Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey

Hyung-Sin Kim, JeongGil Ko, David E. Culler, and Jeongyeup Paek*

***Abstract*—RPL is the IPv6 routing protocol for low-power and lossy networks (LLNs), standardized by IETF in 2012 as RFC6550. Specifically, RPL is designed to be a simple and interoperable networking protocol for resource-constrained devices in industrial, home, and urban environments, intended to support the vision of the Internet of Things (IoT) with thousands of devices interconnected through multihop mesh networks. More than four-years have passed since the standardization of RPL, and we believe that it is time to examine and understand its current state. In this article, we review the history of research efforts in RPL; what aspects have been (and have not been) investigated and evaluated, how they have been studied, what was (and was not) implemented, and what remains for future investigation. We reviewed 97+ RPL-related academic research papers published by major academic publishers and present a topic-oriented survey for these research efforts. Our survey shows that only 40.2% of the papers evaluate RPL through experiments using implementations on real embedded devices, ContikiOS and TinyOS are the two most popular implementations (92.3%), and TelosB was the most frequently used hardware platform (69%) on testbeds that have average and median size of 49.4 and 30.5 nodes, respectively. Furthermore, unfortunately, despite it being approximately four years since its initial standardization, we are yet to see wide adoption of RPL as part of real-world systems and applications. We present our observations on the reasons behind this and suggest directions on which RPL should evolve.**

***Index Terms*—RPL, IPv6, Routing Protocol, Internet of Things (IoT), Low-power and Lossy Networks (LLN)**

I. Introduction

RPL, the IPv6 routing protocol for low-power and lossy networks (LLNs), was designed to be suitable for resource-constrained devices in industrial, home, and urban environments [1]. The main goal of RPL is to provide IPv6 connectivity to a large number of battery-operated embedded wireless devices that use low-power radios to communicate and deliver their data over multiple hops. From the initial design phase, RPL builds upon widely-used routing protocols and research prototypes in the wireless sensor network (WSN) domain such as the collection tree protocol (CTP) [2] and Hydro [3], but is extended and re-designed to be part of, and ready for, IPv6. Specifically, RPL was designed to meet the requirements of several applications in the WSN and Internet of Things (IoT) domain [4]–[7], and is considered a critical component that links the low-power network connectivity to application layers in the IETF protocol suite for LLNs.

More than four-years have passed since the standardization of RPL as RFC6550, and we believe that it is time to look back to examine how researchers are utilizing RPL as part of their system implementations. With its importance and interest, over the past few years there has been considerable amount of effort to characterize, evaluate, and propose enhancements to RPL. These studies range from the domain of optimal parameter selection for target applications to interoperability and performance testing among different implementations. Using open implementations of RPL, some work focuses on evaluating the performance of RPL in testbeds and deployments, while many studies utilize simulated environments to explore and validate the flexibility provided in the RPL standard. We notice that the two most widely used open-sourced RPL implementations are ContikiRPL [8] and TinyRPL [9] within ContikiOS and TinyOS, respectively, and these implementations have been used in almost all RPL research activities that involve real experiments. Given that the RPL standardization process took multiple years, we notice that some work took place prior to the standardization, but most work with RPL occurred after its official standardization in 2012.

Given that the research community and industrial leaders have emphasized the attractiveness of IoT applications in various domains, we started this work with the hope to see RPL be applied in many real-world applications. To understand the current state of RPL, in this work, we review the history of research efforts in RPL; what aspects have been (and have not been) investigated and evaluated, how they have been studied, what was and what was not implemented in open implementations, and what remains for future investigation. Specifically, we have reviewed 97 academic research papers published by major academic publishers with the keyword "RPL" and present a topic-oriented categorization for these research efforts. Based on these observations, we discuss the challenges that RPL (yet) faces today four years after its standardization, and propose points of revision to RFC6550.

II. Background - RPL

We provide a brief background of RPL, the IPv6 routing protocol for LLNs, standardized by IETF in March 2012.

A. Vision and Efforts of IETF RoLL Working Group

IETF chartered the routing over low-power and lossy networks (RoLL) working group in 2008 to standardize a practical IPv6 routing protocol for LLNs (RPL). RoLL expected that

Hyung-Sin Kim and David E. Culler are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA. (email: hs.kim@berkeley.edu, culler@berkeley.edu)

JeongGil Ko is with the Department of Software and Computer Engineering, Ajou Univerisity, Republic of Korea. (email: jgko@ajou.ac.kr)

Jeongyeup Paek is with the School of Computer Science and Engineering, Chung-Ang University, Republic of Korea. (email: jpaek@cau.ac.kr). *Jeongyeup Paek is the corresponding author.

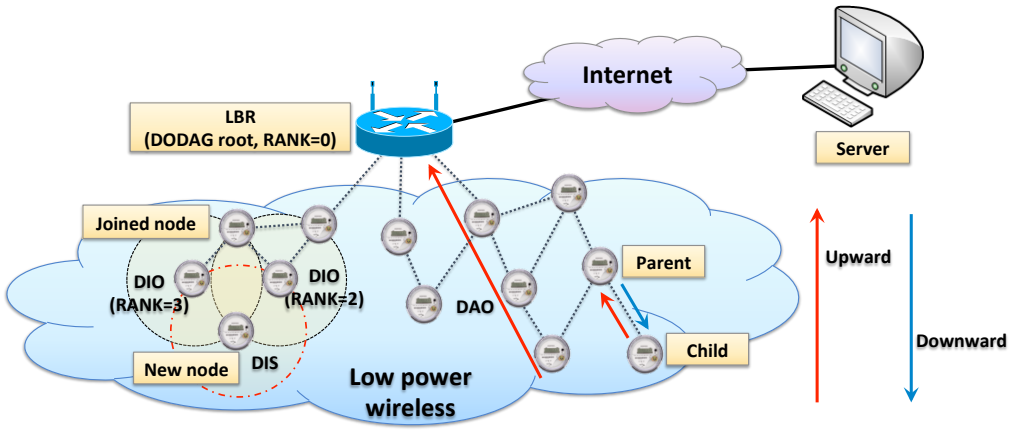


Fig. 1. Basic, simplest operation scenario of RPL with a single DODAG and a single RPL instance.

with the help of RPL standardization, various useful applications would be *realized* through LLN. The main characteristics of LLN are described in RFC6550 as follows [1]:

- LLN comprises *thousands of constrained nodes* that have limited processing power, memory, and sometimes energy (when they are battery operated).
- These constrained nodes are interconnected by *lossy links* that are usually unstable and typically support only low data rates.
- LLN supports *various traffic patterns*, not primarily point-to-point (P2P), but in many cases multipoint-to-point (MP2P) or point-to-multipoint (P2MP).

With this vision, RoLL first published several documents during 2009∼2010 that describe unique routing requirements in LLN by taking four representative types of applications as examples: urban applications in [4], industrial applications in [5], home automation in [6], and building automation in [7]. These requirements can be summarized as follows:

- **Traffic support**: A routing protocol for LLNs must be able to provide *bi-directional connectivity* between arbitrary two nodes in the network, and support *unicast, multicast, and anycast* service.
- **Resource constraint**: It should be implementable in resource constrained devices (e.g., *8-bit devices* with no more than *128kB (host) or 256kB (router) of memory* [7]). For battery-powered nodes, it should provide no more than 1% of duty-cycle [6] and/or *at least five years of lifetime* [5] [7].
- **Path diversity**: It must be able to provide *alternative routes* for reliable packet delivery (>99.9% packet delivery ratio with no more than three retransmissions [7]) over lossy links.
- **Convergence time**: It must converge after the addition of a new node within a few minutes [5], after re-establishment of a node or losing connectivity within *tens of seconds* [5] or *4 seconds* [6], and within *0.5 seconds* [6] if no nodes have moved.
- **Node property awareness**: It must take into account *node characteristics*, such as power budget, memory and sleep interval, for routing. It should route via mains-powered nodes if possible [6].
- **Heterogeneous routing**: It must be able to generate *different routes with different characteristics for different flows* to assure that mission-critical applications cannot be deferred while less critical applications access the network.
- **Security**: It must support message integrity to prevent attackers and/or unauthenticated nodes from manipulating routing functions or participating in the routing decision process.

After additional 3 years efforts, in 2012, RoLL finalized RPL standardization to fulfill the aforementioned requirements. RPL standard is described in RFC6550 [1], its routing metrics in RFC6551 [10], timer algorithm in [11], and its objective functions (OFs) for route calculation are described in [12] [13].

B. RPL's Key Features

We briefly describe key features of RPL, a distance vector type routing protocol that builds *directed acyclic graphs (DAGs)* based on selected routing metrics and constraints. This DAG structure is adopted to efficiently support upstream-dominant traffic patterns with resource constrained nodes.

The basis of RPL is to construct a quasi-forest routing topology, called *destination-oriented directed acyclic graph (DODAG)* rooted at one or more LLN border router (LBR), and support bi-directional IPv6 communication between network devices. Fig. 1 illustrates RPL's control messages and parent selection process with the simplest network structure, a single DODAG in the forest. Each node in RPL advertises routing metrics and constraints through *DODAG information object (DIO)* messages. Upon receiving DIO messages from its neighbors[1], a node chooses routing parents according to its objective function (OF) and routing information in DIO messages (e.g. RANK, DODAG ID), and then constructs a routing topology (i.e., DODAG). Note that a RPL node can

[1]Neighbor table management is out of scope of the RPL standard; RPL expects an external mechanism, such as Neighbor Unreachability Detection (NUD) [14], to verify link properties and reachability of neighbor nodes during the parent selection phase.

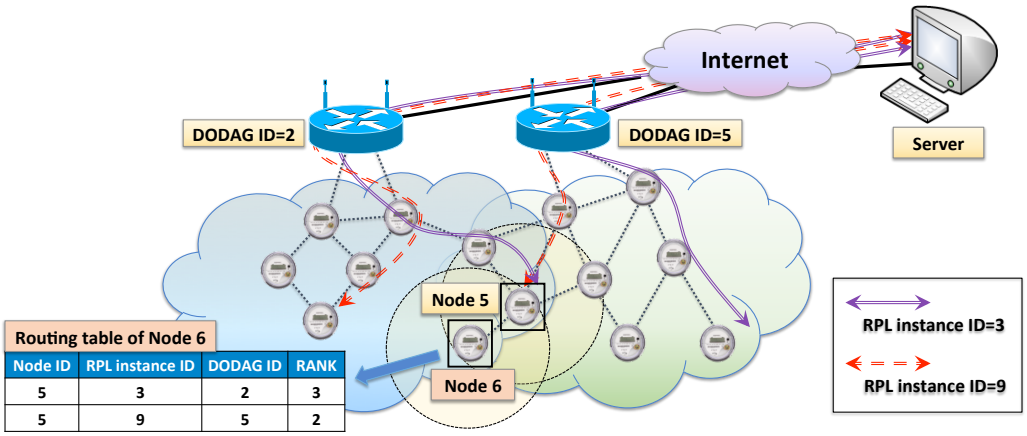


Fig. 2. General DAG structure of RPL with multiple DODAG roots and multiple RPL instances.

have *multiple parent nodes* to achieve reliable packet delivery through path diversity. DIO messages are transmitted based on the *TrickleTimer* [11] to achieve a balance between control overhead (energy consumption) and fast convergence/recovery, and minimize parameter configurations. DIO messages are also transmitted upon request when a DODAG information solicitation (DIS) message is received. RANK is defined and used by the OF to represent the routing distance from a node to an LBR, and link and node metrics (e.g., expected transmission count (ETX)) are used for RANK calculation and parent selection.

Once each node selects routes towards an LBR, RPL uses *destination advertisement object (DAO)* messages for reverse route construction, which advertise routing information on how other nodes can reach various destinations and prefixes within a RPL network when traveling down the RPL DODAG. How a DAO message is processed by each node and the LBR depends on which mode-of-operation (MOP) is used for downward routing: 'storing mode' (table-driven routing) or 'non-storing mode' (source routing). In 'storing mode', each node stores the downward routing information for all descendant nodes in its subtree, where as in 'non-storing mode', only the root node (LBR) stores that information for all nodes in its network. In both cases, basic idea is for the ancestor nodes to process and store the information in DAO messages to create routing entries for the nodes in the subtree.

A node that does not have a route (e.g. newly joined node) may use the *DODAG information solicitation (DIS)* message to solicit a DIO from a RPL node. Its use is analogous to that of a router solicitation (RS) as specified in IPv6 Neighbor Discovery; a node may use DIS to probe its neighborhood for nearby DODAGs.

Fig. 2 illustrates RPL's general DAG structure with multiple DODAGs and multiple RPL instances. Each RPL instance has its own OF for route construction. The use of *multiple instances* enables RPL to provide different routes with different QoS for different flows even for the same destination. Using *multiple DODAG roots* (LBRs) provides multiple exit points to the Internet for path diversity bandwidth and manageability. A RPL instance can use multiple DODAG roots (i.e., a single flow through multiple LBRs) and a DODAG root can be used for multiple RPL instances (i.e., multiple flows through an LBR). The combination of RPL instance ID and DODAG ID (unique ID for each LBR) uniquely identifies a single DODAG in the network. A node can join multiple RPL instances, and join a single DODAG for each RPL instance. When a node belongs to multiple DODAGs, it has a distinct routing identity in each DODAG; RPL exploits *multiple routing entries* to represent a *physically single* neighbor node. As a result, routing overhead (processing, memory, and control packet) depends on not only the number of neighbor nodes but also the number of RPL instances they participate in (i.e., neighbor nodes $\times$ RPL instances).

Lastly, it is also possible to form a single DODAG with multiple LBRs. For example, in the topology of Fig. 2, the server can be a backbone root (called a virtual root in RFC6550) of a DODAG that comprises the two LBRs and all other nodes; they have the same DODAG ID. In this case, RPL requires intimate coordination among virtual roots and LBRs that are connected through reliable communication links to share same DODAG parameters, but RFC 6550 leaves the detailed mechanism for future work.

Prior publications in their respective topics provide good description of RPL's individual features; For example, good overviews of RPL are provided in [15] and [16], and link estimation and table management of RPL are well explained in [17]. Furthermore, DualMOP-RPL [18] and MERPL [19] give good explanation of the storing-mode and non-storing-mode for downward routing in RPL, and QU-RPL [20] and OF-FL [21] provide detailed description of RPL's objective functions. Thus, instead of repeating the detailed overview description of RPL, we focus on the related work survey, their statistics and implications, missing parts, and our position in this work.

III. RPL Research Analysis - Statistics and Summary

In this section we provide summary statistics on the research papers that have investigated RPL. For this study, we searched research publications that can be found on Google Scholar, IEEE Xplorer and ACM Digital Library with keywords '*RPL*',

Topic	2010	2011	2012	2013	2014	2015	2016	Total	with experiments TinyOS	with experiments ContikiOS	simulation only	else*
Upward routing	1	3	1	6	5	5	2	**23**	3	7	13	+1
Downward routing	0	1	0	3	0	5	3	**12**	3	3	6	0
Load balancing	0	0	0	1	1	7	3	**12**	4	3	5	0
Interoperability	1	3	0	1	3	3	1	**12**	5	7	2	+1
Multicast	0	0	0	1	1	1	1	**4**	1	0	3	0
Multi-sink	0	0	2	2	1	1	0	**6**	0	2	4	0
Multi-instance	0	0	0	0	1	1	1	**3**	0	0	3	0
General DAG	0	0	0	0	0	0	0	**0**	0	0	0	0
Interference	0	0	0	1	0	0	2	**3**	1	2	0	0
Mobility	0	0	2	0	2	4	2	**10**	1	2	7	0
LOAD(ng)	0	1	0	2	0	1	0	**4**	0	0	4	0
Security	0	1	1	3	2	3	2	**12**	0	0	6	5 + 1
Survey	0	1	1	0	0	0	2	**4**	0	1	2	1
Total	2	10	7	20/18	16	31/28	19/16	105/97	17/14	28/26	55/52	6 + 3

TABLE I
NUMBER OF RPL-RELATED RESEARCH PAPERS IN EACH SUB-TOPIC, TABULARIZED BY THEIR PUBLICATION YEARS AND EVALUATION METHOD. (TOTAL OF 105 ENTRIES FOR 97 UNIQUE PAPERS, INCLUDING 8 DUPLICATE ENTRIES FOR PAPERS WITH TWO SUB-TOPICS. THE '/' SYMBOL SEPARATES TOTAL COUNT INCLUDING DUPLICATES WITH UNIQUE COUNT.)

'*routing*' and '*lossy*'[2] from 2010 to October 2016. Google Scholar, IEEE Xplorer and ACM Digital Library returned 2200, 654, and 87 publications respectively[3]. Along with the fact that RFC6550 has been cited 961 times during four years, a large number of related publications roughly show that researchers have paid attention to RPL. Among these, we reviewed 97+ research publications that were selected by considering relevance and the number of citations, and by following the papers that cite or are cited by the papers that we have already looked at. We have neglected papers that provide only basic description of RPL or that we believed as only marginally related.

A. Sub-topic Categorization

To begin our statistical study, we have clustered and categorized the papers into 13 sub-topics as shown in Table I. In general, our table has total of 105 entries including 8 duplicate entries that discuss two or more subtopics to sum up above 97 unique papers.

'Upward routing' studies the routing and packet delivery performance of RPL when traffic is flowing upwards from individual LLN devices to an LBR, a typical scenario in data collection WSNs. This category also includes studies on outside-RPL issues such as link estimation and neighbor table management, and internal characteristics of RPL such as parameter selection, interaction with the MAC layer (e.g. duty-cycling), and implementation details. **'Downward routing'** category focuses on the routing and packet delivery performance of RPL when traffic is flowing downwards from an LBR to individual LLN devices. It includes both storing-mode and non-storing-mode downward routing of RPL, and discusses application scenarios where downward control/actuation traffic is important or where bi-directional connectivity is required (e.g. TCP). There were three papers that discussed both 'upward routing' and 'downward routing' together, and some papers focus on 'any-to-any routing' which includes upward and downward routing. In these cases, we have put them in both sub-topic categories.

In addition, **'Load balancing'** category discusses the unevenly distributed parent selection and load imbalance problem of RPL, and **'Interoperability'** investigates the interoperability of RPL for multiple implementations (e.g. TinyOS vs. ContikiOS), multiple mode-of-operations (MOPs) for downward routing (i.e. storing vs. non-storing), distinct link and physical layer (e.g. Bluetooth Low Energy, IEEE802.15.4, and PLC), and how TCP can be supported over RPL. **'Multicast'** studies how RPL can provide multicast service.

'Multi-sink' studies how RPL can make use of multiple DODAG roots (LBRs): both RPL's internal (i.e., cross-DODAG load imbalance) and external (multi-channel scanning and three-tier network architecture) issues. **'Multi-instance'** is a more RPL-standard-specific topic where multiple instances of DODAG (vaguely supported by RPL) can be used to support different traffic types with different QoS requirements on a single physical network. We also include a **'General DAG'** category to count papers that study how RPL can efficiently operate with its general DAG structure (e.g., Fig. 2), which is basically intersection of both 'multi-sink' and 'multi-instance' categories. However, to the best of our knowledge, there is 'zero' research papers in this category despite considerable attention being devoted to it in the RPL RFC. We note that all papers, except the 9 explicitly in these three categories, consider the simplest structure with a single DODAG and a single RPL instance with a single LBR, as shown in Fig. 1.

There are more traditional topics from wireless networking such as, **'Interference'** which studies the impact of external (e.g. WiFi and microwave oven) interference and ways to cope with them, and **'Mobility'** which investigates how mobile endpoints can be supported within a RPL network and the performance implications of doing so. Note that mobility was *NOT* a design consideration for router nodes in the RPL standard. **'LOAD(ng)'** is a very specific topic where RPL is

[2]The two other keywords, 'routing' and 'lossy', which are included in the definition of RPL, help to get relevant results. Note that the acronym 'RPL' is also used in other fields with different meaning such as 'Rat Placental Lactogen', 'Recognition of Prior Learning', and 'Recurrent Pregnancy Loss'.

[3]The results of Google Scholar include almost all results (if not all) of IEEE Xplorer and ACM Digital Library.

compared with the LOAD [22] or LOADng [23] protocols, AODV-like routing protocol for LLNs. Note that AODV was originally designed for MANETs. Although performance evaluation of a new protocol (RPL) through comparison with a prior work is valid and essential, it is interesting because their respective design originally targets different network scenarios. It is more common to compare RPL with other WSN routing protocols (e.g. collection tree protocol (CTP)). **'Security'** discusses the security aspect of RPL, but not in terms of the (optional) security mechanisms and features in the RPL standard, but in other aspects of RPL related to routing and topology.

Finally, there were a few **'Survey'** papers. One of them elaborates the challenges and problems of RPL in its proposal/draft form at the point of time just before the standardization (2011) [16]. We see that many of the concerns raised by this paper are still valid today after the standardization, and review what have been addressed through later research and what other challenges remain. Another paper surveys how mobility has been taken into consideration for RPL when the original standard does not support it [24], and a more general survey of RPL was provided in [25]. A recent article by Zhao et.al [26] also surveys RPL with a focus on support for P2P routing.

We will use these 12 sub-topics (except 'survey' category) for the following discussions, and will describe and discuss the papers in each respective sub-topic in more details later in Section IV.

B. Statistics, Analysis, and Discussion

In this subsection, we provide and analyze the summary statistics on the 97research papers that have investigated RPL based on their sub-topic categorization, publication year, evaluation method, publication venue and demographics.

By Sub-topic:

Table I summarizes our categorization, and shows some interesting observations. First of all, as expected, upward routing performance has been studied the most (23 papers). Following next are the downward routing, load-balancing, interoperability, and security, all with 12 papers. Downward routing and load-balancing topics are long-loved and popular research topics in embedded multihop (quasi-forest) routing protocols due to fundamental trade-off between routing efficiency and resource constraints, such as limited memory and control overhead. In essence, RPL design was optimized for upward routing with an assumption that data collection will be the majority of the traffic in the network. This philosophy allowed the control overhead and memory usage (for routing table) to be low, but sacrificed downward routing efficiency and left the load-balancing issue open, resulting in several research efforts to improve them. Interoperability is essential to apply RPL in real-world use cases, given that network performance depends not only on RPL but also inter-operation of all OSI 7 layers (inter-layer operability) and an IoT network may comprise a diverse set of hardware platforms and software implementations from different vendors (inter-vendor/platform operability).

Security was somewhat surprising because RPL standard provisions for proven Internet standard security mechanisms for external attacks, but researchers mainly studied the internal attacks related to routing disruption. However, as we discuss later, security was studied conceptually using abstract simulations only; there were no real prototype implementation of RPL's security mechanisms defined in the standard.

Next comes mobility with 10 papers. Mobility being a popular topic is somewhat expected; because mobility was not a consideration in the RPL standard and hence unsupported by design[4] RPL has several weaknesses and vulnerabilities under mobility, leaving opportunities wide open for researchers.

Other sub-topics had a similar number of papers. The topic that we feel as somewhat under-studied is the IPv6 multicast over/under RPL despite its importance. Furthermore, we feel that more research effort is needed in the multi-sink subtopic since it will be required to cover large deployment areas via multiple entry points (LBRs) over the DAG structure when thousands of devices are envisioned (e.g. Smart Grid AMI [27]).

What is completely missing here is research efforts to support RPL's general DAG structure. Given that 'multi-instance' is the first-mentioned key feature of RPL in the design principle (section 1.1) of RFC6550 and using 'multi-root' is necessary to apply RPL network in practice (for large-scale deployments), the general DAG structure is a major characteristic of the RPL standard. Moreover, since RPL should be implementable on resource constrained devices and routing complexity of RPL increases with the number of DODAGs and RPL instances, investigating complexity and overhead issues of the general DAG structure, with scalability consideration, is essential. However, these issues have been disregarded by the research community.

Most importantly, we cannot find any published real-world large-scale deployment of applications and systems that successfully (or unsuccessfully) use RPL, which is the very reason why RoLL was organized to develop LLN and RPL. We found one real-world deployment for an e-price tag application in a market [28], but in a limited context and scale. All other experiments were done only on testbeds in research settings. Cisco provides industrial evidence of RPL's usage [29] [27] without publicly published information on specifics or performance. We were very much surprised by the lack of publications on real applications and deployments more than four years after the standardization.

By Publication Year:

As it can be seen from Table I, 19 papers have been published during 2010 $\sim$ 2012 before the official standardization. This is when RPL was in the form of proposed standard draft. From 2013, after the point where RPL was standardized in RFC6550, we see a steep increase in the number of publications; 18 publications in 2013, 16 in 2014, and a burst of 28 papers in 2015. The count is still increasing in 2016 with

[4]To be more precise, RPL standard states that mobile nodes should not forward information. As so RPL does not consider router nodes to be mobile, but leaf nodes can. Several work on mobility in RPL aim at allowing routing nodes to be mobile.

16 published papers so far as of October 2016. We believe this trend will continue; since IoT is one of the current megatrends in technology evolution, and RPL is believed to be one of the key enablers for supporting IPv6 over LLN. There will be more effort in trying to adopt RPL for embedded devices forming the IoT. Furthermore, because we still see several technical challenges remaining in RPL, both fundamental challenges and standardization and/or implementation issues, we expect more research publications in the coming years.

By Evaluation Method:

Out of the 97 research publications reviewed, only 39 (40.2%) papers evaluate RPL (and their proposal, if applicable) through experiments using real embedded devices. More than half (52 papers, 53.6%) of the publications used simulations only for their study, and 6 papers (6.2%) used neither experiments nor simulations to evaluate the work. The right half of Table I presents the number of papers based on their evaluation method for each subtopic. Note that the table includes duplicate entries for cases where a paper discussed more than one subtopic, or used both TinyOS and ContikiOS for evaluation. The 'else*' category indicates papers with neither experiment nor simulation, except for two papers which used RIOT-based experiment and one paper with FreeRTOS-based experiment. This is a bit disappointing. First, the tremendous effort devoted in making RPL an Internet standard was in a hope to see wide adoption of compatible IPv6 protocol suite in real applications and deployments. The concepts and ideas of RPL are from the WSN domain and have been there for a long time. If evaluated only in simulation, fundamental differences are small. Second, more importantly, we know that there is a large gap between simulations and real experiments; what has been evaluated in simulations may not reflect reality. We hoped to see more 'real' evaluation of RPL, its proposed extensions and improvements.

For experiments, TinyRPL in TinyOS and ContikiRPL in ContikiOS were the two most widely used software implementations. These implementations were popular not only because they are open-source, but due to the popularity of their respective operating systems in the WSN/LLN community. Specifically, TinyRPL was used in 14 unique papers and ContikiRPL in 26 papers, with 4 papers using both, adding up to 35 unique papers using these two implementations (92.3%) out of 39 papers that conducted experiments. The only three other implementations were NanoQplus in [18], RIOT [30] [31] in [32] [33], and FreeRTOS in [34].

An interesting point to be noted here is that, no experiment-based evaluation was performed for multi-instance, LOAD(ng), and security subtopics. Also, only a small fraction of multi-sink and mobility-related work have done experiments. Again, this is disappointing, and demands more real-experiment based research in those categories.

As shown in Fig. 3, the major hardware platform used for RPL experiments was the 'TelosB' ('Tmote sky' is equivalent) platform with MSP430 microprocessor and CC2420 radio developed in 2004. It was used in 27 unique papers out of 39 papers that conducted experiments, a 69%. Other platforms used were 'WSN430 open node', 'M3 open node', 'JN5168', 'MSB-A2', 'Zolertia Z1', 'PowerNet', 'WPCDevKit', 'PLC G3', and 'BCM4356', each appearing only once or twice in our list of papers. In the perspective of physical and RF layers, 'WPCDevKit' and 'PLC G3' have a PLC transceiver, only 'BCM4356' has a BLE transceiver, and all other platforms have an IEEE 802.15.4 transceiver. An interesting point is that, despite continuous advancement of IoT platforms, 'TelosB' [35], one of the classic WSN platforms, is still frequently used for research of RPL which was standardized in *2012* (after 8 years). Given that MCU of 'TelosB' has memory of 48kB ROM and 10kB RAM, which is much smaller than recent platforms such as Firestorm [36], RPL implementations considering the use of 'TelosB' may not (and does not) provide the full functionality of RPL.

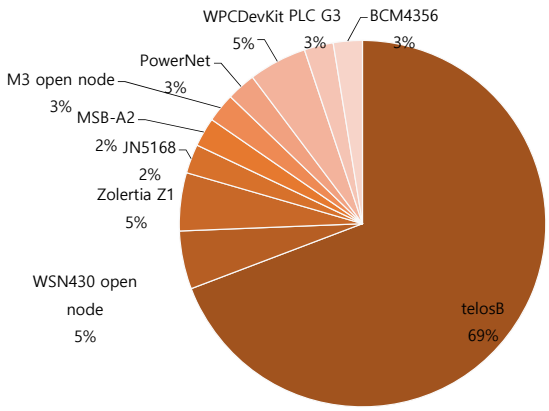


Fig. 3. Distribution of hardware platform.

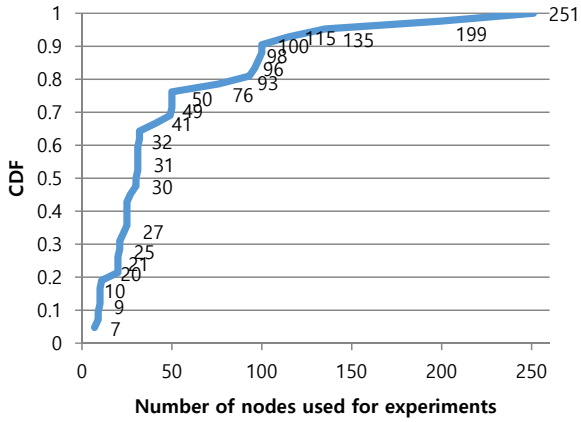


Fig. 4. CDF of number of nodes used for experiments.

Fig. 4 plots the cumulative distribution function of the number of nodes used for experiments in the 39 papers that performed experiment-based evaluation. The average is 49.4 nodes, and the median is 30.5 nodes. Research groups with a large testbed were able to conduct experiments on 90∼251-node networks, but there were also papers with experiments done on only 7∼10 node networks. We believe that at least minimum of 25∼30 nodes are required to see multihop characteristics of RPL, while larger is better to see real-world applicability of the protocol.

For simulations studies, COOJA simulator [37] using ContikiOS/ContikiRPL implementation was the dominant method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2017.2751617, IEEE Communications Surveys & Tutorials

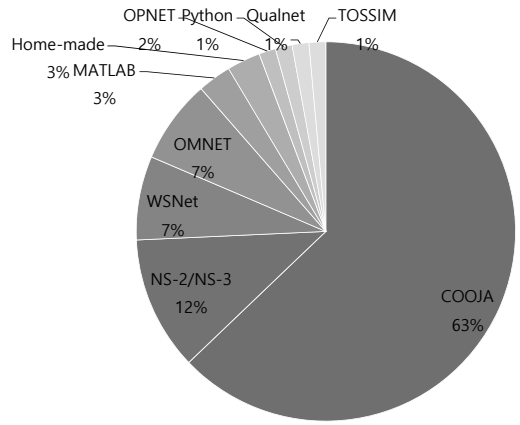
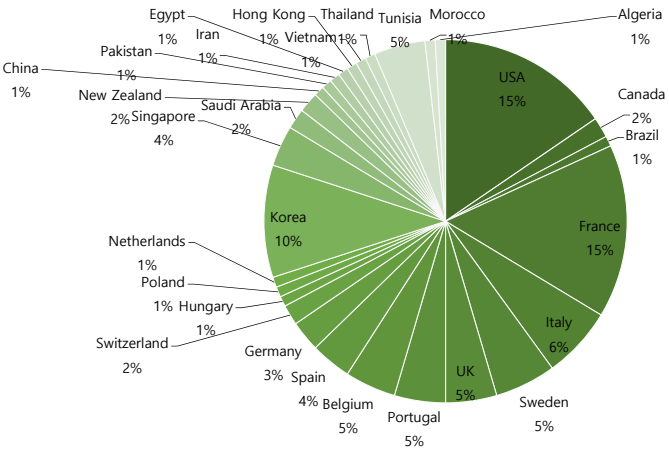


Fig. 5. Distribution of simulation method.



(a) All publications

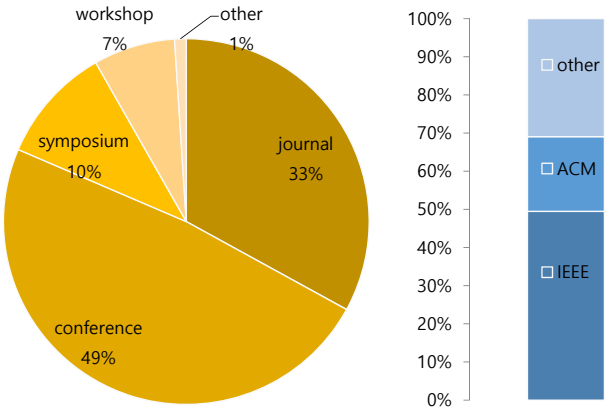
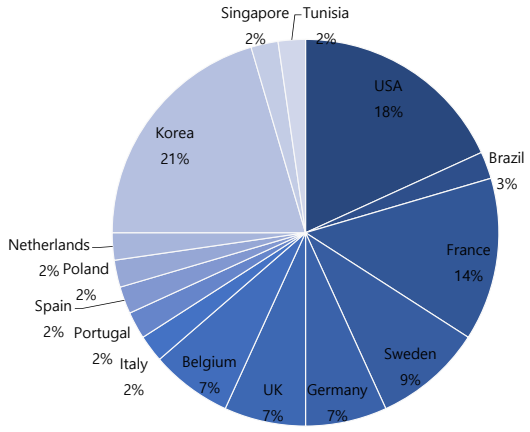


Fig. 6. Distribution of publication venue.



(b) Publications with real experiments

Fig. 7. Distribution of publication demographics (based on first author).

– 62.9% of all the simulation studies used COOJA simulator, as shown in Fig. 5. Runners-up were the NS-2/NS-3 [38], WSNet and OMNET++ simulators with 11.4%, 7.1% and 7.1% respectively. Other simulators used were Matlab, OPNET, Qualnet, Python, TOSSIM, etc. It is interesting that the TOSSIM simulator for TinyOS was used only once in the 97 publications. It turns out that, TOSSIM simulator only supports Mica2/MicaZ platforms for simulations, but these platforms did not have enough RAM to run the BLIP/TinyRPL stack for IPv6 and RPL in TinyOS. The only one paper that used TOSSIM investigated the DODAG root failure detection problem, but implemented the proposed mechanism not on RPL but CTP, due to lack of memory [39].

By Publication Venue and Demographics:

Fig. 6 plots the distribution of venue types for RPL-related publications. 34.4% of the papers were published in international journals, and 64.6% were published in conference/symposium/workshop proceedings where IEEE SmartGridComm and ACM SenSys were the most popular venues with 5 and 4 publications respectively. 49.5% of all the papers were published at venues sponsored by IEEE, and 20.4% at ACM sponsored venues. Other publishers include Elsevier, Springer, Inderscience, Hindawi, etc..

In terms of demographics, USA, France, and Republic of Korea ranked 1, 2, 3 in terms of number of publications and constituted 40% of all the publications, followed by Sweden, Italy, and UK/Portugal/Tunisia as shown in Fig. 7(a). However, as shown in Fig. 7(b), if we restrict the count to only the papers with real experiments, Republic of Korea, USA, and France ranked 1, 2, 3 and constituted 54% of all the publications, and Europe all together had similar number of publications compared to America and Asia combined. One interesting point to note is that, most of the publications from Europe used ContikiRPL implementation for experiments while those from America and Asia mainly used TinyRPL implementation.

IV. Survey of RPL Related Work

In this section we discuss the RPL related work in each of the sub-topics as categorized in subsection III-A. For those readers who would like to focus on our position over the list of related work may skip this section and jump to Section V.

Within the following subsections, we provide "Standard-related discussion" paragraphs to discuss how the research efforts in that particular sub-topic relates to the RPL *standard* (beyond a stand-alone research work), and also "Implication" paragraphs to summarize the possible implications of that discussion to the future of RPL.

A. Upward routing

Given that most of LLN applications focus on monitoring of sensor information, RPL was designed mainly to support upward traffic (from individual LLN devices to an LBR) delivery in LLNs. Thus, a number of studies evaluated the uplink performance of RPL. This category also includes studies on link estimation, neighbor table management and internal characteristics of RPL such as parameter selection and implementation details.

Tsiftes *et al*. first evaluated the performance of ContikiRPL in [8] and Ko *et al*. first evaluated that of TinyRPL in [9], [40], which show that both the two representative RPL implementations provide reliable upward packet delivery. Especially in [9], [40], the authors showed that TinyRPL provides upward packet delivery performance that is comparable to CTP [2]. Kim *et al*. deployed a TinyRPL-based multihop network in an urban marketplace, which confirmed the reliability of TinyRPL's upward packet delivery [28].

However, these results were contradicted by some other work, especially for ContikiRPL, which showed that RPL is problematic even in upward packet delivery (i.e., RPL's main goal) under certain scenarios. Ancillotti *et al*. evaluated ContikiRPL's uplink performance using COOJA simulator, which showed that ContikiRPL makes some nodes maintain unreliable routes even though reliable alternative routes exist, resulting in severe performance degradation for those nodes [15]. The authors pointed out that this is because '*RPL lacks of a complete knowledge of link quality of each parent candidate in dynamic wireless link environments*'; ContikiRPL implementation tracks link quality of a (not carefully selected) small subset of neighbors even though it has more neighbors in the neighbor table. The quoted statement needs to be revised since it is natural that RPL, as an L3 protocol, expects an external L2 mechanism to provide a set of reachable neighbors and their respective link qualities; Neighbor table management and link quality estimation (role of L2) are orthogonal to RPL standard. Precisely, the performance degradation implies lack of a well-designed external neighbor management mechanism in ContikiRPL implementation.

To improve the performance, the authors do not design an independent lower layer protocol for RPL, but two cross layer link estimation mechanisms which use *RPL* control messages for *link* estimation. First, the authors propose Trickle-L^2, a lightweight link estimation procedure that exploits Trickle-based topology maintenance, which rapidly updates link quality of all neighbor nodes by using the sequence number of DIO messages and *TrickleTimer* reset [41]. Second, they designed RPLca+ that includes a fast link quality update of each neighbor based on DIS unicasting and priority-based neighbor table management [17], and evaluated its performance through both COOJA simulations and testbed experiments. A common feature of Trickle-L^2 and RPLca+ is that RPL does not merely depend on data traffic-based link quality updates[5] but actively participate in tracking link quality or other information of neighbor nodes.

Dawans *et al*. evaluate ContikiRPL's performance in a large-scale testbed and observe the same problem: ContikiRPL experiences performance degradation since it cannot obtain up-to-date link qualities of alternative parent nodes [42]. However, the authors take a different approach than [41] and [17]; They make RPL's link quality update reactive to data traffic delivery without using RPL control messages. They show that, by simply setting the initial ETX value for each parent candidate to the *best* value (=1)[6], RPL can trigger more parent changes and measure link quality of more parent candidates. Although this approach improves the performance of ContikiRPL, the authors miss that their strategy has been already used in TinyRPL, which constrains their contribution to only ContikiRPL implementation. In terms of link quality probing, even before the RPL standardization, Hui and Culler propose to forward data traffic through alternative routes temporarily to probe link qualities of a diverse set of nodes [43], which is not implemented in TinyRPL nor ContikiRPL.

Another work [33] pointed out that checking bi-directional connectivity with neighbor nodes through RPL control message exchanges (DIS and DIO) has a scalability issue. To reduce control message transmissions, the authors suggest using multicast DIO rather than unicast DIS/DIO handshaking, and adopted a Bloom filter to include information of many neighbor nodes in a multicast DIO. They evaluated the proposal through testbed experiments.

Khelifi *et al*. find that even when ContikiRPL succeeds in detecting unreliable links, it requires a long detection time (after experiencing many packet losses) due to RPL's reactive nature. The authors designed Pro-RPL based on ContikiRPL which includes not only route cost but also traffic load and energy consumption information in DIO. Each node monitors each neighbor's status by calculating its 'suffering index' upon each DIO reception, and excludes a neighbor node when its suffering index becomes larger than a threshold [44]. However, most of the performance improvement does not come from additional information in DIO nor a new 'suffering index', but from a simple threshold-based neighbor filtering. Moreover, they evaluated their proposal only through simulations.

- **Standard-related discussion 1**: Although the six aforementioned papers tackle RPL's reliability issue, an interesting observation is that they do not tackle the RPL *standard* itself, but lack of an *external* neighbor table management mechanism that RPL relies on. All the six papers focus on ContikiRPL implementation that does not have an external neighbor management mechanism. Their solutions actively use RPL's features (control messages and parameters) for neighbor management and are included as part of '*RPL implementation*'. In contrast to ContikiRPL, TinyRPL includes a simple neighbor reachability detection mechanism based on an ETX threshold and initializes ETX value of

[5]Even though the RPL standard decouples neighbor reachability detection from its scope, RFC6550 notes that such a detection mechanism should preferably be reactive to *data traffic* in order to minimize the overhead.

[6]At that time, ContikiRPL implementation sets the initial ETX value for each parent candidate to five. In ContikiOS 3.0, the latest version released in August 2015, the initial ETX value is changed to two, which triggers more parent changes than the previous setting.

each neighbor to the best value, resulting in no reports of uplink reliability problems in the literature. This implies that even though RPL decouples neighbor management from the *standard*, a RPL *implementation* needs to have a well-designed neighbor management mechanism to provide reasonable performance. Indeed, RFC6550 does write; "*a node SHOULD verify that bidirectional connectivity and adequate link quality is available with a candidate neighbor before it considers that candidate as a DODAG parent*" and "*nodes SHOULD provide a means to filter out a parent whose availability is detected as fluctuating*". We see this as one of the most important issues to be investigated more systematically for applying RPL in real-world use cases; This research is not for improving RPL standard but RPL implementation to apply it in practice.

- **Implication 1**: Neighbor table management is not merely an external mechanism to RPL standard, but an essential component for RPL. This is because RPL is designed for '*LLN*', where neighbor table management is much more challenging than conventional networks due to dynamic/lossy link characteristics and tight resource constraints.

Another body of work evaluates RPL's performance through testbed experiments and pointed out that RPL's uplink performance degradation comes from determining routes *before* sending data packets (i.e., sender-side forwarder selection). They addressed the problem by combining opportunistic routing scheme (i.e., receiver-side forwarder selection) with RPL for performance improvement. Duequnnoy *et al*. . designed ORPL [45], where a sender simply broadcasts a packet and each packet receiver makes the forwarding decision by comparing the cost in the received packet and that of the receiver. The authors used a new routing metric called expected duty-cycles (EDC) [46] for opportunistic routing, and improved both reliability and latency performance in multihop duty-cycled networks relative to ContikiRPL with its shortcomings described above. Their ORPL implementation in ContikiOS also provides a simple neighbor table management mechanism. Gormus *et al*. designed another ORPL [47], where each packet includes a forwarder set (rather than only one forwarder) and any receiver in the set forwards the packet (i.e., combination of sender- and receiver-side forwarder selection).

Ho *et al*. tried to overcome link dynamics by changing preferred parent node rapidly, even during each packet transmission [48]. The authors designed PPS in which a node changes the parent node after experiencing a predetermined number of packet retransmissions, and re-attempts to transmit the packet to a new parent node rather than dropping the packet. However, the authors do not study how this aggressive parent change during each packet transmission can be harmonized with downward routing, and more importantly, provide only simulation results. Furthermore, this idea was already investigated in [43], prior to the RPL standardization.

K. Iwanicki points out that, compared to link failure detection, node failure detection has received considerably less research attention and RPL detects root failures slowly with its *Trickle* and DODAG versioning techniques [39]. The author also found that while a link failure can be detected by a single node, a node failure can be detected by multiple nodes, since it removes all its links. Based on this idea, the author designed RNFD that detects root failures quickly through multi-node cooperation and evaluated its performance through TOSSIM and two testbeds. However, he postponed the integration of RNFD with RPL due to lack of memory in TelosB, and implemented the proposal on top of CTP.

Lastly, some work investigates the effect of parameter settings in RPL networks. Isern *et al*. evaluate ContikiRPL in a large testbed network and revealed that the performance can be improved by optimizing parameters, such as minimum DIO interval (I_{min}), routing metric threshold for the parent change and packet queue size [49], which is the largest testbed experiment in our survey (251 nodes). Kermajani and Gomez show that two parameters (i.e., redundancy count k and minimum DIO transmission interval I_{min}) of *TrickleTimer* have critical impact on the network convergence time of RPL [50]. They also proposed to use *TrickleTimer* for DIS transmissions and showed that their proposal improves both control overhead and network convergence time. Balmau *et al*. addressed a similar issue [51]. Vallati and Mingozzi point out that DIO suppression of Trickle (due to redundancy count k) incurs unfair DIO transmission frequency among nodes, resulting in a trade-off according to k parameter: route quality vs. control overhead [52]. For fair DIO transmission, the authors designed Trickle-F that suppresses DIO considering previous suppression history. Tripathi and Oliveira tackle a trade-off in the *DelayDAO* timer: latency vs. congestion [53]. Although DAO messages are used for building downward routes, they are required in networks with any downward traffic even if majority of the traffic is for upward data collection. Furthermore, DAO messages themselves are sent upward towards the root, and contribute to congestion in the network. The authors adaptively control *DelayDAO* timer by measuring DAO round-trip time with use of (optional) DAO-ACK feature of RPL to mitigate congestion and improve the packet delivery performance of the network. These four studies evaluate RPL and their proposals only through simulations.

- **Standard-related discussion 2**: Although the RPL standard provides some default parameter values, it leaves other parameter choices to individual implementations. RFC6550 says; "*these default values are likely to change with the context and as the technology evolves*". Given that RPL targets various applications, impact of each parameter on RPL's performance needs to be further explored in various environments, which can be useful guidelines for future implementations.
- **Implication 2**: Parameter optimization and/or adaptation are necessary for applying RPL in real-world, since it critically impacts important performance metrics of LLN such as battery lifetime and reliability.

B. Downward routing

Even though RPL focuses mainly on upward traffic delivery, many simple sensor monitoring applications, such as AMI, require electric meters to be 'configured' or actuated in the

downward direction. Moreover, the use of TCP and/or application layer ACKs will mandate bi-directional connectivity [7]. Furthermore, as LLN applications are being diversified, their traffic patterns are diversified as well. Specifically, several applications, such as electronic price tag update in markets, public lighting control, and wireless reprogramming, require reliable downward packet delivery from LBRs to individual LLN endpoints. To support these applications, RPL needs to construct reliable downward routes in addition to upward routes.

However, a number of measurement studies using ContikiRPL and TinyRPL have shown that RPL's downward performance is worse than the upward due to unreliable downlink routes [28], [54]–[56]. Through testbed experiments and field deployment experiences, Kim *et al*. evaluated TinyRPL and found that RPL manages link quality information only for the nodes in the parent set and updates link quality of a parent node only through upward packet transmissions [54] [28]. Note that in RPL, a parent node does not (cannot) detect a link failure nor take any action to fix it even when experiencing consecutive downward transmission failures. Only a child node can fix the problem by changing its parent node. This causes slow route updates in downward traffic-centric (i.e., sparse upward traffic) applications even if actual link quality highly fluctuates.

To alleviate the problem, the authors in [28] propose to eliminate downward routing by enabling the (plugged-in) LBR to cover all the nodes in a single hop with high transmission power. This is an example of how we can use heterogeneous node property (e.g., wall-powered vs. battery-powered) for routing, which is one of RPL's requirements. However, at the same time, this is an indirect and limited solution in that it does not provide multihop downward routing and transmission power is regulated by national standards on spectrum use. ORPL improves the reliability of downward packet delivery by using opportunistic routing capability (i.e., spatial diversity), and showed its performance improvement over ContikiRPL through testbed experiments [45]. However, it still updates the routing metric solely based on upward traffic, which makes downlink underperform uplink. Recently, Kim *et al*. developed DT-RPL which updates link quality both through upward and downward traffic [57]. This is the initial effort for reliable downlink, which needs to be improved further.

- **Standard-related discussion 3**: Reliable downward routing is an important requirement of RPL since it is needed in many applications. Furthermore, downward routing is more challenging than upward routing since RPL focuses on upward route optimization and LLN has resource constraint in terms of memory usage and control overhead. Despite its importance, this topic has been under-investigated compared to upward routing.
- **Implication 3**: Finding reliable downward routes for RPL in dynamic wireless environments is an important but still an (relatively) open research area.

Istomin *et al*. revealed a scalability issue of RPL's downward routing. They performed a simulation study of ContikiRPL and TinyRPL using COOJA simulator and showed that RPL's downlink underperforms dissemination protocols, such as Trickle, in terms of reliability, latency and memory overhead [58]. Specifically, even though RPL outperforms Trickle-based dissemination in sparse topologies thanks to link layer retransmissions, both ContikiRPL and TinyRPL have scalability issues and experience performance degradation in high density scenarios. Between the two, ContikiRPL performs worse than TinyRPL due to neighbor table overflow, frequent global repairs, and topology churns. Even though the authors explicitly note that the simulation environment is not sufficient to define the complex architecture of a city, they did their best to provide realistic results: configuring the simulation topologies according to the real LLN deployments for smart city applications and setting the noise floor and variation values according to the measurements in several testbeds, suburbs, densely inhabited areas and a university campus.

On the other hand, it has been known that RPL's two MOPs for downward routing have their own weaknesses; storing mode has memory overhead while non-storing mode suffers from long packet header and long hop distance [16]. A number of studies have investigated how to efficiently use these two MOPs, which is a standard-related topic. Kiraly *et al*. focused on memory overhead of a storing-mode network, and designed D-RPL to address the issue [59]. Under D-RPL, each node multicasts a downward packet if the destination is not included in its route table, which enables downward packet delivery with small route table size. But, the authors do not provide any experimental results which are essential to evaluate the multicast performance. Duquennoy *et al*. reduced the memory size required for each route table entry of storing mode by using Bloom filter [45], and show its scalability through testbed experiments.

RPL also envisions the use of mixed MOPs in a single network and RFC6550 writes; "*RPL does not support mixed-mode operation, where some nodes source route and other store routing tables: future extensions to RPL may support this mode of operation*". Gan *et al*. take the mixed-MOP approach to relieve memory overhead of storing mode and design MERPL that opportunistically uses non-storing mode in a storing-mode network [19]. Under MERPL, when a node suffers lack of memory, it sends its route table entries to the LBR and removes the entries from its routing table, which enables source routing for those entries. However, MERPL was evaluated using simulations only, which did not show the actual behaviors on resource constrained nodes in real channel environments. Ko *et al*. showed that there exists a serious connectivity problem when two MOPs are mixed within a single network [18]. To address this issue, the authors proposed DualMOP-RPL that supports nodes with different MOPs to communicate gracefully in a single network while preserving the high bi-directional data delivery performance, which was evaluated through both simulations and testbed experiments.

Finally, there are work on supporting P2P communication within RPL framework, which effectively combines the upward and downward routing capability of RPL. Baccelli *et al*. pointed out the inefficiency of the non-storing mode for P2P communications due to large hop distances [60], and

design P2P-RPL that reactively discovers direct routes for P2P traffic by using the flooding-based route discovery of AODV, and show its effectiveness through testbed experiments. P2P-RPL was standardized in RFC6997 [61], a mechanism for P2P-RPL to suppress flooding-based routing control overhead was provided in [62], and Zhao *et al*. further investigate its performance through NS-3 simulations in [26]. However, these studies do not evaluate routing overhead of P2P-RPL nor compare it to that of RPL, despite its importance. Zhao *et al*. designed ER-RPL to reduce flooding-based route discovery overhead of P2P-RPL [63]. ER-RPL includes geographical location information in DIO and uses geographical distance to suppress flooding overhead. Mathematical analysis and NS-3 simulations were used for its performance evaluation including routing overhead and energy consumption. However, no experimental results are provided even though it is questionable whether location-based routing can be applied in complicated real channel environments and flexible deployment scenarios.

- **Standard-related discussion 4**: Regarding efficient and compatible operation of the two MOPs in RPL, many interesting issues have been brought up by research community. However, more systematic and large-scale experimental studies are needed to provide convincing results. Also, it is surprising that non-storing mode is not implemented in any official open RPL implementation[7]. Furthermore, as important future work, we believe that the research should go beyond interoperability between the two MOPs and seek to improve performance of mixed-MOP networks in real environments.
- **Implication 4**: Efficient and interoperable MOP operation is a valuable and RPL-specific research topic, which is directly related to tight resource constraints and heterogeneity in LLN.

C. Load balancing

Since an LLN node is usually a battery-powered device, it is a requirement of RPL to be energy efficient. Given that human intervention is likely to start when the first-dead node occurs, it is more crucial to maximize the minimum lifetime of a node than the average lifetime of all nodes. Thus, RPL needs to balance the traffic load of each node to provide fair energy usage among nodes. Furthermore, in large-scale applications such as smart grid and building automation, nodes near the LBR have to relay heavy traffic even if each node generates light traffic. The RPL standard aims to support LLN comprising thousands of nodes and Cisco's field area network (FAN) solution for smart grids (CG-Mesh) [29] is a good commercial example of large-scale LLNs, which supports up to 5,000 nodes per LBR and envisions millions of nodes within a FAN. Thus, the load balancing issue also needs to be investigated under heavy traffic (congested) scenarios.

RPL decouples the load balancing issue from the main standard (RFC6550) and expects the design of OF to handle it. Two OFs, OF0 [12] and MRHOF [13], are provided by companion standards. However, a number of experimental studies have reported that TinyRPL, RPL with the default OF (OF0) along with the hop count metric for RANK calculation and the ETX for parent selection, has a load imbalance problem since it only focuses on finding a parent node with good link quality, resulting in both unfair energy consumption among nodes and congestion at some bottleneck nodes [56], [64]–[66].

To achieve fair battery lifetime among nodes, Nassiri *et al*. proposed a new parent selection mechanism for RPL [67]. It generates the parent candidate set considering both received signal strength indicator (RSSI) and residual energy, and probabilistically selects the parent node for each data transmission considering traffic load. Iova *et al*. present a new routing metric, expected lifetime (ELT) that incorporates traffic load and link quality [68]. By using ELT, the authors proposed a multi-parent routing for RPL to balance energy consumption among nodes. These studies evaluate their proposals through WSNet simulations [69], which cannot show the behavior of resource constrained nodes. Gaddour *et al*. designed OF-FL, a new objective function that provides a fuzzy logic-based parent selection considering various factors such as residual energy and link quality [21]. The authors compared OF-FL to ContikiRPL with OF0 and MRHOF and showed that OF-FL balances energy consumption among without sacrificing packet delivery performance, but used only simulations.

Some work adopted testbed experiments to show load balancing effects of their proposed schemes. Michel *et al*. investigate load imbalance problem of ORPL and propose ORPL-LB that achieves load balancing under ORPL by sleep interval control and selective ACK transmission [70]. Through testbed experiments, the authors show that ORPL-LB provides fairer battery lifetime among nodes than RPL and ORPL. Most recently (June 2016), Oliveira *et al*. designed ALABAMO that adds load balancing capability to MRHOF [71]. Under ALABAMO, a node propagates the number of transmitted packets by embedding it in DIO, which enables parent selection to consider both traffic load and ETX. They showed the load balancing effect of ALABAMO through testbed experiments which incorporate human activities and various interference sources. But they did not consider a duty-cycling mechanism for performance evaluation and simply assumed that fair relay burden can balance battery lifetime; the impact of load balancing on energy consumption needs to be investigated further.

For congestion alleviation, Liu *et al*. designed LB-RPL that exploits queue utilization to achieve load balancing in a large-scale LLN [72]. An LB-RPL node detects the queue utilization information of its neighbors from how long a neighbor delays its DIO transmission; if a node is congested, it delays the dissemination of routing information. For each data packet transmission, a node probabilistically selects its parent node by using the queue utilization information of its parent candidates. Lodhi *et al*. designed M-RPL which detects traffic congestion through DIO and DAO messages and provides two parent nodes for each node to distribute traffic load [73]. However, these works evaluate their schemes using NS-2 and COOJA simulations only, respectively, and neither conduct experiments

[7]Regarding ContikiRPL, please refer to footnote 12

in a real LLN nor implement the proposed schemes on real embedded devices.

Several other works evaluate schemes through testbed experiments in congested scenarios. Kim *et al*. evaluate TinyRPL on an indoor testbed and show that link-quality metrics such as ETX cannot detect RPL's load imbalance problem under heavy traffic scenarios because a small packet queue of a resource constrained LLN node starts to overflow even before link congestion occurs [20]. To address the issue, the authors designed QU-RPL that achieves load balancing by using a queue utilization-based routing metric and probabilistic parent change. They provided more detailed information of QU-RPL's behavior with additional experiments on another larger testbed in [65]. They did all experiments during the night-time to focus on load balancing effect rather than link dynamics, which makes RPL's or QU-RPL's load balancing behavior in fluctuating link environments still unrevealed. Boubekeur *et al*. propose BD-RPL which restricts the subtree size of each node to relieve congestion [74]. That is, each node accepts the joining request of its child node only when its subtree size is smaller than a predetermine threshold. The authors evaluated their proposal compared to ContikiRPL both through COOJA simulations and testbed experiments. In another work, Kim *et al*. also proposed PC-RPL which achieves better throughput and routing stability by jointly and adaptively controlling the routing topology and transmission power of individual nodes within RPL [66]. The authors evaluate their proposal through experiments on a 49-node multihop testbed.

- **Standard-related discussion 5**: Even though RPL expects a load balancing-aware OF to be designed in companion standards, none of standardized OFs support load balancing. This resulted in many load balancing studies, which also shows the importance of load balancing in LLNs. Building on these prior efforts, further experimental studies incorporating both link dynamics and load balancing effects can be meaningful future work, given that load balancing must be considered jointly with reliability in LLNs.
- **Implication 5**: Load balancing is an important and practical issue of RPL, given that scalability, resource constraint and energy efficiency are main characteristics of LLN.

D. Interoperability

For LLNs to be widely applied in industry, hardware and software implementations from different vendors are required to inter-operate and perform well together. To this end, RPL provides some guidelines for interoperability, and several work have investigated this issue.

Given that TinyRPL and ContikiRPL were implemented separately, several works investigate the interoperability issue when TinyRPL nodes and ContikiRPL nodes are mixed in a single network. Ko *et al*. evaluated the performances of an LLN comprising both RPL implementations and show that ContikiRPL and TinyRPL are inter-operable [9]. However, more importantly, their simulation results revealed that parameter selection (e.g., queue size and retransmission interval) and implementation details have significant effects on the performance and the routing layer's behavior. The authors did another measurement study of the mixed LLN through testbed experiments and showed that the testbed results are worse than the simulation results, but have the similar trend [75]. Guan *et al*. investigated the behavior of the mixed RPL network more deeply through COOJA simulations and testbed experiments [76]. Their results showed that ContikiRPL nodes require longer time for processing a received packet than TinyRPL nodes, which causes more packet drops at ContikiRPL nodes under the mixed RPL scenarios. Due to the same reason, a TinyRPL-based LBR always outperforms a ContikiRPL-based LBR. Furthermore, ContikiRPL generates more DIO transmissions than TinyRPL by resetting *TrickleTimer* more frequently. The results of these practical experimental studies imply that RPL may need to provide more concrete guidelines for parameter selection and implementation details to achieve not only interoperability but also reasonable performance guarantees when heterogeneous RPL implementations co-exist.

A number of studies investigate interoperability between RPL and other layer protocols. Given that RPL needs to support TCP both for performance and legacy compatibility reasons, Kim *et al*. investigated the interoperability issue between TCP and RPL [56]. They provide a measurement study of TCP over TinyRPL on a multihop testbed network and show that TCP and RPL are inter-operable. However, round trip time (RTT) of TCP increases with hop distance given by RPL, which incurs throughput unfairness among nodes when using TCP over RPL. The authors have also found that light-weight implementation of TCP[8] may result in unexpected (from the perspective of full TCP) behavior. The work in [28] investigates the interoperability issue between RPL and a duty-cycling protocol (i.e., lower layer) and shows that RPL's parameters need to be set considering lower-layer characteristics. Specifically, the minimum transmission interval of *TrickleTimer* is required to be larger than the sleep interval of a duty-cycling protocol to mitigate congestion and queue overflow. M. Stolikj *et al*. point out a similar issue but take a different approach [77]. Instead of tuning *TrickleTimer* parameters, the authors designed a cleansing MAC that purges obsolete messages given by Trickle and showed performance improvement through both simulations and testbed experiments.

- **Standard-related discussion 6**: Even before the standardization of RPL, whether the conventional network layering approach should be applied in LLN was a popular discussion topic in WSN community. This is because, in contrast to traditional networks, LLN is an often *stand-alone* network. In other words, in LLNs, it is hard to assume that other layers are already designed well enough for any application and expect RPL to only fill the empty slot (routing layer) as an independent layer. Instead, for each application, all layers interact differently and need to be optimized *together* in an *application-specific* manner. However, RPL provides only vague description on inter-layer operability issue and leaves

[8]Verified using TinyOS/BLIP, but equivalent implications for ContikiOS/uIP

open many challenges related to system-level performance. The inter-layer interaction needs to be further investigated on top of valuable prior efforts in WSN community, such as in [78].

- **Implication 6**: Cross layer interoperability and/or performance optimization are not optional but mandatory to apply RPL in reality, given that it is designed for LLN.

Although almost all RPL-related work investigates various RPL's characteristics on IEEE 802.15.4, RFC6550 says; "*RPL is designed to be able to operate over a variety of different link layers, ... such as but not limited to, low-power wireless or PLC (Power Line Communication) technologies*". Given that RPL considers smart grid as a crucial application and PLC is a major candidate of link layer technologies for smart grid, several studies investigated how RPL can operate on PLC.

Even before RPL standardization, Chauvenet *et al*. first showed an experimental evidence that RPL can operate over PLC [79]. The authors interconnected PLC under RPL by implementing IEEE 802.15.4 stack over PLC [80] and confirmed basic functions. Furthermore, they show IPv6 packet exchanges between a multihop PLC network and a multihop wireless network through Contiki-based testbed experiments. However, in terms of performance, their results reveal that PLC requires much longer latency (i.e., up to several seconds of roundtrip time for a packet) than IEEE 802.15.4 (250kbps) due to its slow data rate (10kbps). Saad *et al*. implemented a PLC module in COOJA simulator and compared the performance of RPL over PLC in COOJA and on a testbed [81]. The authors showed that the simulation results are comparable to the testbed results, which validates their implementation. However at the same time, these results revealed the performance limitation of PLC medium: $<$1kbps throughput and $<$90% reliability for one-hop links and worse performance for multihop nodes in a small-scale network with 7 nodes. Ropitault *et al*. implemented IEEE P1901.2 module as a narrowband PLC technology on OPNET and compared the 'RPL over PLC' performance of OPNET simulations and testbed experiments [82]. The authors also showed that *DelayDAO* timer value of RPL needs to be adjusted for RPL over PLC [83]. Balamau *et al*. implemented IEEE 1901.2 as a MAC layer to support medium voltage PLC and evaluated the performance of RPL over PLC through COOJA simulations [34]. After these fundamental work during 2010 $\sim$ 2014, to the best of our knowledge, no further work has been reported in the literature.

On the other hand, recently (June 2016), Lee *et al*. investigated the interoperability issue between RPL and Bluetooth low energy (BLE), motivated by the fact that BLE is becoming more popular than IEEE 802.15.4 and is in many commercial products [84]. Through testbed experiments, the authors revealed that RPL experiences severe performance degradation when operating on top of BLE, even though RPL was designed to operate on various link layer protocols. To address the issue, they designed an adaptation layer between RPL and BLE, named ALBER, that modifies control packet transmission mechanism, parent change mechanism, and routing metric of RPL considering BLE's unique features and showed that RPL over BLE outperforms RPL over IEEE 802.15.4. We feel that more research effort is needed in this area as we expect a more diverse set of links in the coming IoT era.

E. Multicast

Multicast is an important communication service, not only applications, but also control and management purposes: service discovery, addressing and publish/subscribe schemes. However, it is one of the most vaguely specified features in the RPL standard (within one page). As a result, Cisco's CG-Mesh network, a 5000-node LLN that uses RPL, is known to use simple flooding for IPv6 multicast[9], which, intuitively, is not the most effective way to do multicast in a large-scale multihop network. Afterwards, IETF standardized IPv6 multicast protocol for LLNs (MPL) [85], which provides two multicast modes: Trickle [11] and classic flooding. Trickle is one of the most popular multicast protocols in WSN community and there have been numerous related work since it was first introduced in 2004 (1053 citations until October 2016) [86], such as interoperability issues with link layer in [28] and [77], which is out of scope of this survey.

Several works investigate how to use RPL's routing topology information for efficient multicasting. Oikonomou *et al*. show that RPL's multicast section leaves many questions unanswered and this issue had been overlooked despite its importance [87]. The authors also point out that even though Trickle suppresses re-broadcasting overhead considering whether a received packet contains new information or not, it still has redundant transmission overhead *without using topology information*. Moreover, Trickle rapidly increases transmission interval, which causes large delay when Trickle operates with a duty-cycling mechanism due to packet collision. To alleviate the problem, they designed SMRF that allows a node to forward a received multicasting packet only once, only when it receives the packet from the preferred parent and has interested *children* nodes; SMRF suppresses multicasting overhead by using routing topology given by RPL. Both through simulations and testbed experiments, the authors show that this downward-only multicasting scheme improves delay performance with less overhead, while sacrificing packet delivery ratio. The authors do not argue that SMRF is better than Trickle but say; "*The choice of multicast forwarding algorithm should be based on the anticipated usage of a sensor deployment*".

Tharatipayakul *et al*. design iACK that adds an implicit ACK technique [88] to SMRF by using additional memory, and evaluate its performance through simulations, showing that it improves SMRF in terms of reliability and delay [89]. Fadeel and Elsayed design ESMRF which extends SMRF for bi-directional multicasting service [90]. If a source node is a non-root node, ESMRF embeds the payload in an ICMPv6 message and unicasts it to the LBR so that the LBR multicasts the packet using SMRF. This enables any node to multicast packets through the LBR. Lorente *et al*. tackled that ESMRF's unicast packet delivery to the LBR for triggering multicasting is inefficient and designed BMRF as a solution [91]. While a packet is being unicasted to the LBR, BMRF allows each

[9]As of 2014. No further information released after that yet.

intermediate node to deliver the packet not only upwards but also downwards for the interested children nodes. For reliability and duplicate mitigation, the intermediate node unicasts the packet to the parent node and each child node. However, they evaluate their proposal only through simulations, which cannot reflect unreliability of downward routing under link dynamics (already shown in section IV-B).

- **Standard-related discussion 7**: We see the multicasting issue as one of the most under-investigated areas considering its importance in practice. None of the proposals on multicast/RPL compared their work against MPL standard, and only one of them provides experimental results, which leaves most claimed results unclear. On the other hand, on top of numerous studies on Trickle, design of an efficient multicast protocol by combining RPL's routing capability with Trickle could be a valuable research topic.
- **Implication 7**: Multicasting issue has been under-investigated, which may be one of the reasons for slow RPL adoption in practice.

F. Multi-sink

Multi-sink network is a popular research area in the network community. A number of analytic and simulation studies show that using multiple sinks can improve performance in terms of connectivity, latency and energy consumption [92]–[95]. RPL also provides multi-DODAG operation with multiple LBRs as a key feature to support large-scale applications, such as smart grid and agricultural/habitat monitoring. Even if a single LBR is sufficient to cover the deployment area, using multiple LBRs is strongly recommended to achieve robustness and resilience in case of sudden outage or breakdown of an LBR in practice.

For the multi-sink RPL network, several studies reveal that RPL has a load imbalance problem, not only in a single DODAG but also among multiple DODAGs in a DODAG forest. To alleviate the problem, Kulkarni *et al*. designed TREEB that propagates the DODAG size through DIO messages and balances traffic load among DODAGs by using a new routing metric which considers both *RANK* and DODAG size [96]. Ha *et al*. designed another cross-DODAG load balancing mechanism, named MLEq [97]. Under MLEq, each LBR monitors its traffic load, shares the traffic load with other LBRs through backbone, and calculates its ideal traffic load. Then, it determines the priority of its own DODAG by comparing the ideal load and the current load and propagates this information through DIO messages, which is used for parent selection of each node to achieve load balancing. Thulasiraman proposed a new routing metric RI^{3}M that incorporates traffic load and interference for cross-DODAG load balancing [98]. However, they evaluated their schemes through COOJA and NS-2 simulations, respectively, without any implementation and experiments on resource-constrained LLN nodes.

Kulkarni *et al*. design a PHY layer technique to support multi-DODAG operation on multiple channels [99] [100]. They propose a multi-channel scanning scheme for RPL that sequentially scans all the frequency channels to find all DODAGs and joins the best DODAG, which is similar to the ZigBee approach [101]. When losing connectivity at the currently operating channel, a node tries to join the DODAG at the second best channel. The authors extensively evaluated their scheme both through large-scale COOJA simulations and testbed experiments.

Deru *et al*. point out that if each LBR has its unique DODAG ID (subnet prefix), inter-DODAG parent change becomes more complex than intra-DODAG parent change since it requires global address re-allocation [102]. The authors facilitate multi-LBR operation by configuring a single DODAG with multiple LBRs rooted at a backbone node, which enables multiple LBRs to use the same prefix and a node to use the same global address after changing its LBR. The authors confirmed the operation of their proposal through testbed experiments. They observed routing loops when an LBR failed, but left the issue as an open problem.

Andrea and Simon designed a RPL-based multi-sink network, named HOIST, for an agricultural monitoring system [103]. HOIST is a three-tiered network comprising a mobile sink, multiple static LBRs and many static sensor nodes. Each LBR collects data from static sensor nodes over RPL and saves the information. Then, a mobile sink passes through the deployed area to gather the data from the LBRs, using a modified RPL for the mobility scenario. HOIST was evaluated through small-scale COOJA simulations and experiments only with four nodes, which is far from wide-area monitoring applications.

To the best of our knowledge, little research has studied multi-LBR operation despite its importance in real use cases. RPL's routing behavior in a multi-LBR network can be more complex than a single DODAG and needs to be further investigated with resource constrained nodes in real channel environments.

G. Multi-instance

In practice, many applications generate heterogeneous traffic with different QoS requirements. For example, smart grid generates both regular metering traffic that requires reliable delivery and alarm traffic that requires low latency. To support heterogeneous traffic, RPL provides capability of using multiple instances in a single network. That is, RPL can construct multiple routing topologies with different routing metrics in a single physical network (e.g., an ETX-based topology (MRHOF) and a hop distance-based topology (OF0)).

Rajalingham *et al*. designed a multi-instance-based RPL, named RPL-M to support smart grid [104]. RPL-M constructs two routing topologies in a single network; one uses ETX routing metric to achieve reliable delivery of metering data, and the other uses hop count as the routing metric to minimize latency of alarm traffic. However, the authors used 802.11b for the underlying link layer, which is impractical in LLN scenarios. Banh *et al*. evaluated the performance of multi-instance RPL under heterogeneous scenarios using COOJA simulator, which showed that it provides better QoS for each traffic type than single-instance RPL [105]. Barcelo *et al*. point out that RPL does not provide any mechanism to distribute traffic to multiple instances [106]. They designed C-RPL that

coordinates the collaboration among RPL instances based on game theory. These approaches are conceptually similar to the RPL's 'Link Color Object' idea proposed in RFC6551 [10]. However, none of these works either implement the proposed schemes on real embedded devices or do experiments in real environments.

- **Standard-related discussion 8**: RFC6550 mentions 'multi-instance' as a key feature of RPL in the design principle (section 1.1). However, without any experimental evidence, it is hard to say that using multi-instance is a viable approach for heterogeneous traffic delivery (e.g., OF0 for alarm traffic and MRHOF for regular monitoring traffic), since it is also possible that a shortest route requires even more latency than a minimum-ETX route due to poor link quality (i.e., many retransmissions). Given that the memory space of TelosB platform is almost full when operating a single RPL instance network with the current IPv6/6LoWPAN/RPL stack in ContikiOS or TinyOS, it seems that researchers have primarily focused on other issues that can be investigated with the single instance setup. This should be changed with modern devices that have more memory.
- **Implication 8**: Even though using multiple instances for a RPL network is reasonable in theory, it may be far from *current* needs in practice and/or should be revisited considering implementation complexity and resource constraints of embedded nodes.

H. Interference

Mitigation of wireless interference at the 2.4 GHz ISM band is a popular research area in LLNs since many practical application environments have significant external interference due to other devices in the same frequency band [28], [107]–[109]. Most previous work focuses on improving PHY and MAC layers for interference classification [110], [111], adaptive duty-cycling [112], [113], and error recovery [114], [115] [116]–[118]. Most of these works do not consider routing layer's behavior under interference [110], [111], [114]–[117]. Others test the performance of their schemes in a multihop network without suggesting any improvement to the routing protocol [112], [113], [118].

Even though interference mitigation is not a routing layer issue, link characteristics and the underlying link layer's behavior heavily impacts routing topology, which calls for studies of RPL's behavior under interference. There have been a few works that evaluate RPL under interference scenarios. Han *et al*. experimentally evaluated RPL's performance under WiFi interference [107]. Their results showed that RPL experiences not only severe packet losses but also a large number of redundant parent changes (i.e., topology churns) in the presence of wireless interference. Mohammad *et al*. did a measurement study at various places, such as shopping malls, parking lots, residential complex, and cafeteria, which confirmed the existence of wireless interference at those areas [108]. To mitigate interference, the authors design a cross layer solution, named Oppcast, that combines opportunistic routing with a simple frequency hopping mechanism and a receiver-initiated MAC protocol, and showed that Oppcast outperforms both RPL and ORPL in a testbed network and various real-world fields. Lee *et al*. proposed to use BLE (instead of IEEE 802.15.4, suggested by 6LoWPAN [119]) under RPL to avoid interference because BLE has an adaptive frequency hopping mechanism [84]. Through testbed experiments, they showed that RPL over BLE provides more reliable packet delivery than RPL over IEEE 802.15.4 under wireless interference.

I. Mobility

By design, RPL standard does not consider mobility support. More precisely, there is no mechanism in the RPL standard that is exclusively for, or to explicitly support, mobility, and it is stated that mobile nodes should not forward information. As so RPL does not consider router nodes to be mobile, but only leaf nodes can. However, mobility is becoming an essential part for many important applications of clinical or industrial environments [120]–[122]. These applications require a hybrid multihop network consisting of both static backbone nodes (routers) and mobile nodes (e.g. medical staffs and patients in a hospital, robots and machines in a factory). Thus, a new challenge for RPL is to provide seamless connectivity for mobile nodes in an efficient manner.

Several studies pointed out that RPL experiences significant performance degradation when operating with mobile devices since it does not identify mobile nodes nor provide any specific operation for the mobile nodes. To alleviate the problem, Korbi *et al*. designed ME-RPL which identifies mobile nodes by using an option field in DIO message [123]. Under ME-RPL, a node gives lower priority to mobile parent candidates than static candidates for the parent selection procedure, resulting in more stable connectivity. An ME-RPL node also reduces the DIS transmission interval as it experiences more frequent parent changes, which achieves fast neighbor discovery in unstable environments. Cobarzan *et al*. proposed to mark and identify mobile nodes as in ME-RPL, but restrict the function of a mobile node more strictly than ME-RPL does; a mobile node operates only as a leaf node [124]. To support seamless handover, they make a node having a mobile child node exploit a reverse *TrickelTimer* that exponentially decreases the DIO transmission interval after each DIO transmission. The intuition behind this strategy is that a mobile child node is more likely to be disconnected from the parent node as time goes by.

Other work focuses on providing fast neighbor discovery for seamless handover of mobile nodes. Lee *et al*. modified three features of RPL to support fast neighbor discovery; 1) fast ETX update for a new neighbor node by sending ping packets to the new node immediately, 2) fast *RANK* update by fixing DIO transmission interval (i.e., no *TrickleTimer*), and 3) fast topology reconstruction by sending DIO and DAO messages right after each parent change [125]. Ko *et al*. design a mobility support layer between routing and link layers, named MoMoRo [126]. It checks connectivity between a parent-child link based on upward packet losses, quickly gathers neighborhood information by requesting a unicast reply from each neighbor node immediately after losing connectivity, and

identifies good-link neighbors by using a fuzzy estimator. Fotouhi *et al*. designed mRPL [127] which exploits RSSI for connectivity check and transmits a burst of DIS messages for neighbor discovery when the link quality becomes bad, triggering a handover procedure faster than MoMoRo. To reduce control overhead, it does not reset the *TrickleTimer* when receiving DIS messages during a handover procedure. Fotouhi *et al*. also designed mRPL+, an improvement of mRPL, which allows a parent node to update RSSI values not only for its children nodes but also all neighbor nodes by overhearing data packets [128]. This enables soft handover without burst DIS transmissions in mRPL.

Position information has been used for mobile routing as well. To represent the position information, Goddor *et al*. designed Co-RPL that defines corona ID as the minimum of hop distance from the LBR to each of all neighbor nodes (not the hop distance from the LBR to the parent node) [129]. It improves performance of RPL in mobile scenarios by replacing *RANK* with corona ID. KP-RPL [130] used ETX-based routing between static nodes while using position-based routing between static and mobile nodes, given that all mobile nodes are leaf nodes. Under KP-RPL, each mobile node uses RSSI and Kalman filtering both for position estimation and good-link detection, This is hard to be applied to practical LLN environments due to the use of a predetermined channel model and complex mathematic calculation.

In a scenario where medical applications deliver mobile patient's data not only to the LBR but also to mobile medical staffs, Carels *et al*. consider point-to-point routing under mobile scenarios [131]. To efficiently update downward routes when a node changes its parent node, the authors designed a common ancestor of both old and current parent nodes to transmit a no-path DAO downwards through the old route, which removes outdated routes without end-to-end transmissions of no-path DAOs (from a mobile node to the LBR).

Finally, Oliveira and Vazao survey research on RPL-based mobility support [24]. In addition, the authors evaluate the four routing protocols presented in [123]–[125], [129] through COOJA simulations. The results reveal that a RPL-based mobile routing protocol suffers from severe performance degradation due to large amount of control traffic if it keeps up-to-date routing table. In contrast, less responsive protocols with fewer control traffic provide better packet delivery performance.

J. Performance comparison with LOAD(ng)

In addition to RPL, recently, IETF is drafting another routing protocol for LLNs, LOADng [23] (LLN On-demand Ad-hoc Distance-vector Routing Protocol - Next Generation), which is a lightweight version of AODV (one of the most representative routing protocols in MANET) [132]. LOAD [22] (6LoWPAN Ad Hoc On-Demand Distance Vector Routing) is another earlier Internet draft proposal by different group of people with the same principle and idea of being a lightweight version of AODV for LLN [10]. Given that AODV has been extensively evaluated both through simulations and experiments [133]–[139] and it can support both static and mobile networks, LOAD/LOADng has the potential to be an alternative of RPL. Some researchers have compared the performance of RPL (proactive route management) and LOAD/LOADng (reactive route management) in LLN environments.

Herberg and Clausen compare the performance of RPL and LOAD through NS-2 simulations under bi-directional traffic scenarios [140]. They showed that RPL provides similar reliability to LOAD, but requires more control overhead than LOAD. However, the results are somewhat misleading since the authors set DAO transmission interval of RPL to 15 seconds. This value is much smaller than that required for route management in most static deployments (e.g., Cisco's CG-Mesh network uses 15 minutes), resulting in excessively high control overhead.

Vucinic *et al*. compare the performance of RPL and LOADng through COOJA simulations [141]. They argue that the results in [140] comes from too small DAO transmission interval and show contradictory results; RPL provides lower control overhead than LOADng due to the use of *TrickleTimer*. Furthermore, RPL outperforms LOADng in terms of latency, memory overhead, and hop distance due to its proactive route management. In contrast, LOADng reactively constructs routes through flooding of RREQ and RREP messages, which incurs longer latency to wait for the end of flooding procedure, more memory overhead when saving redundant path information, and longer hop distance when the source receives the first RREP message from a non-shortest path. Tripathi and Oliveira did another simulation study of RPL and LOADng in large-scale networks (up to 2400 nodes) using OMNET++ simulator [142], which shows similar results to the work in [141]. In addition, they revealed that, for LOADng, control overhead increases with the number of application modules, since each traffic session requires a flooding procedure to setup an end-to-end route. In contrast, RPL decouples control overhead from the number of application modules by managing only one DODAG topology in a network. Lastly, Elyengui *et al*. evaluated RPL and LOADng through COOJA simulations under bi-directional traffic scenarios, which revealed that RPL provides less delay, less overhead, and higher reliability than LOADng [143].

Overall, these works show that RPL is a *relatively* good choice to support LLN environments compared to LOADng. Note that this does not mean that the current RPL design is sufficient to be applied to real LLN applications.

K. Security

RPL specifies a security mechanism to protect its routing control messages and topology from *external attackers*, but it explicitly notes that this security mechanism is OPTIONAL to implement. RFC6550 says; "*It may be economically or physically impossible to include sophisticated security provisions in a RPL implementation. Furthermore, many deployments can utilize link-layer or other security mechanisms to meet their security requirements without requiring the use of security in RPL*". Given that a practical network will most likely have

[10]Currently, LOAD seems to have been deprecated while LOADng is still active.

a security mechanism at the link layer (e.g., Cisco's CG-Mesh) which also protects routing messages, to the best of our knowledge, there has been no published research on the security feature of the RPL specification, neither for evaluation nor for improvement.

Is a RPL network free from any security issue with the help of a link layer security mechanism? Based on the survey results, our answer is "probably NOT"; Interestingly, the lack of research on the RPL's *own* security mechanism does not mean that RPL does not have security concerns. Rather, a number of studies (12 papers) investigate RPL's security issues in a different perspective, *internal attackers* that freely transmit and receive secured packets as authenticated members of the network. The researchers tried to manage routing topology and control overhead even when routing control messages fail to be secured (i.e., link layer security failures). This issue is not considered in the RPL standard, which implies that, in practice, ∼14 pages (9.4%) of the current RPL specification that describe the security mechanism needs to be revisited.

A number of studies have reported that RPL is weak for internal attackers, presenting numerous possible attacks: version number attack, *RANK* attack, DIS transmission attack, high-powered DIO transmission attack, Sybil/clone attack, sinkhole/wormhole attack, redundant local repair attack, and selective forwarding attack [144]–[148]. These attacks spoil the routing topology and/or incur large control overhead, which prevents reliable packet delivery and shortens battery lifetime.

Several works develop defenses against internal attacks, including one-way hash chains to defend version number and/or *RANK* attacks. For the version number attack, an attacker maliciously increases the RPL version number to incur frequent global repairs, which increases control overhead. For the *RANK* attack, an attacker decreases its *RANK* to spoil the routing topology and attract traffic from neighbor nodes, which degrades packet delivery performance when combined with sinkhole, wormhole, or selective forwarding attacks. To alleviate these problems, Dvir *et al*. designed VeRA, which uses two one-way hash chains to prevent attackers from both increasing version number and decreasing *RANK* [149]. However, they neither implemented VeRA nor evaluated it through simulation or experiment. Weekly and Pister also secure *RANK* information by using a one-way hash function and evaluate their scheme through C++ simulations [150]. Khan *et al*. propose a Merkle tree-based authentication protocol that uses hashed information and evaluate the scheme through NS-2 simulations [151]. Ye *et al*. consider the scalability issue of the *RANK* authentication, given that using a one-way hash chain to secure 16-bits of *RANK* information leads to unacceptable authentication delays [152]. To address the issue, they quantize the *RANK* information to reduce the length of the hash chain without simulations nor experiments.

A number of studies propose to check the validity of an end-to-end route by message exchanges between each LLN endpoint and the LBR. Weekly and Pister combined a parent fail-over technique with *RANK* authentication scheme to mitigate sinkhole attacks [150]. Given that the LBR knows traffic rate of each node, it calculates end-to-end packet reception ratio (PRR) for each node, makes a blacklist consisting of nodes with too low PRR, and propagates the blacklist through DIO messages. Wallgren *et al*. design Heartbeat protocol to authenticate end-to-end routes against selective forwarding, sinkhole, and wormhole attacks [153]. Heartbeat uses an end-to-end ICMPv6 echo/reply message exchange to validate an end-to-end path between a node and the LBR. These works are sensitive to parameter selection: PRR threshold or ICMPv6 message interval. Perry *et al*. designed TRAIL, which validates an end-to-end path by using *RANK* attestation scheme and uses Bloom filter to minimize the size of control messages [32]. The authors provide security proof and evaluate their scheme through testbed experiments[11]. Landsmann *et al*. recently presented a demo to show the effect of TRAIL [154].

Mayzaud *et al*. address topological inconsistency attacks, which maliciously trigger local repairs (i.e., frequent resets of *TrickleTimer*) [155]. They restrict the number of *TrickleTimer* resets per hour by using a threshold and propose an adaptive threshold control scheme, named AT, which reduces both control overhead and energy consumption. They evaluate AT through COOJA simulations. Finally, Heo *et al*. evaluates the performance of RPL under external stealthy jamming attacks, and show how performance degrades depending on the RPL topology [118]. To overcome the issue, they present Dogde-Jam, a light-weight anti-jamming technique, that uses ACK channel hopping and multi-block data shifting to resist against the stealthy jamming attacks. They implement Dodge-Jam on TelosB nodes, and evaluate its performance through experiments on a multihop LLN testbed.

- **Standard-related discussion 9**: Overall, the survey results present the security issue as one of the subtlest aspects of LLN and IoT. Although many security issues have been investigated apart from RPL's own security mechanism, without any experimental validation, we cannot say that these works are more valuable than the RPL standard. The conflict between the RPL standard and security-related researches implies that we need to carefully formulate what is the exact security role of the routing layer (routing security) in contrast to the link-layer or end-to-end security, and what are the most relevant attacks in the context of LLN and IoT. For now, we cannot hastily confirm whether RPL's security features are meaningful or not, but it is clear that it should be reconsidered.
- **Implication 9**: Routing security issue should be carefully revisited in the context of IoT and LLN.

V. What has not been studied?

RFC6550 [1] is the core document for the RPL standard, and RFC6551 [10] is the companion standard that defines and describes the routing metrics used for path calculation in RPL. Looking back into the standard, we noticed and identified several parts/sub-parts of the RPL standard that have neither been studied in the literature, nor have been implemented in any of the open-source prototype implementations of RPL that we are aware of. We list these points here.

[11] It is the only one experimental work that deals with RPL's security.

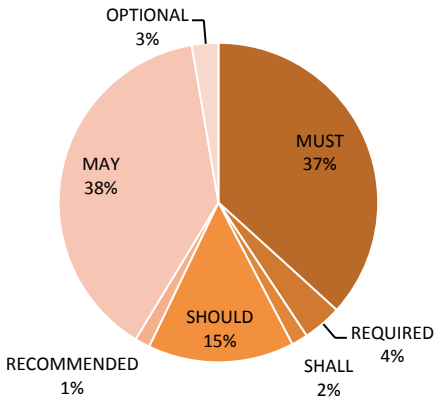


Fig. 8. Distribution of requirement level key words in RFC6550.

First of all, none of RPL's own security mechanisms are implemented in either TinyRPL or ContikiRPL. To be precise, only the "unsecured" security mode of RPL is used; neither "pre-installed" nor "authenticated" modes are used nor implemented. Security features of RPL are described in subsections 3.2.3, 6.1, 6.6, 18.2.7, 20.6∼8, and whole of section 10 of RFC6550, spanning over 14 pages of the standards document. It includes the security modes, secure message formats, key distribution and installation, consistency check (CC) feature, counters, packet processing procedures and more, but none of these mechanisms are implemented in any of the prior work in Section IV. The standard does state that 'unsecured mode does not imply that the RPL network is insecure: it could be using other present security primitives (e.g. link-layer security)', and Cisco's CG-Mesh system is known to use such an approach [29]. However, it is still surprising that there is no RPL implementation (that we are aware of) that implements these security features defined in the RPL standard, and that none of the numerous security-related research work in Section IV-K directly studies these features even after the official standardization.

There are many other features in the standard that are not implemented in both TinyRPL and ContikiRPL. Below are a few. Regarding the downward routing operation, both TinyRPL and ContikiRPL[12] implemented only the 'storing-mode', although some prior work have implemented their own version of the 'non-storing-mode' [18] [29]. However, no prior work seems to have implemented the 'path control' feature, which allows nodes to request for or allow multiple downward routes, described in section 9.9 of the standard. Furthermore, from RFC6551, features such as 'Node State and Attribute Object' (section 3.1), 'Node Energy Object' (section 3.2), 'Throughput/Latency' (sections 4.1 and 4.2), and the 'Link Color Object' (section 4.4) could not be found in any prototype implementations of any prior work that we have found.

It is true that the aforementioned un-implemented features are defined as 'optional' in the RPL standard. Furthermore, our survey of open-source prototype implementations and prior academic research publications might not be exhaustive enough to state that these features were 'never' implemented. However, our position is that *there are too many 'optional' features in RPL*. We acknowledge that this was intended to provide flexibility in the RPL design, but at the same time, it increases the complexity of the standard documents, and hinders more open-source implementations and wide-adoption of the RPL protocol in both academia and industry applications.

To get a glimpse at our hypothesis, we have plotted in Fig. 8 the distribution of 'requirement level key words' [156] in RFC6550. As defined in RFC2119 [156], 1) 'MUST/REQUIRED/SHALL' are the things that must be obeyed from the standard, 2) 'SHOULD/RECOMMENDED' are the things that should be followed unless there exist valid reasons not to, and 3) 'MAY/OPTIONAL' are the things that are left open to the individual implementations. Fig. 8 shows that there are more 'MAY's than 'MUST's, and that the three categories are distributed as 42.3%, 16.3%, and 42.9% where the last 'MAY/OPTIONAL' category is greater than the 'MUST/REQUIRED/SHALL' category. The percentage of 'MAY/OPTIONAL' in RPL (42.9%, RFC6550) is more than IPv6 (31.4%, RFC2460 [157]), AODV (31%, RFC3561 [132]), DSR (25.7%, RFC4728 [158]), OSPF for IPv6 (34.9%, RFC5340 [159]), and RIP-v2 (31.6%, RFC2453 [160]). Of course, the number of words is not a precise measure, but it does indirectly give an idea on how abundant the optional descriptions are in the standard.

Furthermore, many of the suggestions and guidelines proposed in the standard are not clear enough for independent implementations of the standard to interoperate gracefully without manual intervention. For instance, section 15 of the RFC6550 mentions how RPL should/could interoperate with IPv6 Neighbor Discovery (ND) [14] in order to discover neighbors and to perform address resolution and duplicate address detection, but merely suggests that 'care must be taken' rather than proposing a clear instructions on how RPL and IPv6 ND can co-exist (even though RPL is designed as an IPv6 routing protocol). In fact, IPv6 ND and RPL functionalities overlap substantially for router advertisement (RA), router solicitation (RS), neighbor advertisements (NA), and neighbor solicitation (NS), and thus these features can be discarded from RPL, allowing RPL to re-use most of what ND already provides. In addition, the standard specifies that RPL implementations will need to support the use of Neighbor Unreachability Detection (NUD), or an equivalent mechanism, to maintain the reachability of neighboring RPL nodes (Section 8.2.1), but no clear instructions are given even though bi-directional reachability is a critical issue underlying in the RPL operation. These points are also noted in [33] and [161].

Finally, transmission timing of DAO messages is left as an implementation choice. Thus, some implementations use TrickleTimer, some use periodic with and without random jitter, and others use event based transmission only. Furthermore, the timing of, and the events that trigger global repair is

[12] Non-storing mode implementation has been added to ContikiRPL recently (Feb.2 2016) in the latest head of the Contiki GitHub repository. However, the latest official release of Contiki-OS (which is Contiki 3.0 released on Aug.25 2015) still does not have a non-storing mode implementation. This is probably the reason why there is not yet any published research paper that uses the non-storing mode on ContikiRPL.

also left as an implementation choice. These implementation differences could result in significant performance variability across difference implementations, making prior evaluations of RPL hard to compare.

Some of the aforementioned complexity and underspecification problems of the RPL standard have also been noted in [16] before the official standardization, but have not been addressed since. We believe these make RPL's adoption slow, if not impossible.

VI. Suggestions for Future Directions

Our survey of over 97 research papers conveys what aspects of RPL have been studied and what have not. What concerns us most is that there is no real large-scale deployments of a system that successfully uses RPL, at least not in any technical publication that we are aware of. To improve its chance of success, we would like to make suggestions for future directions, both for the standard and for the research community. In this section, we summarize some of the challenges facing RPL standard, and the direction in which the standard must evolve.

A. Considering the Unique Characteristics of LLNs

Our survey shows that many of the challenges that exist within RPL, including the inter-layer interoperability problems, are due to the unique characteristics that LLNs introduce. Specifically, unlike traditional networks, a LLN is typically designed as a *full* stand-alone system. Each system holds its own set of application level requirements, which are reflected in the design and optimization of various networking stack components. As a result, networking components with varying optimization goals at different parts of the stack can cause conflicts and degradation in system-level performance (e.g., more retransmissions leading to higher energy usage) or even lead to interoperability failure. We argue the need for a more comprehensively (but concisely) designed standard, guiding more than a single layer of the LLN networking stack.

B. Simplifying Fancy Features

We believe that 'complexity' is one of the key challenges that RPL faces, making the RPL adoption slow and difficult. 'Complexity' is mainly due to too many features (including optional ones) in the standard, many of which are not used in any implementation, deployment, nor research work so far. Complexity not only makes RPL implementations on resource constrained embedded devices difficult, but also results in vastly different implementations that are not interoperable due to different un-implemented (optional) features.

We argue that this complexity issue must be tackled by removing most, if not all, the unnecessary and optional features of RPL. For example, nodes in a RPL LLN should use the same MOP, OF, metrics, and constraints for interoperability. This renders objective functions and the whole concept of RPL instances useless, as it only leads to more issues with interoperability [161]. Other examples of potentially unnecessary features include many of the security features mentioned in Section V, RPL messages that overlap with IPv6 ND features (RA, RS, NA, NS), and local and floating DODAGs. This simplification also enables to use limited memory space of embedded devices to implement important functionalities other than RPL, such as neighbor management, TCP, and application mechanisms.

C. Providing Complete Interoperability Guideline

Another challenging aspect of RPL is 'interoperability' between RPL implementations, which mainly comes from under-specification of the standard document. This not only adversely affects performance, but may even result in non-interoperable implementations. There are numerous under-specifications in addition to those mentioned in Section V, but we refer to a well documented list in Section 7.3 and 7.4 of [161] for interested readers.

To alleviate the problem, interoperability must be improved by clearly specifying what have been under-specified in the standard. Furthermore, some fancy features of the standard aforementioned in Section VI-B, if not be simplified (to single MOP, single OF, single metric and constraint by removing all redundant features), should be made interoperable in all cases (e.g. even if nodes use different MOPs [18]). This approach can be particularly useful when nodes in an LLN have different capabilities in terms of resource constraints, e.g., some nodes are more powerful devices than others, and can support more features while all nodes are interoperating.

D. Reflecting Industry Effort

To develop a system that can be successfully applied in practice, it is necessary to consider industry trends and demands. Specifically, as IoT is growing quickly with tremendous momentum, RPL should evolve to keep up with industrial requirements. For example, it is a noteworthy observation that many commercialized IoT or WSN systems exploit wall-powered nodes as routers; most, if not all, battery-powered nodes are leaf nodes. This is a simple and practical example of using heterogeneous node capabilities as a way of maintaining a routing capability while providing long battery lifetimes. Such a use of heterogeneous node capabilities in real use cases should be further considered for routing protocol and system designs. Google Nest recently released their source code of 'Thread' [162] which contains another IPv6 routing protocol for LLNs. Following these industry efforts can be a great help to evaluate and improve RPL. Lastly, with rapid development of SoC (System-on-Chip) technology and single-board microcontrollers, it is time to escape from TelosBs (2004 generation) and utilize more recent embedded hardware platforms, such as the Firestorm motes [36] or other Cortex M3-based platforms [163], for RPL deployments.

E. Research Directions

Finally, as a research community, we should encourage and promote meaningful investigations and publications in the direction of reducing complexity and improving interoperability of RPL on top of its performance improvement.

In doing this, we emphasize again that *RPL aims to support LLNs, distinct from traditional networks*. In this point of

view, first, we strongly recommend that new research work should build on valuable prior work from the decade-long research within the WSN community. Second, system-level performance should be evaluated using testbeds with real embedded nodes to reflect real environments. Specifically, as a research community, a systematic approach (including time, node, and environment configuration for each testing purpose) is required for testbed-based performance evaluation, which can reveal not only the performance results but also the reasons behind them. We feel that providing an LLN testbed benchmark for researchers to test their proposals can be helpful [164]. More importantly, we need real-world large-scale deployment evidence of RPL-based applications and systems. Only then, would we be able to make a judgment on whether RPL has succeeded as an Internet standard routing protocol for LLNs contributing to the coming (or already emerging) Internet of Things era.

VII. Conclusion

Our work presents a survey of how the RPL routing protocol has been used and evaluated by examining 97 papers that study RPL with a focus on those that utilize open-source RPL implementations. We noticed that the core portions of RPL have been extensively evaluated through both testbed experiments and simulations for adaptation to numerous application domains. Specifically, 40.2% of the papers have evaluated RPL through experiments using prototype implementation on real embedded devices, but 53.6% have used simulation only for their studies. Among the publications that provide experimental evaluation, ContikiOS and TinyOS were the two most popular implementations (92.3%), TelosB was the most frequently used hardware platform on testbeds (69%), and the testbeds comprise average and median size of 49.4 and 30.5 nodes respectively. Through our studies, we were also able to notice that many of the optional RPL functionalities were not well supported (nor needed) in many scenarios in which RPL was targeted to be applied. Furthermore, despite approximately four years since its initial standardization, we are yet to see wide adoption of RPL as part of real-world systems and applications.

The original design philosophy of RPL was in providing an inter-operable and simple routing protocol for LLNs. However, based on our observations, we make a subjective note that it is questionable whether or not the optional features of RPL, which overly complicate the RPL standard documents itself, along with seldom (or never)-used functionalities such as multiple instance support or security mechanisms, diversify the applications that RPL can really support and allow independent developers to implement inter-operable systems. On the negative-side, we note that such complications may even prevent the adoption of RPL by forcing implementation complexity on the resource-limited computing platforms and also on the developers of such systems.

We started this work to examine the usability of RPL in various applications; specifically, how RPL was implemented, what functionalities were actively used, and under what configurations it was evaluated. With our findings, we ask the following question: "Is the RPL standard, the way it is specified in RFC6550/RFC6551, suitable for providing an interoperable and simple routing solution for real-world LLN applications?" We leave the answer to the research community.

Acknowledgements

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03031348 and NRF-2016R1A6A3A03007799), in part by the industrial infrastructure program for fundamental technologies (N0002312) funded by the Ministry of Trade, Industry & Energy (MOTIE) of Korea, and in part by NSF CPS-1239552.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC 6550*, Mar. 2012.

[2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[3] S. Dawson-Haggerty, A. Tavakoli, and D. Culler, "Hydro: A Hybrid Routing Protocol for Low-Power and Lossy Networks," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2010.

[4] T. W. Ed., P. T. Ed., A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Routing Requirements for Urban Low-Power and Lossy Networks," *RFC 5548*, May 2009.

[5] E. K. Pister, E. P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," *RFC 5673*, Oct. 2009.

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

[7] E. J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks," *RFC 5867*, Jun. 2010.

[8] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power Wireless IPv6 Routing with ContikiRPL," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.

[9] 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)*, Apr. 2011.

[10] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," *RFC 6551*, Mar. 2012.

[11] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," *RFC 6206*, Mar. 2011.

[12] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," *RFC 6552*, Mar. 2012.

[13] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," *RFC 6719*, Sep. 2012.

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

[15] E. Ancillotti, R. Bruno, and M. Conti, "The Role of the RPL Routing Protocol for Smart Grid Communications," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 75–83, Jan. 2013.

[16] T. Clausen, U. Herberg, and M. Philipp, "A Critical Evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2011.

[17] E. Ancillotti, R. Bruno, and M. Conti, "Reliable Data Delivery With the IETF Routing Protocol for Low-Power and Lossy Networks," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 3, pp. 1864–1877, Aug 2014.

[18] J. Ko, J. Jeong, J. Park, J. A. Jun, O. Gnawali, and J. Paek, "DualMOP-RPL: Supporting Multiple Modes of Downward Routing in a Single RPL Network," *ACM Transactions on Sensor Networks*, vol. 11, no. 2, pp. 39:1–39:20, Mar. 2015.

[19] W. Gan, Z. Shi, C. Zhang, L. Sun, and D. Ionescu, "MERPL: A more memory-efficient storing mode in RPL," in *IEEE International Conference on Networks (ICON)*, Dec. 2013.

[20] H.-S. Kim, J. Paek, and S. Bahk, "QU-RPL: Queue Utilization based RPL for Load Balancing in Large Scale Industrial Applications," in *IEEE International Conference on Sensing, Communication and Networking (SECON)*, Jun. 2015.

[21] O. Gaddour, A. Koubaa, N. Baccour, and M. Abid, "OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'14)*, May 2014.

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

[23] T. Clausen, A. C. de Verdiere, J. Yi, A. Niktash, Y. Igarashi, H. Satoh, U. Bergerg, C. Lavenu, T. Lys, and J. Dean, "The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)," *Internet Draft*, Jul. 2016.

[24] A. Oliveira and T. Vazo, "Low-power and Lossy Networks under Mobility: A Survey," *Computer Networks*, 2016.

[25] O. Gaddour and A. Koubaa, "RPL in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163 – 3178, 2012.

[26] Zhao, Ming and Kumar, Arun and Joo Chong, Peter Han and Lu, Rongxing, "A Comprehensive Study of RPL and P2P-RPL Routing Protocols: Implementation, Challenges and Opportunities," *Peer-to-Peer Networking and Applications*, pp. 1–25, Jul. 2016.

[27] Itron, "Enabling Smart Grid Applications Over a Multi-Application IPv6 Network," [Online] Available: https://www.youtube.com/watch?v=bvGpLp8cSgE (last accessed on May 2017).

[28] H.-S. Kim, H. Cho, M.-S. Lee, J. Paek, J. Ko, and S. Bahk, "MarketNet: An Asymmetric Transmission Power-based Wireless System for Managing e-Price Tags in Markets," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2015.

[29] Cisco, "Connected Grid Networks for Smart Grid - Field Area Network," [Online] Available: http://www.cisco.com/c/en/us/solutions/industries/energy/external-utilities-smart-grid/field-area-network.html (last accessed on May 2017).

[30] "RIOT OS," [Online] Available: http://riot-os.org/api/group__net__gnrc__rpl.html (last accessed on May 2017).

[31] "RIOT RPL," [Online] Available: https://github.com/RIOT-OS/RIOT/tree/master/sys/net/gnrc/routing/rpl (last accessed on May 2017).

[32] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "TRAIL: Topology Authentication in RPL," in *European Workshop on Wireless Sensor Networks (EWSN)*, 2016.

[33] C. Gündogan, C. Adjih, O. Hahm, and E. Baccelli, "Let healthy links bloom: Scalable Link Checks in Low-Power Wireless Networks for Smart Health," in *ACM International Workshop on Pervasive Wireless Healthcare (MobileHealth)*, Jul. 2016.

[34] O. Balmau, D. Dzung, A. Karaaa, V. Nesovic, A. Paunovic, Y. A. Pignolet, and N. A. Tehrani, "Evaluation of RPL for Medium Voltage Power Line Communication," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2014.

[35] Moteiv Corporation, "Tmote Sky," Was available at http://www.moteiv.com/products/tmotesky.php. No longer in production as of May 2017.

[36] M. P. Andersen, G. Fierro, and D. E. Culler, "System Design for a Synergistic, Low Power Mote/BLE Embedded Platform," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2016.

[37] N. Tsiftes, J. Eriksson, N. Finne, F. Österlind, J. Höglund, and A. Dunkels, "A Framework for Low-power IPv6 Routing Simulation, Experimentation, and Evaluation," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Sep. 2010.

[38] L. Bartolozzi, T. Pecorella, and R. Fantacci, "Ns-3 RPL Module: IPv6 Routing Protocol for Low Power and Lossy Networks," in *International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS)*, Mar. 2012.

[39] K. Iwanicki, "RNFD: Routing-Layer Detection of DODAG (Root) Node Failures in Low-Power Wireless Networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2016.

[40] 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)*, Apr. 2011.

[41] E. Ancillotti, R. Bruno, M. Conti, E. Mingozzi, and C. Vallati, "Trickle-L2: Lightweight Link Quality Estimation through Trickle in RPL Networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2014.

[42] S. Dawans, S. Duquennoy, and O. Bonaventure, "On Link Estimation in Dense RPL Deployments," in *IEEE Conference on Local Computer Networks - Workshops*, Oct. 2012.

[43] J. W. Hui and D. E. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2008.

[44] N. Khelifi, S. Oteafy, H. Hassanein, and H. Youssef, "Proactive Maintenance in RPL for 6LowPAN," in *International Conference on Wireless Communications and Mobile Computing (IWCMC)*, Aug. 2015, pp. 993–999.

[45] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the Tree Bloom: Scalable Opportunistic Routing with ORPL," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2013.

[46] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low Power, Low Delay: Opportunistic Routing Meets Duty Cycling," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2012.

[47] S. Gormus, Z. Tosato, Filippoand Fan, Z. Bocus, and P. Kulkarni, "Opportunistic RPL for Reliable AMI Mesh Networks," *Wireless Networks*, vol. 20, no. 8, pp. 2147–2164, 2014.

[48] Q. D. Ho, Y. Gao, G. Rajalingham, and T. Le-Ngoc, "Robustness of the Routing Protocol for Low-Power and Lossy Networks (RPL) in Smart Grid's Neighbor-area Networks," in *IEEE International Conference on Communications (ICC)*, Jun. 2015.

[49] J. Isern, A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "Large-Scale Performance Evaluation of the IETF Internet of Things Protocol Suite for Smart City Solutions," in *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor & Ubiquitous Networks (PE-WASUN)*, 2015.

[50] H. Kermajani and C. Gomez, "On the Network Convergence Process in RPL over IEEE 802.15.4 Multihop Networks: Improvement and Trade-Offs," *Sensors*, vol. 14, no. 7, pp. 11 993–12 022, 2014.

[51] O. Balmau, D. Dzung, and Y. A. Pignolet, "Recipes for Faster Failure Recovery in Smart Grid Communication Networks," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2014.

[52] C. Vallati and E. Mingozzi, "Trickle-F: Fair Broadcast Suppression to Improve Energy-efficient Route Formation with the RPL Routing Protocol," in *IFIP SustainIT'13*, Oct. 2013.

[53] J. Tripathi and J. C. de Oliveira, "On Adaptive Timers for Improved RPL Operation in Low-power and Lossy Sensor Networks," in *IEEE International Conference on Communication Systems and Networks (COMSNETS)*, Jan. 2013.

[54] H.-S. Kim, M.-S. Lee, Y.-J. Choi, J. Ko, and S. Bahk, "Reliable and Energy-Efficient Downward Packet Delivery in Asymmetric Transmission Power-Based Networks," *ACM Transactions on Sensor Networks*, vol. 12, no. 4, pp. 34:1–34:25, Sep. 2016.

[55] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2015.

[56] H.-S. Kim, H. Im, M.-S. Lee, J. Paek, and S. Bahk, "A Measurement Study of TCP over RPL in Low-power and Lossy Networks," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 647–655, Dec. 2015.

[57] H.-S. Kim, H. Cho, H. Kim, and S. Bahk, "DT-RPL: Diverse Bidirectional Traffic Delivery through RPL Routing Protocol in Low Power and Lossy Networks," *Computer Networks*, vol. 126, pp. 150–161, Oct. 2017.

[58] T. Istomin, C. Kiraly, and G. P. Picco, "Is RPL Ready for Actuation? A Comparative Evaluation in a Smart City Scenario," in *European Workshop on Wireless Sensor Networks (EWSN)*, Feb. 2015.

[59] C. Kiraly, T. Istomin, O. Iova, and G. P. Picco, "D-RPL: Overcoming memory limitations in RPL point-to-multipoint routing," in *IEEE Conference on Local Computer Networks (LCN)*, Oct. 2015.

[60] E. Baccelli, M. Philipp, and M. Goyal, "The P2P-RPL Routing Protocol for IPv6 Sensor Networks: Testbed Experiments," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2011.

[61] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks," *RFC 6997*, Aug. 2013.

[62] M. Goyal, E. Baccelli, A. Brandt, and J. Martocci, "A Mechanism to Measure the Routing Metrics along a Point-to-Point Route in a Low-Power and Lossy Network," *RFC 6998*, Aug. 2013.

[63] M. Zhao, I. W. H. Ho, and P. H. J. Chong, "An energy-efficient region-based rpl routing protocol for low-power and lossy networks," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1319–1333, Dec 2016.

[64] S. Jeong, H. S. Kim, S. G. Yoon, and S. Bahk, "Q-BT: Queue-Based Burst Transmission Over an Asynchronous Duty-Cycle MAC Protocol," *IEEE Communications Letters*, vol. 20, no. 4, pp. 812–815, Apr. 2016.

[65] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load Balancing under Heavy Traffic in RPL Routing Protocol for Low Power and Lossy Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 964–979, Apr. 2017.

[66] H.-S. Kim, J. Paek, D. E. Culler, and S. Bahk, "Do Not Lose Bandwidth: Adaptive Transmission Power and Multihop Topology Control," in *The 13th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'17)*, June 2017.

[67] M. Nassiri, M. Boujari, and S. V. Azhari, "Energy-aware and Load Balanced Parent Selection in RPL Routing for Wireless Sensor Networks," *International Journal of Wireless and Mobile Computing*, vol. 9, no. 3, 2015.

[68] O. Iova, F. Theoleyre, and T. Noel, "Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network," *Ad Hoc Networks*, vol. 29, pp. 45 – 62, 2015.

[69] G. Chelius, A. Fraboulet, and E. B. Hamida, "An Event-driven Simulator for Large Scale Wireless Sensor Networks," [Online] Available: http://wsnet.gforge.inria.fr (last accessed May 2017).

[70] M. Michel, S. Duquennoy, B. Quoitin, and T. Voigt, "Load-Balanced Data Collection through Opportunistic Routing," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 2015.

[71] T. B. Oliveira, P. H. Gomes, D. G. Gomes, and B. Krishnamachari, "ALABAMO: A LoAd BAlancing MOdel for RPL," in *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, Jun. 2016.

[72] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, "Load Balanced Routing for Low Power and Lossy Networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013.

[73] M. A. Lodhi, A. Rehman, M. M. Khan, and F. B. Hussain, "Multiple path rpl for low power lossy networks," in *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, Aug. 2015.

[74] F. Boubekeur, L. Blin, R. Leone, and P. Medagliani, "Bounding Degrees on RPL," in *ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, 2015.

[75] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, J.-P. Vasseur, M. Durvy, A. Terzis, A. Dunkels, and D. Culler, "Beyond Interoperability: Pushing the Performance of Sensor Network IP Stacks," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2011.

[76] L. Guan, K. Kuladinithi, T. Ptsch, and C. Goerg, "A deeper understanding of interoperability between TinyRPL and ContikiRPL," in *IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Apr. 2014.

[77] M. Stolikj, T. M. M. Meyfroyt, P. J. L. Cuijpers, and J. J. Lukkien, "Improving the Performance of Trickle-Based Data Dissemination in Low-Power Networks," in *European Workshop on Wireless Sensor Networks (EWSN)*, Feb. 2015.

[78] J. W. Hui, "An Extended Internet Architecture for Low-Power Wireless Networks - Design and Implementation," Ph.D. dissertation, EECS Department, University of California, Berkeley, Sep. 2008.

[79] C. Chauvenet, B. Tourancheau, D. Genon-Catalot, P. E. Goudet, and M. Pouillot, "A Communication Stack over PLC for Multi Physical Layer IPv6 Networking," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2010.

[80] C. Chauvenet, B. Tourancheau, and D. Genon-Catalot, "802.15.4, a MAC Layer Solution for PLC," in *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, May 2010.

[81] L. B. Saad, C. Chauvenet, and B. Tourancheau, "Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies," in *International Conference on Sensor Technologies and Applications (SENSORCOMM)*, Sep. 2011.

[82] T. Ropitault, A. Lampropulos, R. Vedantham, and P. Chiummiento, "Realistic Model for Narrowband PLC for Advanced Metering Infrastructure," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2013.

[83] T. Ropitault, A. Lampropulos, A. Pelov, L. Toutain, R. Vedantham, and P. Chiummiento, "Doing it right - Recommendations for RPL in PLC-based networks for the Smart Grid," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2014.

[84] T. Lee, M.-S. Lee, H.-S. Kim, and S. Bahk, "A Synergistic Architecture for RPL over BLE," in *IEEE International Conference on Sensing, Communication and Networking (SECON)*, Jun. 2016.

[85] J. Hui and R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)," *RFC 7311*, Feb. 2016.

[86] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.

[87] G. Oikonomou, I. Phillips, and T. Tryfonas, "IPv6 Multicast Forwarding in RPL-based Wireless Sensor Networks," *Wireless Personal Communications*, vol. 73, no. 3, pp. 1089–1116, Dec. 2013.

[88] M. A. Mahmood, W. K. G. Seah, and I. Welch, "Reliability in Wireless Sensor Networks: A Survey and Challenges Ahead," *Computer Networks*, vol. 79, pp. 166–187, Mar. 2015.

[89] K. Tharatipayakul, S. Gordon, and K. Kaemarungsi, "iACK: Implicit Acknowledgements to Improve Multicast Reliability in Wireless Sensor Networks," in *International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, May 2014.

[90] K. Q. Abdel Fadeel and K. El Sayed, "ESMRF: Enhanced Stateless Multicast RPL Forwarding For IPv6-based Low-Power and Lossy Networks," in *ACM Workshop on IoT challenges in Mobile and Industrial Systems (IoT-Sys)*, May 2015.

[91] G. G. Lorente, B. Lemmens, M. Carlier, A. Braeken, and K. Steenhaut, "BMRF: Bidirectional Multicast RPL Forwarding," *Ad Hoc Networks*, vol. 54, pp. 69 – 84, Jan. 2017.

[92] F. Fabbri, C. Buratti, and R. Verdone, "A Multi-Sink Multi-Hop Wireless Sensor Network Over a Square Region: Connectivity and Energy Consumption Issues," in *IEEE Globecom Workshops*, Nov. 2008.

[93] H. Kim, Y. Seok, N. Choi, Y. Choi, and T. Kwon, *Optimal Multi-sink Positioning and Energy-Efficient Routing in Wireless Sensor Networks*, 2005.

[94] A. Das and D. Dutta, "Data Acquisition in Multiple-sink Sensor Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 3, pp. 82–85, Jul. 2005.

[95] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The Tenet Architecture for Tiered Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, Jul. 2010.

[96] P. Kulkarni, S. Gormus, and Z. Fan, "Tree Balancing in Smart Grid Advanced Metering Infrastructure Mesh Networks," in *IEEE International Conference on Internet of Things (iThings)*, Nov. 2012.

[97] M. Ha, K. Kwon, D. Kim, and P. Y. Kong, "Dynamic and Distributed Load Balancing Scheme in Multi-gateway Based 6LoWPAN," in *IEEE International Conference on Internet of Things (iThings)*, Sep. 2014.

[98] P. Thulasiraman, "RPL routing for multigateway AMI networks under interference constraints," in *IEEE International Conference on Communications (ICC)*, Jun. 2013.

[99] P. Kulkarni, S. Gormus, Z. Fan, and F. Ramos, "AMI Mesh Networks: A Practical Solution and Its Performance Evaluation," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1469–1481, Sep. 2012.

[100] P. Kulkarni, S. Gormus, Z. Fan, and B. Motz, "A Mesh-radio-based Solution for Smart Metering Networks," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 86–95, Jul. 2012.

[101] ZigBee Alliance, "ZigBee Specification," 2006. [Online]. Available: http://www.zigbee.org

[102] L. Deru, S. Dawans, M. Ocaña, B. Quoitin, and O. Bonaventure, "Redundant Border Routers for Mission-critical 6LoWPAN Networks," in *International Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Dec. 2013.

[103] K. Andrea and R. Simon, "Design and Evaluation of an RPL-based Multi-Sink Routing Protocol for Low-Power and Lossy Networks," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2015.

[104] G. Rajalingham, Y. Gao, Q.-D. Ho, and T. Le-Ngoc, "Quality of Service Differentiation for Smart Grid Neighbor Area Networks Through Multiple RPL Instances," in *ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, 2014.

[105] M. Banh, H. Mac, N. Nguyen, K. H. Phung, N. H. Thanh, and K. Steenhaut, "Performance Evaluation of Multiple RPL Routing Tree Instances for Internet of Things Applications," in *International*

Conference on Advanced Technologies for Communications (ATC), Oct. 2015.

[106] M. Barcelo, A. Correa, J. L. Vicario, and A. Morell, "Cooperative Interaction among Multiple RPL Instances in Wireless Sensor Networks," *Computer Communications*, vol. 81, pp. 61 – 71, 2016.

[107] D. Han and O. Gnawali, "Performance of rpl under wireless interference," *IEEE Communications Magazine*, vol. 51, no. 12, pp. 137–143, Dec. 2013.

[108] M. Mohammad, X. Guo, and M. C. Chan, "Oppcast: Exploiting spatial and channel diversity for robust data collection in urban environments," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2016.

[109] W. Sun, J. Paek, and S. Choi, "CV-Track: Leveraging Carrier Frequency Offset Variation for BLE Signal Detection," in *The 4th ACM Workshop on Hot Topics in Wireless (HotWireless'17)*, Oct. 2017.

[110] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L. . Nordn, and P. Gunningberg, "SoNIC: Classifying interference in 802.15.4 sensor networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2013.

[111] A. Hithnawi, H. Shafagh, and S. Duquennoy, "TIIM: Technology-independent Interference Mitigation for Low-power Wireless Networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2015.

[112] M. Sha, G. Hackmann, and C. Lu, "Energy-efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.

[113] X. Zheng, Z. Cao, J. Wang, Y. He, and Y. Liu, "ZiSense: Towards Interference Resilient Duty Cycling in Wireless Sensor Networks," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2014.

[114] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-fi Interference in Low Power ZigBee Networks," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[115] T. Mandel and J. Mache, "Practical Error Correction for Resource-constrained Wireless Networks: Unlocking the Full Power of the CRC," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2013.

[116] W. Dong, J. Yu, and X. Liu, "CARE: Corruption-Aware Retransmission with Adaptive Coding for the Low-Power Wireless," in *IEEE International Conference on Network Protocols (ICNP)*, Nov. 2015.

[117] A. Hithnawi, S. Li, H. Shafagh, J. Gross, and S. Duquennoy, "CrossZig: Combating Cross-Technology Interference in Low-Power Wireless Networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2016.

[118] J. Heo, J. J. Kim, J. Paek, and S. Bahk, "Dodge-Jam: Anti-Jamming Technique for Low-power and Lossy Wireless Networks," in *The 14th IEEE International Conference on Sensing, Communication and Networking (SECON'17)*, June 2017.

[119] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," *RFC 6282*, Sep. 2011.

[120] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1947–1960, Nov. 2010.

[121] J. Park, W. Nam, T. Kim, J. Choi, S. Lee, D. Yoon, J. Paek, and J. Ko, "Glasses for the Third Eye: Improving Clinical Data Analysis with Motion Sensor-based Filtering," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2017.

[122] P. Park, C. Fischione, A. Bonivento, K. H. Johansson, and A. Sangiovanni-Vincent, "Breath: An adaptive protocol for industrial control applications using wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 6, pp. 821–838, Jun. 2011.

[123] I. E. Korbi, M. B. Brahim, C. Adjih, and L. A. Saidane, "Mobility Enhanced RPL for Wireless Sensor Networks," in *3rd International Conference on the Network of the Future (NOF)*, Nov. 2012.

[124] C. Cobrzan, J. Montavont, and T. Nol, "Analysis and Performance Evaluation of RPL under Mobility," in *IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2014.

[125] K. C. Lee, R. Sudhaakar, J. Ning, L. Dai, S. Addepalli, J. P. Vasseur, and M. Gerla, "A Comprehensive Evaluation of RPL under Mobility," *International Journal of Vehicular Technology*, vol. 2012, 2012.

[126] J. Ko and M. Chang, "MoMoRo: Providing Mobility Support for Low-Power Wireless Applications," *IEEE Systems Journal*, vol. 9, no. 2, pp. 585–594, Jun. 2015.

[127] H. Fotouhi, D. Moreira, and M. Alves, "mRPL: Boosting Mobility in the Internet of Things," *Ad Hoc Networks*, vol. 26, pp. 17 – 35, 2015.

[128] H. Fotouhi, "Reliable Mobility Support in Low-Power Wireless Networks," *PhD Thesis at Polytechnic Institue of Porto*, 2015.

[129] O. Gaddour, A. Kouba, R. Rangarajan, O. Cheikhrouhou, E. Tovar, and M. Abid, "Co-RPL: RPL routing for mobile low power wireless sensor networks using Corona mechanism," in *IEEE International Symposium on Industrial Embedded Systems (SIES)*, Jun. 2014.

[130] M. Barcelo, A. Correa, J. L. Vicario, A. Morell, and X. Vilajosana, "Addressing Mobility in RPL With Position Assisted Metrics," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2151–2161, Apr. 2016.

[131] D. Carels, E. D. Poorter, I. Moerman, and P. Demeester, "RPL Mobility Support for Point-to-point Traffic Flows Towards Mobile Nodes," *Int. J. Distrib. Sen. Netw.*, vol. 2015, Jan. 2015.

[132] C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On-demand Distance Vector (AODV) routing," *RFC 3561*, Jul. 2003.

[133] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[134] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, 2001.

[135] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A Framework to Systematically Analyze the Impact of Mobility on Performance of RouTing protocols for Ad hoc NeTworks," in *Proceedings of the 22nd International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[136] M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2001.

[137] K.-W. Chin, J. Judge, A. Williams, and R. Kermode, "Implementation Experience with MANET Routing Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 5, pp. 49–59, 2002.

[138] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions," in *International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2004.

[139] K.-W. Chin, "The Behavior of MANET Routing Protocols in Realistic Environments," in *IEEE Asia-Pacific Conference on Communications (APCC)*, 2005.

[140] U. Herberg and T. Clausen, "A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-directional Traffic in Low-power and Lossy Networks," in *ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN)*, 2011.

[141] M. Vuini, B. Tourancheau, and A. Duda, "Performance Comparison of the RPL and LOADng Routing Protocols in a Home Automation Scenario," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013.

[142] J. Tripathi and J. C. de Oliveira, "Proactive versus Reactive Revisited: IPv6 Routing for Low Power Lossy Networks," in *47th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2013.

[143] S. Elyengui, R. Bouhouchi, and T. Ezzedine, "LOADng Routing Protocol Evaluation for Bidirectional Data flow in AMI Mesh Networks," *CoRR*, vol. abs/1506.06357, 2015.

[144] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, "The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3685–3692, Oct. 2013.

[145] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A Study of RPL DODAG Version Attacks," in *International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, Jun. 2014.

[146] A. Rghioui, A. Khannous, and M. Bouhorma, "Denial-of-Service Attacks on 6LoWPAN-RPL Networks: Threats and an Intrusion Detection System Proposition," *Journal of Advanced Computer Science and Technology*, vol. 3, no. 2, 2014.

[147] F. Medjek, D. Tandjaoui, M. R. Abdmeziem, and N. Djedjig, "Analytical Evaluation of the Impacts of Sybil Attacks against RPL under Mobility," in *International Symposium on Programming and Systems (ISPS)*, Apr. 2015.

[148] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459 – 473, May 2016.

[149] A. Dvir, T. Holczer, and L. Buttyan, "VeRA - Version Number and Rank Authentication in RPL," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Oct. 2011.

[150] K. Weekly and K. Pister, "Evaluating Sinkhole Defense Techniques in RPL Networks," in *IEEE International Conference on Network Protocols (ICNP)*, Oct. 2012.
[151] F. I. Khan, T. Shon, T. Lee, and K. Kim, "Wormhole Attack Prevention Mechanism for RPL based LLN Network," in *International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2013.
[152] L. Ye, V. Fodor, T. Giannetsos, and P. Papadimitratos, "Path Metric Authentication for Low-Power and Lossy Networks," in *ACM International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater)*, 2015.
[153] L. Wallgrem, S. Raza, and T. Voigt, "Routing Attacks and Countermeasures in the RPL-based Internet of Things," *International Journal of Distributed Sensor Networks*, 2013.
[154] M. Landsmann, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "Demo: Topological Robustness of RPL with TRAIL," in *European Workshop on Wireless Sensor Networks (EWSN)*, 2016.
[155] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schnwlder, "Mitigation of Topological Inconsistency Attacks in RPL-based Low-power Lossy Networks," *International Journal of Network Management*, vol. 25, no. 5, pp. 320–339, 2015.
[156] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," *RFC 2119*, Mar. 1997.
[157] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *RFC 2460*, Dec. 1998.
[158] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," *RFC 4728*, Feb. 2007.
[159] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6," *RFC 5340*, Jul. 2008.
[160] G. Malkin, "RIP Version 2," *RFC 2453*, Nov. 1998.
[161] A. Parasuram, "An Analysis of the RPL Routing Standard for Low Power and Lossy Networks," Master's thesis, EECS Department, University of California, Berkeley, May 2016.
[162] Google Nest, "OpenThread," [Online] Available: https://github.com/openthread/openthread, 2016.
[163] J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Bruenig, T. Schmid, Q. Wang, P. Dutta, and A. Terzis, "Low Power or High Performance? A Tradeoff Whose Time Has Come (and Nearly Gone)," in *European Workshop on Wireless Sensor Networks (EWSN)*, 2012.
[164] S. Duquennoy, O. Landsiedel, C. A. Boano, M. Zimmerling, J. Beutel, M. C. Chan, O. Gnawali, M. Mohammad, L. Mottola, L. Thiele, X. Vilajosana, T. Voigt, and T. Watteyne, "A Benchmark for Low-power Wireless Networking: Poster Abstract," in *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2016.

Hyung-Sin Kim received his B.S. degree in Electrical Engineering from Seoul National University (SNU), Seoul, Korea in 2009. He received his M.S. degree in 2011 and his Ph.D. degree in 2016, all in Electrical Engineering and Computer Science (EECS) from SNU, all with outstanding thesis awards. From March 2016 to August 2016, he was a postdoctoral researcher at Network Laboratory (NETLAB) at SNU led by Prof. Saewoong Bahk. He is currently a postdoctoral researcher of EECS at the University of California at Berkeley and working at Building Energy Transportation Systems (BETS) group led by Prof. David E. Culler. He received the Qualcomm Korea Fellowship in 2011, and the National Research Foundation (NRF) Global Ph.D. Fellowship and Postdoctoral Fellowship in 2011 and 2016, respectively. His research interests include development of Internet of Things (IoT) systems for urban marketplaces and smart buildings, and of network and communication protocols for low power wireless systems.

JeongGil Ko received his Bachelors in Engineering degree in computer science and engineering from Korea University in 2007. He received his Master of Science in Engineering and Doctor of Philosophy degrees in Computer Science from the Johns Hopkins University in 2009 and 2012, respectively. At Johns Hopkins, JeongGil Ko was a member of the Hopkins interNetworking Research Group (HiNRG) led by Dr. Andreas Terzis. In 2010, he was at the Stanford Information Networking Group with Dr. Philip Levis at Stanford University as a visiting researcher. From June 2012 to August 2015, JeongGil Ko was a senior researcher at the Electronics and Telecommunications Research Institute. He is a recipient of the Abel Wolman Fellowship awarded by the Whiting School of Engineering at the Johns Hopkins University in 2007. His research interests are in the general area of developing web and cloud-based sensing systems with ambient intelligence for the Internet of Things and Cyber Physical Systems. JeongGil Ko is currently with the Department of Software and Computer Engineering, College of Information Technology and also jointly with the Department of Biomedical Informatics, School of Medicine at Ajou University as an assistant professor and leads the Ajou Embedded Intelligent Systems Laboratory.

David E. Culler (Fellow, IEEE) received the B.A. degree in mathematics from the University of California at Berkeley, Berkeley, in 1980 and the M.S. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1985 and 1989, respectively. He is a Professor of Electrical Engineering and Computer Sciences at the University of California at Berkeley. He has been on the faculty at Berkeley since 1989, where he holds the Howard Friesen Chair. Prof. Culler is a member of the National Academy of Engineering and a Fellow of the Association for Computing Machinery (ACM). He was selected for ACM's Sigmod Outstanding Achievement Award, Scientific American's Top 50 Researchers, and Technology Review's 10 Technologies that Will Change the World. He received the National Science Foundation (NSF) Presidential Young Investigators NSF Presidential Faculty Fellowship in 1992. He was co-founder and CTO of Arch Rock Corporation and serves on several corporate technical advisory boards.

Jeongyeup Paek received his B.S. degree from Seoul National University in 2003 and his M.S. degree from University of Southern California in 2005, both in Electrical Engineering. He then received his Ph.D. degree in Computer Science from the University of Southern California (USC) in 2010 where he was a member of the Networked Systems Laboratory (NSL) led by Dr. Ramesh Govindan. He worked at Deutsche Telekom Inc. R&D Labs USA as a research intern in 2010, and then joined Cisco Systems Inc. in 2011 where he was a Technical Leader in the Internet of Things Group, Connected Energy Networks Business Unit (formerly the Smart Grid Business Unit). In 2014, he was with the Hongik University, Department of Computer Information Communication as an assistant professor. Jeongyeup Paek is currently an assistant professor at Chung-Ang University, School of Computer Science and Engineering, Seoul, Republic of Korea.