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

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, yet robust dissemination based solutions (e.g., flooding) against the actuation support of 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 (860+ nodes) for smart city applications, arguably one of the most promising application domains for low-power wireless actuation. This deployment 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 originally prompted by a very concrete necessity. Our research team was sought after for collaboration by a company deploying a large-scale low-power wireless infrastructure in a city, constituted by 860+ IEEE 802.15.4 nodes mounted on lampposts. The goal is to support remote monitoring and control of public lighting, but other "smart city" applications are envisioned to run on this infrastructure. The collaboration objective stated by the company was to improve their current network stack supporting the target applications. We soon found out that this stack is very rudimentary: each application command is simply *flooded*, with a few mechanisms to mellow the effect of collisions. The company engineers motivated the choice with the need to have a simple yet robust solution, which could be rapidly deployed as a demonstrator in small-scale installations. Our goal was to identify an existing solution providing better performance, to be used in the final, large-scale deployment. *"Beating flooding: that's going to be a piece of cake"* we thought cockily—a thought probably shared by many of the readers. This paper shows that, in practice, the winner is not so clear. Further, as the playing field of our protocol competition is the topology and scale of a real smart city deployment—a

scenario at the forefront of current technological trends—we believe our findings raise important questions about the state of the art in low-power wireless actuation.

Protocols under study. Although the literature on wireless sensing is significantly broader than its actuation counterpart, a few protocols exist, which we concisely survey in Section 2, along with works related to ours.

Nevertheless, RPL [20] has been one of our candidate protocols since the beginning, due to several reasons, and is therefore the focus of this paper. First of all, it is a standard, and provides interoperability with mainstream Internet technology: both were very desirable properties from the company's viewpoint. Moreover, the RPL standard is designed to support both the many-to-one traffic characteristic of sensing and the unicast or multicast one-to-many necessary to large-scale actuation. For instance, in our context a dimming command is sent to a single lamp or to a group of lamps, determined automatically (e.g., based on environmental conditions or motion sensors) or by an operator (e.g., due to maintenance). In this respect, using a single protocol for both sensing and actuation was perceived as a plus by the company.

RPL is described in Section 3 along with our baseline, which includes the custom flooding protocol already deployed by the company and the well-known Trickle [15] protocol. The reason for including the latter is twofold. On one hand, Trickle can be regarded as a more efficient and reliable flooding, and another representative of the class of protocols enabling *dissemination* to the entire network. On the other hand, protocol complexity was an issue for the company; they were used to the simplicity of flooding, and reluctant to embark in the implementation of a protocol they did not design. Therefore, Trickle constituted not only another useful point of comparison, but also a less radical alternative to flooding.

In practice, however, several implementations of the RPL standard exist, which bear significant differences [13]. We focused on TinyRPL [23] and ContikiRPL [22], arguably the most popular implementations available for TinyOS and Contiki, respectively. Nevertheless, while the RPL standard specifies the "downward routing" supporting unicast and multicast traffic, essential to actuation, both these reference implementations focus on the efficient support for many-to-one traffic, in line with the mentioned asymmetry between sensing and actuation. Therefore, we include in our study other protocols. In addition to ORPL [5], which aims to improve the scalability of downward routing in RPL, we compare against TM [10] and SMRF [16] as they are expressly designed to support multicast atop Trickle and RPL, respectively, and allow us to investigate the performance gain they achieve w.r.t. these base protocols.

Scenario, methodology and findings. We cast our comparison in the smart city scenario which motivated the collaboration with the company, leveraging the first-hand information we can obtain from it. In particular, as we describe in Section 4 along with other details of the concrete scenario at hand, we have complete knowledge of the placement of the 864 nodes planned (many already deployed) and their grouping into 13 clusters served by independent gateways.

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. We base our study on the Cooja emulator [17], which

allows us to execute the exact code of the selected protocol implementations; testing the many combinations under consideration in real testbed would take a remarkable effort.

The use of simulation has also some 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 Cooja, but also experiment with settings that aim to reproduce the presence of interference and background noise present in the urban environment we target. We discuss the simulation setup and the chosen performance metrics in Section 5.

Section 6 discusses the results of our study, which is geared towards answering a very simple question: *"Is RPL ready for actuation?"* The answer is not a positive one. Pragmatically, two out of the three popular RPL implementations we used required significant tuning and debugging before reaching acceptable performance. In general, and even with these improved versions, the baseline dissemination protocols we consider perform surprisingly better under several conditions, despite their inherent simplicity. We summarize our findings and their practical implications in Section 7.

Finally, Section 8 ends the paper with brief concluding remarks, including opportunities for future work on the topic.

2 Related Work

Although multiple performance studies have been conducted for the protocols discussed in this paper, up to our knowledge, they were never confronted all together under a broad range of conditions and topologies specific for urban networks.

Most experimental studies of RPL explore its data collection performance and the topology stability [7, 11, 13, 19] and only few works deal with one-to-many traffic required for actuation. The authors of [12] study the downward routing of TinyRPL in an indoor testbed of 30 nodes and report results that match our observations for small clusters under low interference. In [1] the design of downward routing of RPL is criticized and the negative impact of the noise variation on the DODAG stability is reported, though the presented experimental evaluation is limited to many-to-one routing in an indoor testbed. One-to-many routing performance of ORPL and RPL is studied in [5] in the 135-node indoor Indriya testbed, using two TX power levels to modify density of the network, showing that ORPL outperforms ContikiRPL. Contrary to our scenario, RPL is used with ContikiMAC. The multicast protocols, TM and SMRF, are evaluated in [16] both in simulation and on real hardware. The simulations were run on regular linear and tree-like synthetic topologies of up to 21 nodes and using a simplistic unit disk graph radio model. The real-world evaluation was limited to a network of 11 nodes.

Some other protocols, although interesting for the scope of this paper, were excluded from our study: *i)* an implementation of wirelessHART and ISA100.11a [18] have recently been open sourced, but it is hardware specific and cannot be used in simulations; *ii)* LWB [6] (and other constructive interference based protocols) cannot be realistically simulated with current tools due to its reliance on low level physical effects, moreover, it requires hardware homogeneity to operate correctly, a requirement deemed too stringent for a city-scale deployment.

3 The Contenders: Dissemination Protocols vs. RPL Variants

In this section we concisely describe the protocols we consider in this study. Protocols can be broadly classified into dissemination protocols, constituting the baseline for this study, and RPL variants. Moreover, due to the relevant role played by multicast in actuation, we also include extensions to Trickle and RPL.

Dissemination protocols. These protocols are not designed for actuation, yet, they can provide the necessary unicast or multicast capabilities by simply including the target address in the message transmitted by the sender and by delivering the message to the application only if such address matches the receiver's. In this class we consider:

- The *flooding* protocol currently operational in our reference smart city deployment. It implements a simple scheme, in which nodes simply repeat incoming messages after a small random delay, using link-level broadcast. There are no retransmissions, and duplicates are filtered by using a cache storing message identifiers, and a time-to-live (TTL) associated to messages to limit their rebroadcasting.
- *Trickle* [15] improves on flooding by rebroadcasting only when necessary (i.e., a message has been heard less than K times) and with an adaptive period that depends on the amount of new information to be disseminated. When a new message is received, the period is reset to the minimum, and doubled when there is no new data, until the maximum period is reached. This mechanism provides eventual delivery of the message to all nodes, a guarantee not provided by pure flooding.

RPL variants. *RPL* is part of the standard 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) stack [20]. It specifies support both for many-to-one routing, useful for data collection, as well as unicast and multicast one-to-many operation, useful for actuation. For many-to-one routing, RPL forms and maintains a DODAG (Destination-Oriented Directed Acyclic Graph) rooted at the sink node. This graph is optimized according to a so called Objective Function, which defines quality metrics for the calculation of a rank for each node in the network, and governs parent selection. For downward routing, possible destinations must periodically send Destination Advertisement Object (DAO) messages to establish and maintain a path from the root. In the so-called storing mode, state is maintained in routing entries along the path, while in non-storing mode, state information is confined to the root node and source routing is used. We study only the former mode, supported by the implementations we consider. The RPL standard allow for some freedom in its implementation. Therefore, it is important to compare specific implementations, as there are substantial differences even for the same protocol, as shown, e.g., in [13]. We consider three variants:

- *ContikiRPL* [22] is the reference implementation for Contiki. We use a daily build (2013-11-29), since the latest point release at the time experiments were conducted (version 2.6) showed unacceptably low performance.
- *TinyRPL* is the reference implementation for TinyOS;
- *Opportunistic RPL (ORPL)* [5] replaces the RPL path building mechanism with a hash- or bitmap-based one, adding also support for multiple downlink paths, and uses link-level anycast-based opportunistic forwarding [14]. The latter requires a duty-cycling MAC; we use ContikiMAC [4] as it is the only one supported.

Multicast extensions. Since support for multicast is essential for some applications involving actuation, we include the following extensions to the protocols above:

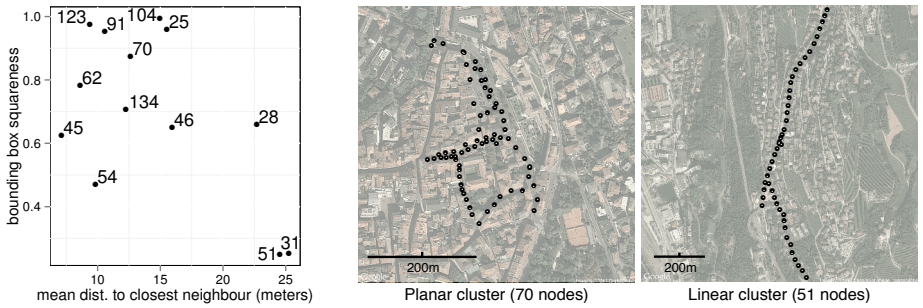

Fig. 1. Summary statistics about the geometry of all 13 topologies, and the topologies of two selected clusters. Note the different scale of the maps.

- *Trickle Multicast*[1] *(TM)* [10] extends Trickle with multicast addressing: since messages are anyway delivered to all nodes, joining a group is as simple as setting a local address filter. Multiple message source nodes are also supported, albeit not used in our case. Finally, TM changes Trickle's version based retransmission mechanism to a window-based one, mitigating the problem that a newer packet overwrites an older one if the latter was not yet delivered.
- *Stateless Multicast RPL Forwarding (SMRF)* [16] is based on ContikiRPL, and relies on the hierarchical logical topology built by RPL to deliver multicast messages along DODAG branches from the root down to the group members. A multicast message is handed from parent nodes to their children as 1-hop broadcasts without L2 acknowledgments or retries, resulting in less overhead but also lower reliability.

4 The Playing Field: A Smart City Deployment

Our network topology is based on a real-world city deployment on lampposts of 864 nodes, divided into 13 clusters with size between 25 to 134 nodes. For simplicity, we identify clusters based on the number of nodes contained. Each cluster has its dedicated gateway connecting the multi-hop WSN to the Internet. Commands arriving from a command center through reliable links are relayed by these gateways to the nodes.

There are several peculiarities of this installation compared to those found in research laboratories. First, topologies are determined by the urban structure, and could contain semi-regular structures such as long multi-hop stripes or circles. The left side of Fig. 1 shows a comparison of cluster geometries. We characterize clusters by three metrics: *i)* the number of nodes in the given cluster, shown by the point label; *ii)* the distance to the closest neighbor, averaged over all nodes, on the x-axis; *iii)* the aspect ratio of a bounding box aligned with the largest span, on the y-axis, indicating how "linear" is a cluster. The right side of Fig. 1 shows the topology of two selected clusters.

Second, power utilization is not a hard constraint, since nodes on lampposts are mains-connected. Therefore, our protocols do not have to rely on duty-cycling MACs.

Third, radio noise and interference from e.g. IEEE 802.11 could be different from the usual conditions of testbeds: it can be very high in a deployment with lampposts

[1] The open source implementation of TM is based on Draft 1 of the specification. In later versions TM was renamed to Multicast Protocol for Low Power and Lossy Networks (MPL).

Table 1. Radio parameters used in the MRM simulation model.

Parameter	Values	Notes
SNR reception threshold	6 dB	based on CC2420 characteristics
Receiver sensitivity	-95 dBm	based on CC2420 characteristics
Transmitter output power	0 dBm	maximum value for CC2420
Extra system gain: mean / std. dev.	0 dB / 2 dB	MRM default
Radio frequency	2400 MHz	studied ISM band
Capture effect preamble	64 μs	MRM default
Capture effect threshold	3 dB	based on CC2420 characteristics
Noise floor (N_{avg})	$-70 \ldots -95$ dBm	explored range; based on measurements
Background noise std. dev. (N_{sd})	$1 \ldots 10$	explored range; based on measurements

in front of large residential buildings with access points in every apartment, while it can be lower in another cluster with lampposts next to a suburban road. Therefore, we specifically deal with background radio noise in our modeling.

5 The Rules of the Game: Simulation Settings

We have selected Cooja [17] to drive our simulations as most of the protocol implementations we used are for the Contiki OS [22], for which Cooja provides full support. Furthermore, its hardware emulation features allow running the compiled application binary, allowing us to use also protocols (e.g., TinyRPL) developed for other OSes.

Signal propagation model. We base our simulation on Cooja's widely used multi-path ray tracing model (MRM). It models radio hardware properties, such as transmission power, radio sensitivity, and antenna gain. It also models the effect of background noise and interference on reception through signal-to-interference-and-noise ratio (SINR), and the capture effect. MRM is also capable of simulating multi-path effects if some obstacles are defined, however, it handles only a limited amount of square obstacles; clearly not sufficient to model the complex architecture of a city. Table 1 describes the radio parameters we used, and their ranges, based on the properties of the well-known TelosB platform and its CC2420 radio chip.

Noise floor and noise variance. The noise floor is essentially a measure of average background radio noise, including ambient effects such as thermal noise, but also radio noise generated by man-made systems, such as other networks and protocols operating in the same ISM frequency band, or microwave ovens. Its value directly influences signal-to-noise ratio (SNR), thus also radio reception range and protocol performance. The background noise level in a dense urban environment can be relatively high and with high variance. By variance here we mean short-term variations, not stable differences in the noise floor of different nodes. This is somewhat in contrast with some works that assume a noise free environments [21], and with the conditions found in the testbeds commonly used in experiments.

To verify this statement we performed a set of measurements that, although relatively short-term, confirm that the differences are significant. Our tests in the Indriya [3] and TWIST [9] testbeds show mean values between -90 and -98 dBm and standard deviations between 2 and 4 dBm, depending on the node and the channel. On the other hand, we performed tests in an urban environment, where noise floor measurements were run on all IEEE 802.15.4 radio channels. The chosen environments were: *i)* the center of a large (10 M inhabitants) city at noon and at midnight; *ii)* the center of a small (100 K inhabitants) city; *iii)* a suburban road; *iv)* a suburban road near a university campus. The

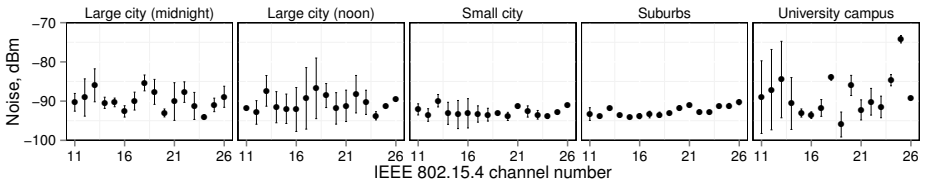

Fig. 2. Noise measurements in various environments: mean and standard deviation.

results, shown in Fig. 2, indicate that the noise levels and their variance in commonly used testbeds are very different from the ones found in an urban environment. Mean noise levels are usually between -85 and -95 dBm, but in some cases as high as -75 dBm, while the standard deviation of noise varies between 0 and 10 dBm. Finally, the noise floor varies according to the environment, e.g., between a densely populated urban area and a suburban street due to different levels of 802.11 interference.

MRM approximates background noise as a Gaussian distribution with a given mean and standard deviation, taking independent and identically distributed samples at each node at each time instance. We use this simplified abstraction, even if real distributions have temporal and spatial correlations and in some cases exhibit heavy tail properties. In what follows, we use the notation N_{avg} and N_{sd} to represent the noise floor and its standard deviation, respectively, and $MRM(N_{avg}, N_{sd})$ to indicate an MRM model with the respective parameters. We also use the notion of R_{th} for the *theoretical radio reception range*: an estimate of the maximum radio reception range calculated using Friis transmission equation based on the noise level (for simplicity, we use mean noise), parameters of the radio subsystem, and a transmission power of 0 dBm.

Of course, our radio model is not perfect. We chose to concentrate on noise floor as the factor governing radio reception range, and noise variance and interference as the studied sources of unreliability, while our model neglects effects such as hardware variations, spatial variations, or other effects usually expressed by higher than free-space path loss exponents in empirical models. For a detailed discussion of these, we refer the reader to [8,21]. Even if we do not directly account for these factors, we extend the studied noise floor range for two reasons: first, decreasing the transmission power with D dBi has the same effect as increasing the noise floor with the same amount, therefore, one can account for systems that operate at different transmission powers by looking at a shifted version of the figures; second, the effect of a higher path loss exponent is similar, but not identical, to a higher noise floor.

Protocol settings. All of the protocols described in Section 3, even the simplest ones, provide a high degree of customization through parameters such as buffer sizes, timeouts, retransmission and hop counters. Wherever possible, we choose to use the default values, as these are likely to be first choice in a real deployment and the ones for which the implementation has been tested the most and for which to expect more consistent and stable operation. We did, however, include tuned and fixed versions of two protocols (ContikiRPL and ORPL), since a careful review of their first results showed clear discrepancies w.r.t. expectations. For reproducibility of the experiments, the most important protocol parameters used in our study are summarized in Tables 2 and 3.

Application setup and performance metrics. We test the performance of the selected protocols when sending commands from the gateway to other nodes. This mirrors the

Table 2. Layer 3 parameters.

Protocol	L3 parameters	L2
Flooding	rebroadcast delay: 8–80ms	Contiki
Trickle	I_{min}=1/32s, I_{max}=1/2s, K=1	
ContikiRPL	routing table size: 70, routing metric: ETX	
TinyRPL		TinyOS
ORPL	routing metric: EDC	Contiki+RDC
TM	I_{min}=1/8s, I_{max}=256s, K=1	Contiki
SMRF	routing metric: ETX, rebroadcast delay: 30–125ms	

Table 3. Layer 2 parameters.

Parameter	Contiki	TinyOS
CCA backoff	128ms–1.2s	0.3–10ms
backoff increase	exponential	none
no-ACK retry delay	as backoff	103ms
no-ACK TX attempts	5	
neighbour table size	20	

operation of the current infrastructure, in which a command (e.g., issued by an operator) is forwarded to the appropriate gateway, which in turn disseminates it to the cluster. We use messages with 6 B of payload, enough to fit a command code and 1–2 parameters.

The evaluation of scalability in terms of traffic load is not our goal, given that actuation commands are issued relatively sparingly. Instead, the reliability and timeliness of commands is more important; therefore, we concentrate on the performance of delivering isolated actuation commands. In each experiment, the protocols are given a warm-up time to bootstrap and stabilize topology, after which a periodic traffic starts to be generated at the gateway node. We collect statistics only after the warm-up time. The gateway is sending $B = 2000$ commands, each command destined to a node chosen with uniform random selection, with an inter-command interval (ICI) set to 5 s, such that overlap between subsequent transmissions is minimized. Thus, each message travels on an unloaded network, except for topology maintenance traffic. For further 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 *delivery delay* of the selected protocols. We define the PDR of a given node as the ratio of packets (i.e., actuation commands) received to the number of packets sent to the node. The *delivery delay* is the time elapsed between the gateway sending the packet and the node receiving it. PDR and *delivery delay* were calculated for each destination node. Metrics were further averaged over all nodes of the cluster to derive statistics at the cluster level.

In addition to the metrics above, we consider also *network utilization, per actuation command*, expressed in bytes sent over the radio. This can be seen as an indirect measure of channel utilization and thus coexistence with other systems using the same 2.4 GHz ISM band, as well as scalability under traffic load. Furthermore, it can provide indications about energy consumption, which is however not relevant to our specific smart city scenario, since nodes are mains-powered. To compute this metric we considered all the data and control traffic after the warm-up time. This value is then normalized w.r.t. the number of actuation commands sent during the simulation.

6 The Game is On: Simulation Results

We compare the selected protocols w.r.t. the characteristics of the radio propagation environment and the network topology. The impact of noise floor is studied first. We initially neglect the effect of noise variance by setting $N_{sd} = 1$ dBm, and then analyze the impact of this parameter. Space limitations force us to show the noise analysis

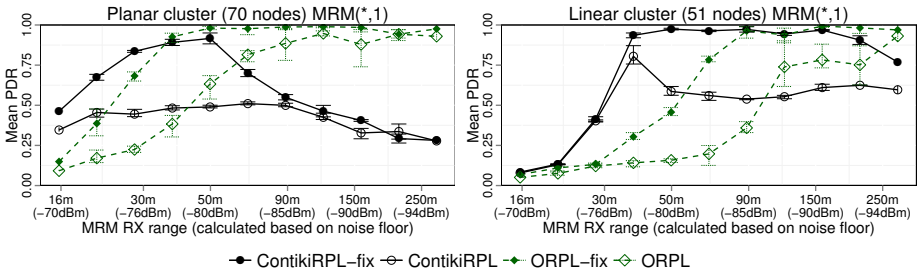

Fig. 3. PDR achieved by ContikiRPL, ORPL, and their debugged variants.

only on the two distinctive clusters shown on Fig. 1: a 70-node "planar" one and a 51-node "linear" one. We then focus dually on two interesting noise configurations, and analyze the impact of topology and scale, by showing results for all the clusters. As a last step, we compare the performance of dedicated multicast protocols against simpler dissemination- and unicast-based implementations of group commands. Before delving into our results, however, we need to describe the modifications we made to ContikiRPL and ORPL to render their performance acceptable in our scenario.

6.1 Debugging ContikiRPL and ORPL

During our first experiments, ContikiRPL showed lower performance than expected. Log inspection identified routing table management as the culprit. When a node rejoins the DODAG through a different branch, the next-hop entry at the branching point is not updated until it expires. Traffic along the original route causes routing errors, triggering unnecessary DODAG reconstructions through version increase. A vicious circle is formed: version increase causes churn, churn causes routing errors and version increase.

The default configuration of ORPL also performed poorly. A custom configuration using the less efficient hash-based filters instead of bitmap ones, and the minimum ContikiMAC period (125 ms) tested in [5], showed a degrading performance over simulation runtime, due to three problems. The so-called false positive mechanism stalled nodes, so we disabled it; it is anyway useless with bitmap filters. We also disabled the bitmap aging mechanism, which periodically cleared part of the bitmap. Finally, we had to modify the input filters in the reception path to solve occasional memory corruptions.

The positive effect of our "debugging" is shown in Fig. 3, where we compare our versions against the original ones. We show PDR as a function of noise floor on our reference clusters, although the improvements hold for all topologies and for the other metrics. Both our versions consistently match or outperform original ones, often with remarkable performance gains; therefore, hereafter they are the only ones we report. Code modifications will be submitted to the maintainers of ContikiRPL and ORPL.

6.2 Impact of Noise Floor

Fig. 4 shows the protocol performance as a function of noise floor, similarly to Fig. 3 but this time with all metrics and for all studied protocols, except multicast ones.

From a reliability (PDR) perspective, only Trickle performs well on both clusters and at all noise floor levels—at least where the graph is still connected. This is expected,

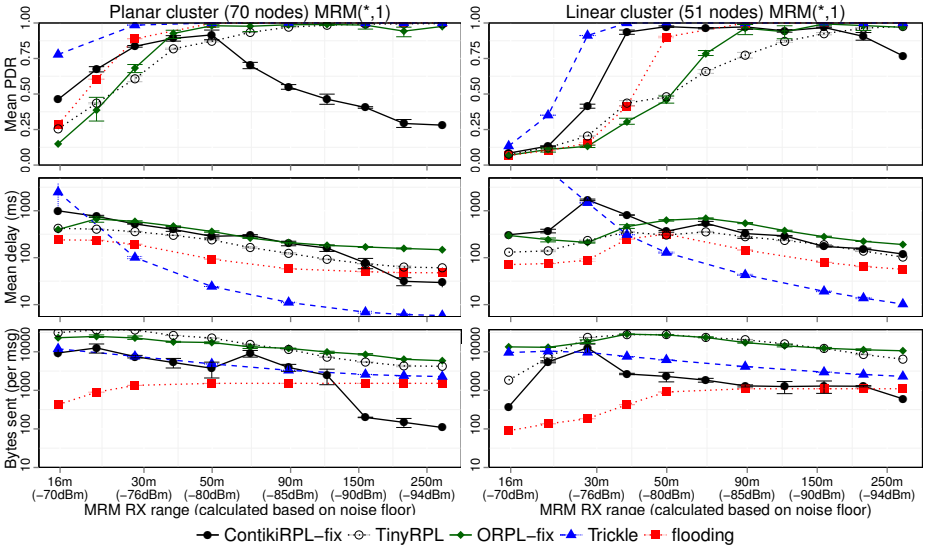

Fig. 4. Effect of noise floor on protocol performance: PDR, delay, and network utilization (rows) on the two selected clusters (columns). R_{th} is calculated based on the noise floor (x-axis).

since it is the only protocol enjoying unlimited retransmissions; in case of a single message, sooner or later Trickle delivers.

On the planar cluster, ContikiRPL-fix handles high noise better than flooding. In this situation, the reception range is so small that nodes have only few neighbors connected with weak quality links; the L2 retransmissions of ContikiRPL-fix are more effective than the multi-path properties of flooding. Nevertheless, below -75 dBm, as links become more reliable and link-level broadcast becomes more efficient, flooding takes the lead. On the linear cluster, ContikiRPL-fix provides good results below -77 dBm, while flooding requires -80 dBm. However, ContikiRPL-fix was not able to reach $PDR = 100\%$, even in medium-noise scenarios where its competitors achieve full reliability. ContikiRPL-fix shows very poor performance at low noise, especially on the planar cluster. In these conditions, the increased radio range increases the number of neighbors, overfilling the neighbor table capacity. Further, when a next-hop node is not found among the neighbors, a global DODAG repair is triggered. This negative impact of high network density on ContikiRPL implementation was also reported in [2]. Although increasing the size of the neighbor table could remedy this issue, our attempt to do so failed due to insufficient RAM in low-cost MCUs such as the MSP430F1611. ORPL-fix performs worse than ContikiRPL-fix at high noise, but on the other hand does not suffer from the problem above, 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, again consistently on both clusters. It is not affected by the same issue affecting ContikiRPL-fix in dense networks, since it does not rely on the neighbor table when resolving 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.

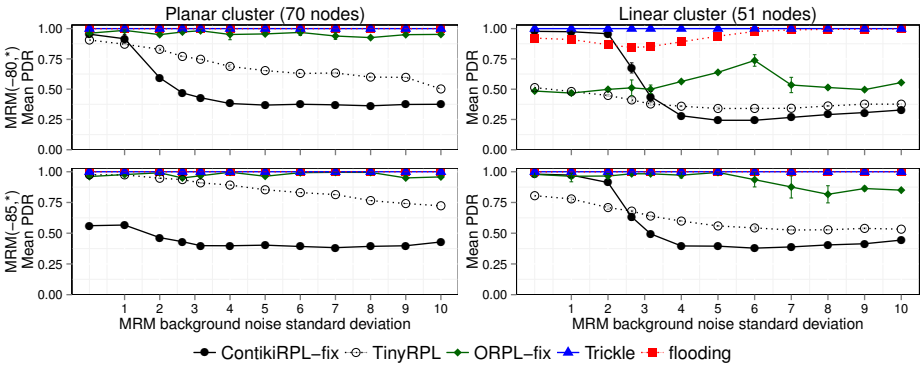

Fig. 5. Effect of background noise variance on PDR. Two clusters (columns) and signal propagation models (rows).

Regarding the *average delay* (timeliness) of actuation command delivery, differences were so significant that we had to resort to a logarithmic scale. Trickle achieves best performance for low noise situations. In high noise, it is seemingly overcome by other protocols, but in this range it is the only protocol with high PDR values, which makes its average delay non-comparable to other protocols. Flooding follows with a significant increase in delivery times, albeit still contained between 100 and 200 ms. TinyRPL achieved relatively small delays, although a comparison at high noise is unfair, since its PDR is lower than that of other protocols. 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, network utilization 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 that of RPL variants under most conditions, except for very low noise situations. This is likely due to two aspects of our setting: *i)* we use small messages, which however we deem realistic for actuation, where typically only a short command is to be delivered; *ii)* we send messages every $ICI = 5$ s, for which RPL cannot amortize topology maintenance costs among real messages. However, again, this interval is realistic, and actually our partner company confirmed that higher ICI values are common. Finally, the high network utilization of ORPL-fix is partly justified by the repeated send attempts of the duty-cycling MAC.

6.3 Impact of Noise Floor Variance

Another important, and often neglected, factor influencing protocol performance is the variance of background noise. Fig. 5 shows how the PDR of various protocols is influenced by N_{sd}, given a fixed mean value. The top row shows the effect of noise variance fixing the mean at -85 dBm. Figures for -90 dBm would be relevant, but less interesting, since PDR would be at 100% for most protocols. Instead, we show results for -80 dBm, where the effect of noise variance on flooding can also be observed.

Background noise variance hinders the performance of both ContikiRPL-fix and TinyRPL, but in a different way. TinyRPL shows a smooth, almost linear decline with

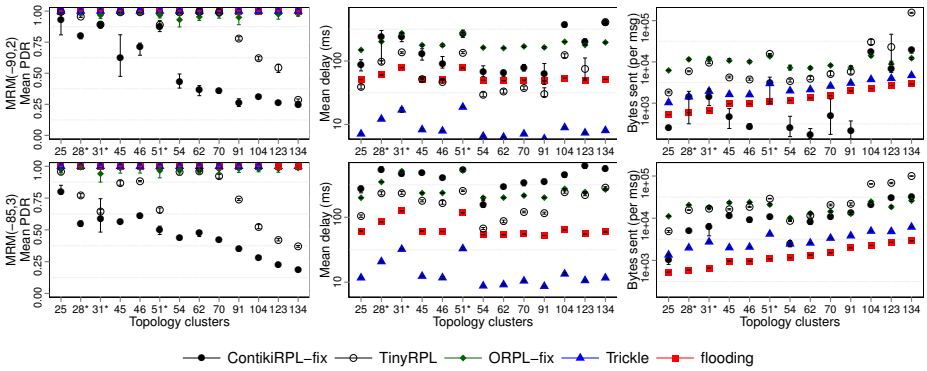

Fig. 6. Results for all the clusters. PDR, delay and network utilization (columns); low noise and high noise scenarios (rows).

increased N_{sd}, due to two effects: *i)* an increase of the so called "gray" area, meaning that there are more links with PDR far from both 0 and 1; *ii)* above a certain noise level, the link quality estimator itself becomes imprecise. In the case of ContikiRPL-fix, instead, the decrease is also due to the neighbor table issue already seen before: with higher variance, but same mean value, there are also lucky cases where a node receives messages from nodes farther away. These are then added to the neighbor table, which gets saturated, triggering frequent topology rebuilds.

Flooding and Trickle show near perfect PDR independent of the noise variance at -85 dBm. Flooding has only slight performance decrease when noise floor increases to -80 dBm, since it does not rely on link quality information. It can even benefit from an increased variance, as packets are diffused through multiple paths concurrently, benefiting from "lucky" situations where a packet is delivered across a lossy link. We have verified at higher mean noise levels that the effect is the same for Trickle.

Finally, the case of ORPL-fix is interesting as it is the consequence of the superposition of multiple effects. On one hand, opportunistic forwarding benefits from increased multi-path options, improving performance as N_{sd} increases. On the other hand, ORPL-fix relies on link quality estimation to select next hops and maintain the RPL topology. At higher values of variance, this reverses the trend, reducing performance.

6.4 All Clusters: Trends and Effects of Topology Geometry

After exploring the effect of both noise floor and noise variance, we concentrate on scalability and the effects of other topology characteristics. To do so, however, due to space limitations we must restrict simulations to two representative sets of parameters: $MRM(-90, 2)$ as a typical low noise scenario, and $MRM(-85, 3)$ as a higher noise scenario. Fig. 6 shows the performance statistics for all clusters. Recall that we use the number of nodes to identify clusters and that Fig. 1 shows topology geometry statistics.

Both ContikiRPL-fix and TinyRPL show scalability issues as the number of nodes grows, as already discussed. 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 mean noise levels below $-85dBm$.

Other generic trends that could be observed are the high PDR reached by flooding and Trickle independent of the number of nodes, and delay which is almost independent of the number of nodes for the same protocols. Network utilization instead increases almost linearly with number of nodes for all protocols.

There are, however, outliers from the above trends, and some of these could be correlated to topology characteristics. In fact, the three clusters marked by asterisks, have "special" topologies and are noticeable outliers in all performance figures. Clusters 31 and 51 are sparse and long (deployed along suburban roads), which is also reflected in the closest neighbor distance and squareness value in Fig. 1. Cluster 28 is U-shaped, having two long branches. This special form is not captured by our squareness metric, nonetheless, in some aspects it behaves 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 clusters (can only be seen in high noise), but it is not clear whether longer node distances, smaller neighborhoods, or longer multi-hop paths are to blame. ContikiRPL-fix instead shows a performance increase in low noise, and in this case we know it is due to the decreased neighborhoods.

6.5 Targeting Groups of Nodes

One of the requirements of the smart city scenario is the ability to deliver the same command to a group of nodes. TM and SMRF provide direct support for this functionality via multicast, while the other protocols must resort to potentially inefficient solutions. Dissemination protocols have no choice but reaching the entire network. Instead, RPL variants must resort to implementing a single group command as a "batch" of unicast messages, each addressed to a node of the target group. The tradeoff between the dedicated multicast support of TM and SMRF and the potential inefficiency of these solutions is precisely the subject of this section.

In principle, the group size bears an effect on the relative performance of protocols. However, in practice this parameter does not affect dissemination protocols, as they reach the entire network anyway. Further, we found out that group communication via unicast batches is affected more strongly by another parameter, i.e., the inter-message interval (IMI) between two messages of the same batch. The IMI essentially provides traffic shaping, and we show that, this is key to their performance. Therefore, we focus only on a group size of 10 nodes, as this is a typical value used in the system of our partner company, and show the impact of different IMI values. Fig. 7 shows the PDR of all the protocols considered, for the selected reference clusters and radio models. TM, SMRF, flooding, and Trickle are shown as horizontal lines, as their performance, unlike RPL variants relying on unicast batches, does not depend on the IMI.

TM inherits the excellent reliability of Trickle; both achieve $PDR = 100\%$, along with flooding. Instead, the other multicast protocol, SMRF, provides $PDR > 75\%$ only in the planar topology under low noise; its performance is unacceptable in the other cases. SMRF builds on top of ContikiRPL, inheriting some of its limitations; further, its reliance on 1-hop broadcasts without L2 acknowledgments or retries appears to bear a more negative effect than speculated by its authors in [16].

As for RPL variants using unicast batches, it is evident that some traffic shaping is necessary. Indeed, simply sending messages back-to-back ($IMI = 0$) has a

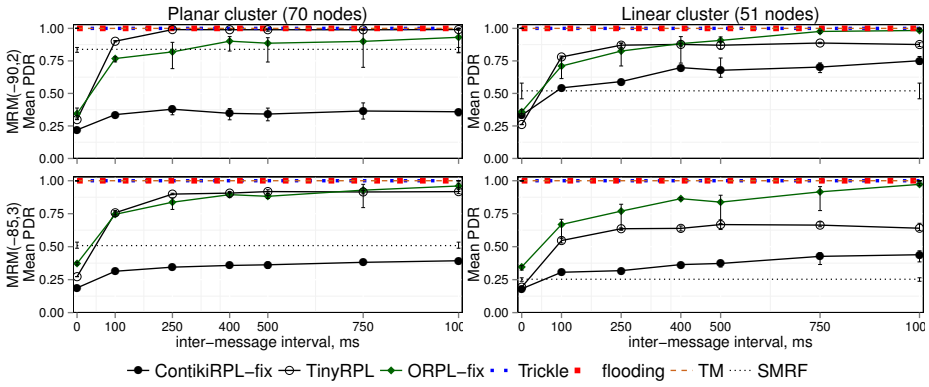

Fig. 7. PDR of multicast, dissemination and unicast protocols, sending commands to multiple destinations. For the unicast protocols, the impact of inter-message interval is also demonstrated. Columns: topology clusters; rows: radio models.

disastrous effect on performance, presumably due to internal buffers and interference among nodes. However, once more, different implementations behave differently. For TinyRPL, $IMI = 250$ ms is enough to reach its peak performance, which in the planar case with low noise reaches $PDR = 100\%$. For ContikiRPL-fix and ORPL-fix, instead, the larger the IMI, the better the performance. Shaping influences more the former protocol than the latter, and is more marked in the planar cluster than in the linear one. Unfortunately for these two protocols, IMI cannot be increased arbitrarily, as this increases the delivery delay and skews significantly the receipt time of group nodes; an appropriate value must be determined on a per-application basis.

7 The Verdict, and Some Lessons Learned

As stated in the introduction, we were convinced that flooding was going to be left in the dust by the RPL competition. However, our results tell a different story. RPL and its variants were outperformed by flooding, as well as by our other baseline protocol Trickle, under several conditions which can be summarized as follows:

- in situations with low background noise (or high range and therefore neighbor density), RPL implementations suffered from various reasons, including scalability issues and topology reconfigurations leading to increased packet losses;
- in situations with high noise variance, typical in dense urban scenarios, RPL suffered from unstable links and link estimation errors.

On the contrary, ContikiRPL-fix outperforms flooding in situations with high noise (low density) where long multi-hop paths are required to deliver messages, especially on the linear topology where multiple dissemination paths are not available. In this case link-level retransmission gave an edge to RPL. Nevertheless, even in this situation only ContikiRPL-fix was capable of outperforming flooding, while the performance of the other RPL variants remained below, with significant differences among the various implementations. Furthermore, in the aforementioned situation the simple dissemination scheme of Trickle outperforms ContikiRPL-fix, and appears to be the best choice.

Interestingly, Trickle is also the fastest protocol, while flooding is the one with the lowest network utilization, leaving few reasons to choose the RPL variants over our

dissemination baseline. To be fair to RPL, the extra overhead of maintaining a topology is expected to be amortized by the many-to-one data collection traffic we are not considering here. The topology built by RPL is actually optimized for this latter traffic, 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 relatively lower-traffic of actuation.

As for multicast support, dissemination protocols and TM provide the best reliability, with the latter inheriting the nice properties of Trickle. Unfortunately, SMRF similarly inherits the limitations of RPL and does not improve much over the alternative, provided by the other variants of RPL, of targeting a group of nodes via multiple unicasts. In this latter scenario, ORPL-fix often provides the best performance due to its optimized unicast. However, depending on the radio channel conditions, TinyRPL performs comparably or better, without dedicated mechanisms.

Protocol	**code**	**data**
Flooding	18	4.9
Trickle	19	4.5
TinyRPL	36	8.7
ContikiRPL	42	9.1
ORPL	45	8.5
TM	43	7.9
SMRF	40	6.6

Table 4. Memory requirements (OS included), in kB.

The shortcomings of RPL we hitherto illustrated are further exacerbated by implementation considerations. We already reported the difficulties we encountered in using ContikiRPL and ORPL out of the box. Furthermore, the superior performance of baseline dissemination protocols is also complemented by their simplicity, which translates in significantly less demands in terms of memory consumption, as shown in Table 4. As a term of comparison, the popular TMote Sky has only 48 kB of flash and 10 kB of RAM; RPL protocols would use most of available resources.

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

8 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 the state-of-the-art RPL protocol and its variants to the problem of large-scale low-power wireless actuation. We performed our study by simulation, for practical reasons concerned with the high number of protocols and settings under study. However, we borrowed directly from our smart city deployment the placement of nodes, allowing us to experiment with a real network scale and topology. Our findings suggest that RPL still leaves much to be desired when used for low-power wireless actuation. Some of the RPL implementations we tested required some debugging before they perform acceptably in our context. All of them performed worse, under several conditions, than the dissemination protocols we chose as baseline, which however are considerably simpler, and occupy far less memory.

There are obvious opportunities for future work on the topic of this paper. First, our findings are specific to our deployment, and should be validated in other smart city deployments, or other kinds of large-scale actuation scenarios. A real-world validation is also clearly desirable; we are currently negotiating the possibility to run experiments on the smart city infrastructure we analyzed here, to validate our results. 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.

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 *7th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2011.
2. S. Dawans, S. Duquennoy, and O. Bonaventure. On link estimation in dense RPL deployments. In *Int. Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, 2012.
3. M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed. In *TRIDENTCOM*, volume 90 of *LNCS*. Springer, 2011.
4. A. Dunkels. The contikimac radio duty cycling protocol. Technical Report T2011:13, Swedish Institute of Computer Science, 2011.
5. S. Duquennoy, O. Landsiedel, and T. Voigt. Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2013.
6. F. Ferrari, M. Zimmerling, L. Thiele, and L. Mottola. The low-power wireless bus. *Proc. of the 11th Int. Conference on Information Processing in Sensor Networks - IPSN '12*, 2012.
7. O. Gaddour and A. Koubâa. RPL in a nutshell: A survey. *Computer Networks*, 56(14), 2012.
8. A. Goldsmith. *Wireless communications*. Cambridge university press, 2005.
9. V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST. In *Proc. of the 2nd Int. workshop on Multi-hop ad hoc networks: from theory to reality - REALMAN '06*, New York, May 2006.
10. J. Hui and R. Kelsey. Multicast protocol for low power and lossy networks (MPL), IETF draft, 2014.
11. O. Iova, F. Theoleyre, and T. Noel. Stability and efficiency of RPL under realistic conditions in wireless sensor networks. In *24th Int. Symp. on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013.
12. J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. Evaluating the Performance of RPL and 6LoWPAN in TinyOS. In *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN)*, 2011.
13. J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. Contikirpl and tinyrpl: Happy together. In *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN)*, 2011.
14. O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: Opportunistic Routing meets Duty Cycling. In *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2012.
15. P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm. RFC 6206, 2011.
16. G. Oikonomou, I. Phillips, and T. Tryfonas. IPv6 Multicast Forwarding in RPL-Based Wireless Sensor Networks. *Wireless Personal Communications*, 2013.
17. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *31st Int. Conf. on Local Computer Networks*, 2006.
18. S. Petersen and S. Carlsen. WirelessHART versus ISA100.11a: the format war hits the factory floor. *Industrial Electronics Magazine, IEEE*, 5(december):23–34, 2011.
19. I. Radoi, A. Shenoy, and D. Arvind. Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks. In *Int. Conf. on Wireless and Mobile Communications*, 2012.
20. T. Winter et al. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550, 2012.
21. M. Z. Zamalloa and B. Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks (TOSN)*, 3(2):7, 2007.
22. Contiki operating system official website. `http://contiki-os.org`.
23. TinyOS official website. `http://tinyos.net`.