RPL, the Routing Standard for the Internet of Things … Or Is It?

Oana Iova, Gian Pietro Picco, Timofei Istomin, and Csaba Kiraly

***Abstract*—RPL, the IPv6 Routing Protocol for Low-Power and Lossy Networks, is considered the *de facto* routing protocol for the Internet of Things (IoT). Since its standardization, RPL contributed to the advancement of communications in the world of tiny, embedded, networking devices, by providing, along with other standards, a baseline architecture for IoT. Several years later, we analyze the extent to which RPL lived up to the expectations defined by the IETF requirements, and tie our analysis to current trends, identifying the challenges RPL must face to remain on the forefront of IoT technology.**

***Index Terms*—RPL, Internet of Things, routing, standards**

I. Introduction

RPL, the IPv6 Routing Protocol for Low-Power and Lossy Networks [1], was standardized by the IETF in 2011 to establish a common ground for building interoperable commercial appliances in growing markets enabled by low-power and lossy networks (LLNs). Meanwhile, the Internet of Things (IoT) emerged, envisioning ubiquitous and global connectivity among the billions objects that we use in the everyday life.

Given the significant overlapping between LLNs and IoT, and the fact that IPv6 is an essential feature for the latter, RPL has rapidly become *the* routing protocol for IoT, incorporating the protocol stack defined by IETF in this scope, atop the IEEE 802.15.4 MAC and PHY layers (Figure 1a). The success of RPL as an IoT standard is also witnessed by the fact that the companies part of the ZigBee Alliance have adopted RPL as their underlying technology (Figure 1b). We provide a concise overview of RPL in Section II.

Nevertheless, the design goals behind RPL date back to the routing requirements determined by IETF in 2009 by considering the applications domains of home and building automation [2], [3] and of urban and industrial networks [4], [5]. After several years, it is important to ascertain the extent to which these requirements have been fulfilled by RPL; we offer our analysis of the state of the art in Section III.

A similarly important question, however, is whether RPL is well-prepared to sustain today's rapidly-evolving field of IoT, which defines slightly different scenarios from those examined by IETF in 2009. In Section IV, we identify a few specific challenges that RPL must face to remain on the forefront of IoT technology. Finally, Section V contains concluding remarks concerning the future of RPL.

Manuscript received September 27, 2016.

Oana Iova, Timofei Istomin, and Gian Pietro Picco are with University of Trento, Italy (email: {oanateodora.iova, timofei.istomin, gianpietro.picco}@unitn.it. Csaba Kiraly is with Bruno Kessler Foundation, Italy (email: kiraly@fbk.eu).

II. RPL in a Nutshell

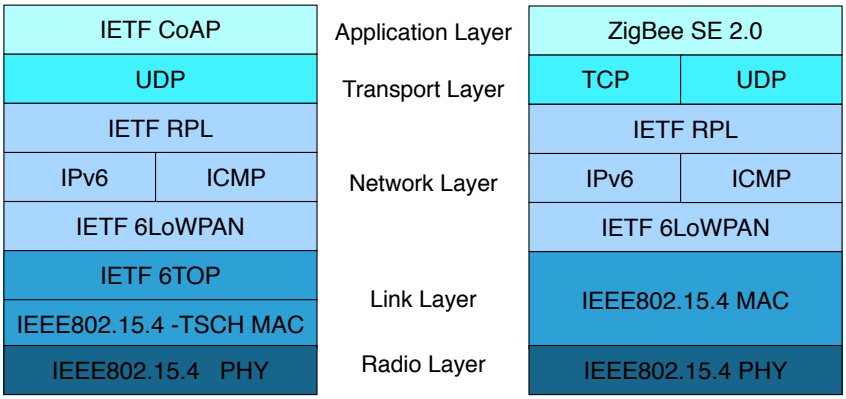
As described in the standard [1], RPL creates a routing topology in the form of a Destination-Oriented Directed Acyclic Graph (DODAG): a directed graph without cycles, oriented towards a *root* node, e.g., a border router. By default, each node maintains multiple parents towards the root; a *preferred* one is used for actually forwarding data packets upwards towards the root (Figure 2b), while the others are kept as backup routes. This scheme, called *multipoint-to-point communication* in RPL, naturally supports communication from the devices to the root with minimal routing state.

The topology is created and maintained via control packets called DODAG Information Objects (DIO), advertised by each node. The packet contains the routing metric (e.g., link quality, residual energy) and an objective function used by each node to select the parents among its neighbors. DIO packets are rebroadcast by each node according to an adaptive technique, the Trickle algorithm [6], which strikes a tradeoff between reactivity to topology changes and energy efficiency. Trickle ensures that DIOs are advertised aggressively when the network is unstable, and instead rebroadcast at an increasingly slow pace while the network is stable.

To support the dual traffic pattern from the root to the devices, called *point-to-multipoint communication* in RPL, the standard requires additional control messages and routing state. Specifically, each node in the network must announce itself as a possible destination to the root by sending Destination Advertisement Object (DAO) control packets. These messages are propagated "upwards" in the DODAG topology, via a parent that may coincide with the preferred parent (Figure 2c), therefore establishing "downwards" routes along the way.

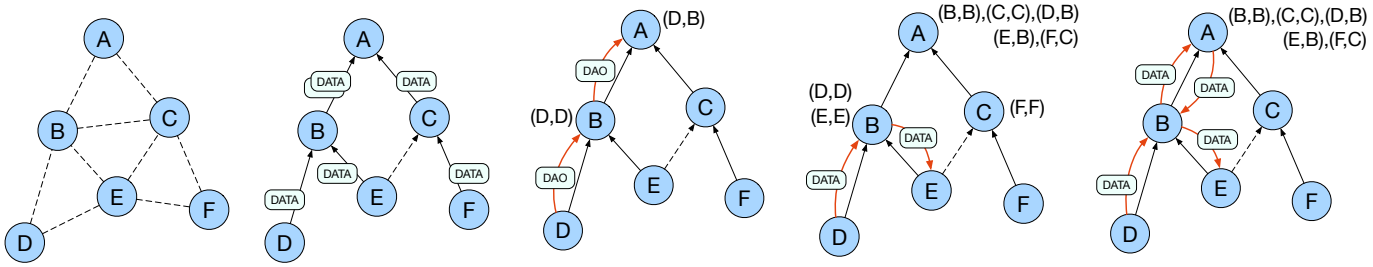
As supporting this type of traffic could put more strain on the nodes memory due to the increased routing state, RPL defines two modes of operation: *storing* and *non-storing*. In storing mode, each node keeps a routing table containing mappings between all destinations reachable via its sub-DODAG and their respective next-hop nodes, learned while receiving DAOs. In non-storing mode, the root is the only network node maintaining routing information; the root exploits this global view for source routing, i.e., by including routing information directly into the packet itself.

These two modes of operation affect also the ability to support communication between any two nodes in the network, called *point-to-point communication* in RPL, and achieved as a combination of the techniques used for the previous two types of traffic. In storing mode, data packets travel upwards until they reach a node with routing information about



(a) IETF standardized stack. (b) ZigbeeIP stack.

Fig. 1: Protocol stacks for the Internet of Things.



(a) A sample wireless network. (b) Multipoint-to-point communication. (c) Point-to-multipoint route construction: storing mode. (d) Point-to-point communication: storing mode. (e) Point-to-point communication: non storing mode.

Fig. 2: Routing in RPL. Existing routes are shown next to the network nodes.

their destination, i.e., a common ancestor (Figure 2d); from that point on, they proceed downwards, following the routes previously established by DAO packets. The same technique is used in non-storing mode; however, in this case, packets must travel upwards all the way to the root (the only node maintaining routing information) before being redirected to their destination (Figure 2e).

III. Did RPL Live Up to Expectations?

RPL was standardized based on requirements published 7 years ago[1], a relatively long time in the fast-paced world of networked computing in general and IoT in particular. This section revisits the original requirements for RPL stated by IETF, and concisely reports about the extent to which they are met by the state of the art. We frequently refer to the IETF documents that focused on the paradigmatic applications of home [2] and building automation [3], and urban [4] and industrial networks [5]. Due to space limitations, we focus on what we argue are the key dimensions, therefore omitting considerations on other relevant aspects, e.g., including zero-configuration or security.

A. Traffic pattern

Original requirements. RPL was designed to account for different traffic patterns characteristic of the aforementioned reference applications. Multipoint-to-point communication is required by devices with sensing capabilities, which typically monitor the environment by periodically acquiring samples of physical quantities (e.g., temperature, pollution, humidity) and send them to a central unit. Applications may also need the dual point-to-multipoint pattern where communication is directed from the central unit to the rest of the network. This is often required to send queries to sensors or, when a control loop is present, to send actuation commands—e.g., switching lights or activating window blinds in a smart home [2]. This point-to-multipoint pattern may rely on multicast routing, which is actually a requirement in some reference scenarios [3], [4]. Finally, in alternative to a centralized controller gathering data and issuing commands, devices might cooperate with each other in a decentralized fashion by relying on point-to-point communication; for instance, lamps and appliances may automatically change their own state or the state of other appliances based on information gathered by sensor nodes nearby.

> The routing protocol should support communication: *i)* from devices to a central unit (multipoint-to-point) *ii)* from the central unit to the rest of the network (point-to-multipoint) *iii)* directly among devices (point-to-point).

Current status. As we discussed in Section II, all three types of traffic patterns above are supported by the RPL standard, although it emphasizes multipoint-to-point communication, naturally supported by the DODAG routing topology. On the contrary, point-to-point communication is inherently costly, as shown in Figures 2d–2e; if two nodes want to communicate with each other, packets must be sent upwards either to the first common ancestor (in storing mode), or all the way up to the root (in non-storing mode), from where they are re-routed downwards to their destination. This strategy creates congestion close to the root and greatly increases overhead and latency, causing problems especially in use cases involving replies, e.g., a query result or a command outcome. In RPL, this is the price to pay for reconciling different patterns in a single protocol, yielding the benefit of broader applicability at the expense of optimal performance.

These shortcomings motivated a new IETF standard protocol, *Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks (P2P-RPL)*, described in RFC 6697. This protocol is meant to complement RPL by allowing nodes to discover *on demand* routes to one or more nodes, via a temporary DODAG rooted at the node initiating communication. However, the efficiency of this combined solution is yet to be thoroughly studied.

As for multicast routing in point-to-multipoint communication, RPL supports it only in storing mode. Moreover, the standard only briefly describes how DAO messages should be used for group registration, without providing a full description of multicast operation. However, a complementary RFC 7731, *Multicast Protocol for Low-Power and Lossy Networks (MPL)*, has been recently standardized, defining multicast operation atop two protocols: Trickle and plain flooding. Evaluations in simulation (e.g., [7]) show that MPL is highly reliable, but can have high delay and energy consumption if Trickle parameters are not chosen carefully.

> RPL supports all types of communication. However, for an efficient point-to-point communication and multicast routing, complementary RFCs must be implemented, increasing code complexity and memory footprint.

B. Mobility

Original requirements. In all the motivating scenarios analyzed by IETF it is foreseen that, although devices are currently mostly fixed, a variety of devices with reduced mobility (e.g., remote controls, vacuum cleaner robots, wearable healthcare devices) should be accommodated by RPL. Moreover, it is stated that industrial applications require the support of nodes located on vehicles or machines moving at speeds up to 35 km/h [5]. Devices should not act as routers while in motion; on the other hand, they should reestablish end-to-end communication with a static device within 5 s [3].

> The routing protocol should allow devices that are moving to connect to the static routing topology.

Current status. The requirement above is fulfilled by RPL by allowing a mobile node to attach as a leaf to any node belonging to the routing topology. Communication from the mobile node to the root is in principle quite straightforward, as the former only requires a preferred parent, which may as well be the point of attachment. Point-to-multipoint communication, instead, is more complex; as the mobile node moves from a parent to another, information disseminated via DAOs may rapidly become obsolete.

Solutions for this case are not specified in the standard nor, to the best of our knowledge, realized in publicly-available reference implementations. Moreover, the under-specification in the standard is a problem also for multipoint-to-point; recent studies (e.g., [8]) show that currently implemented RPL mechanisms fail to rapidly detect when the preferred parent of a mobile node becomes unreachable as it moves, leading to high packet loss.

> RPL allows a mobile node to attach/detach to/from the routing topology, but communication becomes highly unreliable.

C. Resource Heterogeneity

Original requirements. The devices targeted by RPL range from low-end battery-powered embedded devices severely constrained in memory and processing capabilities, up to high-end devices like smart watches or smartphones. This is in line with the reference scenarios; for example, home automation devices range from dumb, resource-constrained smoke alarms and temperature sensors to high-resource surveillance cameras. This diversity in hardware capabilities and application scenarios should be accounted for at the network layer. On the other hand, resource-constrained devices are envisioned to constitute the majority of network nodes; the need to limit battery replacement sets a lifetime goal of several years.

> The routing protocol must take into account the heterogeneity of devices, and specifically address the memory and energy requirements of resource-constrained devices.

Current status. As mentioned in Section II, RPL defines two modes—storing and non-storing—meant to address different resource capabilities of the network nodes. However, the RPL standard does *not* allow these two modes of operation to be mixed in the same network, severely undermining their practical use. Research solutions allowing nodes with different modes to operate in the same RPL network exist (e.g., [9]) but are yet to be standardized.

Even if this problem were to be solved, however, another threat to memory-constrained devices comes from the protocol footprint. RPL inherits the interoperability benefits of IPv6, but also its complexity; for instance, the need to construct and parse IPv6 packets with floating header options and using 6LoWPAN address compression increases significantly code memory needs. This is evident in Table I, which compares the RPL implementations for TinyOS and Contiki, the most popular operating systems for low-power devices, with some well-known wireless sensor network protocols. The table shows

Protocol name	Contiki	TinyOS
Collection tree	25.2	17.5
Trickle	21.2	14.2
Collection tree + Trickle	26.0	23.0
AODV	23.4	—
RPL	42.9	32.8
RPL + MPL	46.4	—
P2P-RPL	44.2	—

TABLE I: Code memory requirements (OS included), in kB.

that RPL requires almost twice the memory than a collection tree protocol [10]; the latter provides only multipoint-to-point communication, but it can be combined with the Trickle dissemination protocol (providing point-to-multipoint communication) by only marginally increasing the footprint, remaining significantly smaller than in RPL. On the popular TMote Sky platform, the footprint of RPL and of the variants discussed in Section III-A is very close to the 48 kB limit for code memory, leaving almost no memory to the application—despite the fact that these RPL implementations are far from supporting all the features specified in the RPL standard.

Code memory is not even the most severe problem; for example, in the Contiki implementation the 10 kB RAM of a TMote Sky limits the neighbor and routing tables to 20 and 50 entries, respectively, severely limiting scalability as discussed next.

> RPL has too large of a footprint for resource-constrained devices, and requires all devices in a network to run the same mode of operation, limiting heterogeneity.

D. Scalability

Original requirements. Scalability is a very important characteristic of a routing protocol, bearing a direct impact on network reliability and performance. The applications studied by IETF have different scalability requirements, depending on the area where networks are deployed. For example, in home automation the routing protocol must support at least 250 devices, with larger numbers envisioned in the future [2]. Other scenarios imply even higher numbers of deployed devices; in smart cities, networks of $10^2 - 10^7$ nodes are envisioned, possibly organized into regions containing $10^2 - 10^4$ devices each [4]. While these numbers are very large, and unprecedented in the roll-outs prior to the issuing of the RPL standard, they align with deployment experiences[1] and projected trends[2].

> The routing protocol should support very large networks, possibly organized in sub-networks up to thousands devices.

Current status. Recent real-world evaluations [11] show that the scalability of multipoint-to-point communication is quite good. It does, however, become questionable when unicast point-to-multipoint communication is used. In non-storing mode, the limiting factor is the size of the packet header containing the source routing information; this can include up to 8 IPv6 addresses (64 with a compressed header), but the longer the header the higher the overhead and the route repair latency. The impact on scalability, however, has yet to be evaluated; non-storing mode has been introduced only very recently in Contiki, and is currently not supported by TinyOS or industrial implementations (e.g., from Cisco[3]).

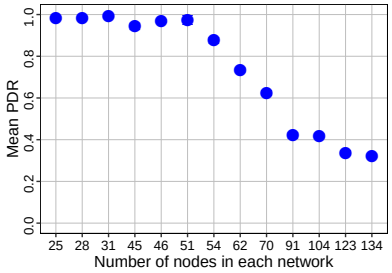


Fig. 3: As network size increases, the reliability of downward traffic decreases, as a consequence of more nodes having memory problems.

In storing mode, instead, the limiting factor is the memory available to store neighbor and routing tables. The nodes close to the root must store routing state for almost the entire DODAG, which can be challenging for resource-constrained devices. Moreover, the RPL standard does not specify the action to take after a parent refuses to install a new downward route (e.g., because its routing table is full), therefore undermining the reliability of downward traffic. We recently analyzed the problem in our own work, where we focused on emulation of the Contiki RPL implementation over realistic topologies taken from a smart city deployment [12]. Figure 3 illustrates the problem: the reliability of downward traffic (e.g., for actuation) decreases severely as soon as the size of the network approaches the size of the routing table, which we set to 50 entries—the maximum allowed on TMote Sky, as already discussed. In practice, even for devices slightly more powerful than motes, this means that the scalability of RPL remains a far cry from the thousands of nodes mentioned above.

Until now we considered a flat address space. However, RPL can support prefix-based hierarchical routing, e.g., via the Prefix Information Option (PIO) mechanism, which allows routers to own an independent prefix and distribute it for autoconfiguration in their own sub-DODAG. However, the prefix assignment is not part of the standard, and can lead to performance deterioration [13].

> RPL has serious scalability issues with point-to-multipoint traffic, especially when configured in storing mode. Prefix-based sub-netting can alleviate the problem, but requires better integration with RPL

[1] For instance, e.g., http://www.smartsantander.eu

[2] An example are the recent statements by several companies about tens of billions of devices connected by the IoT in the near future.

[3] RPL configuration guide for Cisco IOS, http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/rpl/configuration/15-mt/rpl-15-mt-book.pdf.

topology maintenance.

E. Reliability and Robustness

Original requirements. The quality of radio links among devices can be very unstable, especially in an urban setting, due to wireless channel effects (path loss, shadowing, fading), and interference from other communication systems. It is important for the routing protocol to react to changes in connectivity by rapidly reconfiguring the topology while maintaining a low control overhead and therefore energy expenditure. Specific metrics are stated for some of the reference scenarios considered; for instance, in home automation, the routing protocol is expected to converge within 0.5 s if no nodes have moved, and within 4 s otherwise [2].

Clearly, changes in connectivity ultimately affect communication reliability, whose critical role depends on the target application and functionality. For instance, it is expected that monitoring-only applications can tolerate a reliability as low as 90% [5], while those involving actuation require higher reliability [3]; in some application domains, message loss is even considered out of compliance and subject to fines [5], requiring a reliability very close to 100%.

The routing protocol should be robust w.r.t. the inherent dynamicity of wireless links, quickly restoring communication to maintain high levels of reliability; some applications cannot afford data loss.

Current status. RPL relies on the Trickle algorithm [6] to efficiently maintain the routing topology, enabling quick reaction to connectivity changes while minimizing overhead during stable conditions. Nevertheless, when a link quality metric is used for routing, even small variations in link quality estimation can yield changes of preferred parent, as the objective function constantly searches for the best path towards the root; the consequent reaction of the Trickle algorithm generates unnecessary overhead.

The IETF addressed the problem in RFC 6719, *Minimum Rank with Hysteresis Objective Function (MRHOF)*, allowing a node to change its parent only when the new path towards the root differs significantly from the old one. The latter aspect is defined by a threshold, yielding a tradeoff between route stability and optimality; if the threshold is too high, parents are less likely to change and routes are more stable, but their quality may degrade significantly before they are reconfigured. The proper configuration of the threshold is therefore non-trivial, and not well investigated in the literature.

On the other hand, the overall reliability of RPL has been analyzed by several studies in real-world testbeds. A large 135-node experiment [11] showed values of 97% and 92% of delivered packets for the multipoint-to-point and point-to-multipoint traffic respectively, while the most challenging point-to-point traffic showed a very low reliability of 74%, due to scalability problems. For the last two traffic patterns, the authors even had to reduce by half the number of addressable nodes in the network for the routing tables to fit into the RAM, due to the aforementioned memory problems.

For multipoint-to-point traffic, RPL mechanisms deal effectively with the vagaries of wireless communication and yield good performance and reliability; the other two traffic patterns suffer from poor reliability.

IV. RPL for IoT: The Challenges Ahead

Having examined how RPL fares w.r.t. its original requirements, we now turn our attention to current IoT trends and identify new requirements that may challenge future adoption of RPL as the routing protocol for IoT.

A. What Will Be the Dominant Traffic Pattern?

At the time of RPL standardization, multipoint-to-point communication was the main traffic pattern, motivated by the need to support distributed monitoring applications, and fueled by a decade of research on wireless sensor networks.

Today, *monitoring* is still a fundamental element of IoT architectures, but it is increasingly associated with *control* in a plethora of application domains. In its simplest incarnation, control entails the ability to send commands (e.g., for actuation) from a centralized controller via the same network used for monitoring; however, the most disruptive scenarios envision devices that interact and automatically take actions via in-network feedback loops. As a consequence, point-to-multipoint and point-to-point communication become even more critical than multipoint-to-point.

These two traffic patterns are addressed by RPL, which actually has a competitive advantage as it supports the networking requirements of both monitoring and control *i)* in an integrated fashion, and *ii)* in a large-scale setting. Unfortunately, as discussed in Section III, these are exactly the traffic patterns for which RPL does *not* provide efficient solutions. Paradoxically, they are also the paradigms for which the integration with IP is key, as it enables direct addressing of each device.

Point-to-multipoint and point-to-point communication received significantly less attention in RPL, yielding implementations with poor performance; this may prevent future adoption of RPL in the ever-increasing IoT applications with a control component.

B. Mobile Devices: Norm or Exception?

Scenarios for IoT assume devices embedded in things, which can be relocated (e.g., in asset tracking), or carried and even worn by people, and therefore inherently mobile. In these ever-increasing mobile scenarios, the ability to opportunistically exploit the presence of a networking "backbone" to relay information regardless of the current location is fundamental, yet is largely overlooked by RPL, as discussed in Section III-B.

Moreover, mobility often implies the need for context-aware interactions; for instance, the scope of a query or command issued by a node may need to be restricted to only a portion of the network surrounding it, based on system or application level properties (e.g., the floor building where the person carrying the device is walking).

Mobility is an unavoidable requirement in IoT scenarios, for which RPL currently provides unsatisfactory performance; further developments should also consider support for context-aware routing.

C. Resource-constrained Devices: Norm or Exception?

The devices enabling the IoT vision are inherently resource-constrained. This is unlikely to change in the future; technological developments will push the boundary of computing and communication power for the devices we know today, but at the same time generate new breeds of devices even smaller and therefore resource-constrained—a recurring trend in the last decades.

In this respect, the current demands of RPL in terms of code and data memory are still too high. In part, this is a consequence of IPv6 compliance. However, it is also the result of a standard that simply addresses (and requires) too much and, ironically, of implementations that neglect the few opportunities the standard allows to explicitly cater for resource-constrained devices—e.g., notably including non-storing mode. Specialized RPL variants must be devised, guaranteeing a better tradeoff between performance and flexibility and allowing the selection of the appropriate protocol components or variants based on the device (and application) constraints at hand.

Resource-constrained devices are here to stay. However, the current definition of RPL is often at odds with the requirements they pose, and the implementations are often ignoring the few provisions the standard already offers to ameliorate the situation.

D. New Approaches to Network Stack Design

When the RPL standard was in the making, the predominant routing approach was incarnated by the CTP protocol [10], from which RPL borrows several techniques. Around the time the standard was issued, however, two alternative routing paradigms emerged in the closely-related wireless sensor network research community: opportunistic routing and synchronous transmissions. The former approach, popularized by ORW [14], considers all neighbors as potential forwarders, therefore reducing delay and energy consumption and increasing robustness by exploiting spatial diversity. The second approach, popularized by Glossy [15], removes completely the need for a routing (and MAC) layer, by providing a reliable network flooding primitive; reliability is achieved by guaranteeing that rebroadcasting occurs within a very short time bound necessary to exploit constructive interference and the capture effect, and yielding also network-wide time synchronization.

Both approaches are reported to significantly improve over the state of the art, in terms of reliability, latency, and energy consumption. Opportunistic routing has already inspired a RPL variant [11] aimed at improving downward traffic; this is not yet the case for synchronous transmissions, whose even higher performance gains are currently obtained in networks of homogeneous devices using the same radio chip [15].

New approaches with significantly better performance have emerged since the RPL standard was defined, and should therefore receive attention.

E. New Wireless Technology

The new scenarios fostered by IoT motivated the development of new wireless technology tackling lower power consumption, increased range, or both. Some of these, e.g., Wi-Fi HaLow (IEEE 802.11ah), are a natural evolution of existing technologies, and likely to be easily accommodated into RPL. At the other extreme, Low-Power Wide-Area Networks (LPWANs) hold the potential to radically change the picture of IoT networking as currently defined by RPL. With a low-power budget similar to IEEE 802.15.4 devices, LPWANs provide long-range communication (2–5 km in urban environments, 30 km and higher otherwise) at the price of reduced bandwidth and data rates. At a minimum, LPWANs are likely to become an inescapable alternative for interconnecting different sub-networks at a geographical scale; however, they also hold the potential to induce a rethinking of multi-hop routing strategies. In between these extremes, IoT technologies hitherto applied mostly on consumer appliances are being optimized for ultra low-power consumption, as in the case of Bluetooth Low Energy (BLE), therefore becoming an appealing alternative for fulfilling goals similar to RPL.

The scenarios fostered by IoT are triggering a new wave of wireless technologies that can significantly redefine the goals, and therefore the mechanisms, of RPL.

V. What Future for RPL?

In this paper we analyzed the extent to which the RPL standard lived up to the expectations defined by IETF requirements, and highlighted other challenges that emerged since its definition as a standard. RPL had its quota of successes, clearly contributing to the advancement of communications in the world of tiny, embedded, networking devices.

The concise analysis we provided in this paper, however, also highlighted several weaknesses that, in our opinion, may undermine a larger IoT adoption of RPL. The complexity of today's IoT makes it difficult for a single standard to paper over the significant differences in application requirements and heterogeneity in hardware constraints, both likely to be exacerbated in the near future.

We argue that, to remain successful in the IoT domain, RPL needs a re-targeting. For example, RPL could become a general standard *framework*, providing the "glue" for a suite of inter-related standards, each focused on specific communication technologies and/or application profiles. Together, the framework and the specific standards would provide interoperable alternatives for composing the network stack most suitable for the application at hand, therefore balancing the desire to support a broad range of features with the efficiency necessary when a vertical domain is tackled.

In part, this effort is already ongoing, as witnessed by the definition of the ancillary standards we mentioned in Section III. However, standardization bodies must keep up

with the latest developments and trends, and accordingly re-charter the respective working groups (e.g., IETF ROLL), in an effort to create a "standard ecosystem" around RPL and weave it into state-of-the-art approaches from related research communities.

REFERENCES

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and Alexander R. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550, IETF, March 2012.

[2] A. Brandt, J. Buron, and G. Porcu. Home automation routing requirements in low-power and lossy networks. RFC 5826, IETF, April 2010.

[3] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen. Building automation routing requirements in low-power and lossy networks. RFC 5867, IETF, June 2010.

[4] M. Dohler, T. Watteyne, T. Winter, and D. Barthel. Routing requirements for urban low-power and lossy networks. RFC 5548, IETF, May 2009.

[5] K. Pister, P. Thubert, S. Dwars, and T. Phinney. Industrial routing requirements in low-power and lossy networks. RFC 5673, IETF, October 2009.

[6] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm. RFC 6206, IETF, March 2011.

[7] T. Clausen, A. C. de Verdiere, and J. Yi. Performance analysis of Trickle as a flooding mechanism. In *Proc. of the IEEE Int. Conf. on Communication Technology (ICCT)*, 2013.

[8] C. Cobarzan, J. Montavont, and T. Noel. Analysis and performance evaluation of RPL under mobility. In *Proc. of the IEEE Int. Symp. on Computers and Communications (ISCC)*, 2014.

[9] J. Ko, J. Jeong, J. Park, J. Jun, O. Gnawali, and J. Paek. DualMOP-RPL: Supporting multiple modes of downward routing in a single RPL network. *ACM Trans. on Sensor Networks*, 11(2), March 2015.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proc. of the ACM Int. Conf. on Embedded Networked Sensor Systems (Sensys)*, 2009.

[11] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the tree bloom: Scalable opportunistic routing with ORPL. In *Proc. of the ACM Int. Conf. on Embedded Networked Sensor Systems (Sensys)*, 2013.

[12] T. Istomin, C. Kiraly, and G.P. Picco. Is RPL ready for actuation? a comparative evaluation in a smart city scenario. In *Proc. of the European Conference in Wireless Sensor Networks (EWSN)*, 2015.

[13] E. Ancillotti, R. Bruno, and M. Conti. On the interplay between RPL and address autoconfiguration protocols in LLNs. In *Proc. of the Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*, pages 1275–1282, July 2013.

[14] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *Proc. of the ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2012.

[15] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient Network Flooding and Time Synchronization with Glossy. In *Proc. of the ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011.

BIOGRAPHIES

Oana Iova is a postdoctoral researcher at the University of Trento. Her research is in creating energy-efficient wireless systems for Internet of Things applications using as building blocks both emerging wide-area wireless communication and conventional short-range technologies. She received her PhD from University of Strasbourg, France, in 2014 and her M.S. degree in Computer Science from Ecole Normale Superieure de Lyon, France, in 2011.

Gian Pietro Picco is a Professor at the Department of Information Engineering and Computer Science (DISI) at the University of Trento, Italy. His research spans the fields of software engineering, middleware, and networking, and is oriented in particular towards wireless sensor networks, mobile computing, and large-scale distributed systems. He is an associate editor for ACM Trans. on Sensor Networks (TOSN) and IEEE Trans. on Software Engineering (TSE).

Timofei Istomin is a PhD candidate at the University of Trento, doing research on protocols for autonomous wireless sensor and actuator networks. He received his M.S. degree in Computer Science from Lomonosov Moscow State University in 2003 and worked as a software engineer designing and implementing communication protocols for wireless networked embedded systems.

Csaba Kiraly is a researcher at the Bruno Kessler Foundation, Trento, Italy. His main research interests are in design and performance evaluation of networking protocols, with particular focus on WSN and IoT systems. Csaba Kiraly is co-author of more than 30 scientific papers. He graduated in Informatics and Telecommunications from the Budapest University of Technology and Economics in 2001, and he received his PhD from the same university.