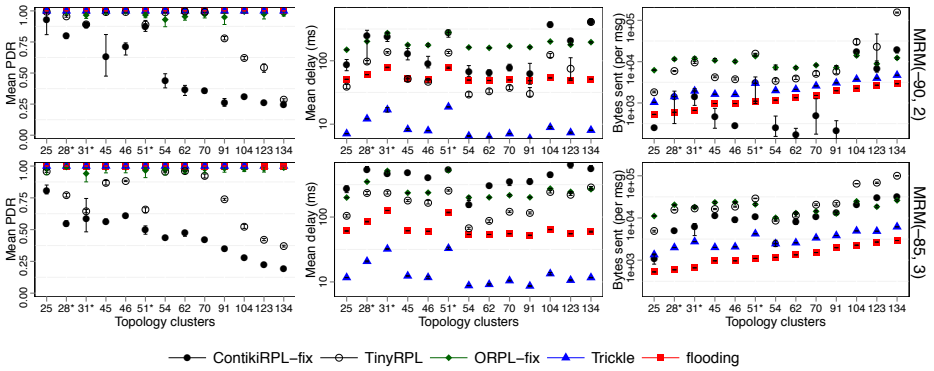

Fig. 4. Results for all the clusters. PDR, delay and network utilization (columns); low noise and high noise scenarios (rows).

shaped with the two long branches behaving like the linear topologies. In these clusters, delays are increased for all protocols due to larger hop-counts in paths. Trickle and TinyRPL also shows increased network utilization, while flooding is not affected. The performance of TinyRPL is decreased in these special clusters (can only be seen in high noise), but it is not clear whether larger node distances or longer multi-hop paths are to blame. ContikiRPL-fix instead shows a performance increase but only at lower noise, and in this case we know it is due to the decreased neighborhoods.

4 Discussion, Conclusions, and Future Work

Our involvement in the design of the network stack for a smart city infrastructure was the opportunity to study the applicability of RPL and its variants to the problem of large-scale low-power wireless actuation. We performed our study by simulation, borrowing the actual placement of nodes to experiment with a real network scale and topology.

We were convinced that flooding was going to be left in the dust by RPL; our results tell a different story. The RPL variants were outperformed by flooding (and Trickle) especially in low-noise conditions (or high range and therefore neighbor density), where RPL suffered from various reasons, including scalability issues and topology reconstructions leading to increased packet losses. ContikiRPL-fix outperforms flooding, thanks to link-level retransmissions, in situations with high noise (low density) where long multi-hop paths are required to deliver messages, e.g., on linear topologies. However, in these cases the other RPL variants are outperformed by flooding, with significant differences among the various implementations. Further, the simple dissemination scheme of Trickle outperforms ContikiRPL-fix, and appears to be the best choice.

Trickle is also the fastest protocol, while flooding is the one with the lowest network utilization, leaving few reasons to choose RPL over our dissemination baseline. To be fair to RPL, the overhead of maintaining its topology is expected to be amortized by the many-to-one data collection traffic (not considered here) for which it is optimized, and "reused" for actuation. Our results, however, show that this reuse falls

short of expectation, suggesting that a dedicated and complementary solution, possibly dissemination-based, should be used for the relatively lower-traffic of actuation.

These RPL shortcomings are exacerbated by implementation considerations, beyond the difficulties we encountered in using ContikiRPL and ORPL out of the box. The superior performance of dissemination protocols is complemented by their simplicity, yielding less demands in terms of memory consumption. In fact, the studied RPL implementations occupied almost all RAM and Flash memory of the popular TMote Sky.

In summary, the verdict is against RPL. Aside from relatively immature RPL implementations, it is hard to beat the simplicity and robustness of dissemination protocols.

There are obvious opportunities for future work on the topic of this paper: our findings are specific to our smart city target scenario, and should be validated in other kinds of large-scale actuation scenarios, possibly through real-world experiments. Finally, this paper poses a research question, namely, whether low-power wireless actuation really needs the complexity of maintaining a routing topology, or instead dissemination protocols should be the foundation to be optimized towards this functionality.

Acknowledgments. This work was partially funded by EIT ICT Labs (Activity 12149) and by Algorab S.r.l., which also provided information about the smart city deployment.

References

1. T. Clausen, U. Herberg, and M. Philipp. A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL). In *Proc. of WiMob*, 2011.
2. M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed. In *Proc. of TRIDENTCOM*. Springer, 2011.
3. S. Duquennoy, O. Landsiedel, and T. Voigt. Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *Proc. of SenSys*, 2013.
4. F. Ferrari, M. Zimmerling, L. Thiele, and L. Mottola. The low-power wireless bus. In *Proc. of IPSN*, 2012.
5. O. Gaddour and A. Koubâa. RPL in a nutshell: A survey. *Computer Networks*, 56(14), 2012.
6. V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST. In *Proc. of REALMAN*, 2006.
7. O. Iova, F. Theoleyre, and T. Noel. Stability and efficiency of RPL under realistic conditions in wireless sensor networks. In *Proc. of PIMRC*, 2013.
8. J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. Evaluating the Performance of RPL and 6LoWPAN in TinyOS. In *Proc. of the IP+SN Workshop*, 2011.
9. J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. ContikiRPL and TinyRPL: Happy together. In *Proc. of the IP+SN Workshop*, 2011.
10. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proc. of LCN*, 2006.
11. S. Petersen and S. Carlsen. WirelessHART versus ISA100.11a: the format war hits the factory floor. *IEEE Industrial Electronics*, 5(4), 2011.
12. I. Radoi, A. Shenoy, and D. Arvind. Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks. In *Proc. of ICWMC*, 2012.
13. T. Winter et al. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550.
14. M. Z. Zamalloa and B. Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Trans. on Sensor Networks (TOSN)*, 3(2):7, 2007.
15. Contiki operating system official website. `http://contiki-os.org`.
16. TinyOS official website. `http://tinyos.net`.