Load Balancing In Routing Protocol For Low Power And Lossy Networks (RPL) Used In Iot

Manish Mishra, Dr. Piyush Shukla, Dr. Rajeev Pandey

Abstract: RPL is the IPv6 routing protocol for low-power and lossy networks, standardized by IETF in 2012 as RFC6550. Specifically, RPL is designed to be a simple and inter-operable networking protocol for resource-constrained devices in industrial, home, and urban environments, intended to support the vision of the Internet of Things with thousands of devices interconnected through multihop mesh networks. More than six-years have passed since the standardization of RPL, and we believe that it is time to examine and understand its current state. In this paper, 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 over 97 [41] 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. The RPL routing protocol published in RFC 6550 was designed for efficient and reliable data collection in low-power and lossy networks. Specifically, it constructs a Destination Oriented Directed Acyclic Graph (DODAG) for data forwarding. However, due to the uneven deployment of sensor nodes in large areas, and the heterogeneous traffic patterns in the network, some sensor nodes may have much heavier workload in terms of packets forwarded than others. Such unbalanced workload distribution will result in these sensor nodes quickly exhausting their energy, and therefore shorten the overall network lifetime. In this paper, we propose a load balanced routing protocol based on the RPL protocol, named LB-RPL, to achieve balanced workload distribution in the network. Targeted at the low-power and lossy network environments, LB-RPL detects workload imbalance in a distributed and non-intrusive fashion. In addition, it optimizes the data forwarding path by jointly considering both workload distribution and link-layer communication qualities. We demonstrate the performance superiority of our LB-RPL protocol over original RPL through extensive simulations.

Index Terms: Buffer capability utilization; Cooja Simulator; IoT; Load balanced routing; workload detection; Low power and lossy networks, RPL.

1. INTRODUCTION

1.1. RPL

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],[5],[6],[7], and is considered a critical component that links the low-power network connectivity to application layers in the IETF protocol suite for LLNs. These studies range from the domain of optimal param-eter selection for target applications to interoperability and performance testing among different implementations. Using open implementations of RPL, some work focuses on evalu-ating 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 implementa-tions 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.

- *Manish Mishra is currently pursuing masters degree program in Computer Science and Engineering in University Institute of technology-Rajeev Gandhi Proudyigiki Vishwavidyalaya, Bhopal, India, PH-9039059567. E-mail: mpmishra96@gmail.com*
- *Dr. Piyush Shukla is currently working as an Assistant Professor in Department of Computer science and Engineering,in University Institute of technology-Rajeev Gandhi Proudyigiki Vishwavidyalaya, Bhopal, India PH-9425378576 E-mail: pphdwss@gmail.com*
- *Dr. Rajeev Pandey is currently working as an Assistant Professor in Department of Computer science and Engineering,in University Institute of technology-Rajeev Gandhi Proudyigiki Vishwavidyalaya, Bhopal, India PH-7987263852 E-mail: rajeev98iet@gmail.com*

1.2. 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 with the help of RPL standardization, vari-ous 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 deliv-ery 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 dif-ferent 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 require-ments. 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] and [13]. 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 Neighbour Discovery; a node may use DIS to probe its neighbourhood for nearby DODAGs. 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 con-trol packet) depends on not only the number of neighbor nodes but also the number of RPL instances they participate in (i.e., neighbor nodes × 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[50]. 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.

1.3. INTERNET OF THINGS (IoT)

Internet of Things is the greatest challenge and opportunity for today's embedded IP-based devices such as sensors, home appliances, machinery, building automation equipment, and even toys. Over the past decade, advances achieved in terms of microcontrollers, power, and microelectronics technology motivated industry to benefit IP-based smart devices (called Smart Objects) services on the Internet. These services not only contain data created by humans but also data highly related to the physical world including sensor data, monitoring and control machines, and other types of physical items. This new field of Internet provides applications that are crucially related to stability, efficiency, and safety. Building automation, health, energy efficiency, smart grid communication, environmental monitoring are instances of such applications [44]. Exact estimation of IoT dimensions is difficult. Users don't affect its growth. Due to a rapid increase in the number of embedded IP-based devices, it is assumed that soon the IoT will outpace all other equipment on the Internet in terms of size (number of nodes) and that its growth will continue to increase at a high rate. The greatest potential for future growth is

low-power wireless networks and equipment that are not yet IP-based. 6LoWPAN technology was developed to realize Internet of embedded devices. This will be achieved by simplifying IPv6 capabilities and header compression format and by taking the nature of wireless networks into account [44]. As per reports from Forbes.com, the market for the Internet of Things is expected to reach around US$ 267 billion by 2020. Analysis from Gartner underlines that around 8.4 billion objects, with investments amounting to US$ 273 billion, will be interconnected with each other in the current year (2017-18). This represents a 31 per cent increase over the figures of 2016-17.[48]

Some of the key applications of IoT include[48]:

- Smart cities, retail points, smart grids, agriculture and farming, homes and offices
- The Internet of Vehicles (IoV)
- Connected cars
- Connected railways infrastructure
- Wearable devices
- Software defined networking
- The industrial Internet
- Energy management

2. LITERATURE SURVEY

2.1 RESEARCH ANALYSIS - STATISTICS AND SUMMARY

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 commu-nity. 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 [14], RIOT [20]–[23], and FreeRTOS in [24]. 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 equiva-lent) 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 plat-forms have an IEEE 802.15.4 transceiver. An interesting point is that, despite continuous advancement of IoT platforms, 'TelosB' [25], 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 [26], RPL implementations considering the use of 'TelosB' may not (and does not) provide the full functionality of RPL.

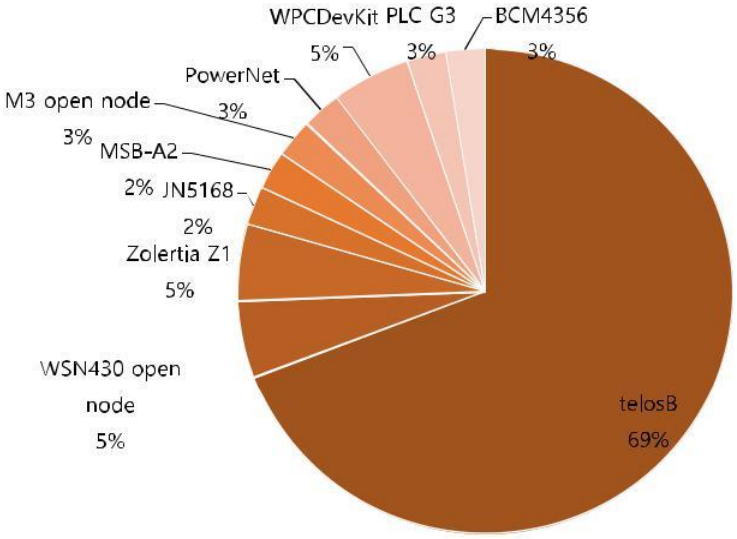

***Fig. 1**. Distribution of hardware platform*

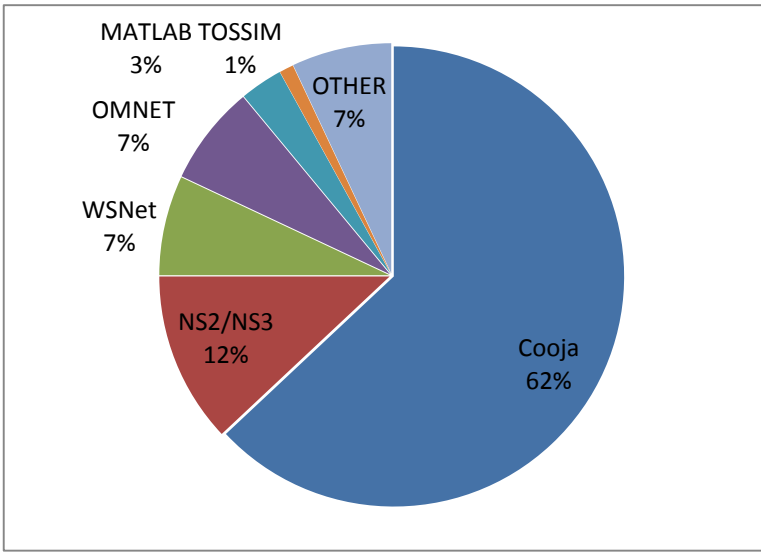

***Fig. 2**. Distribution of simulation method.*

For simulations studies, COOJA simulator [27] using ContikiOS/ContikiRPL implementation was the dominant method – 62.9% [41]of all the simulation studies used COOJA simulator, as shown in Fig. 2. Runners-up were the NS-2/NS-3 [28], 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[41]. It turns out that, TOSSIM simulator only supports Mica2/MicaZ platforms for simula-tions,

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 [29].

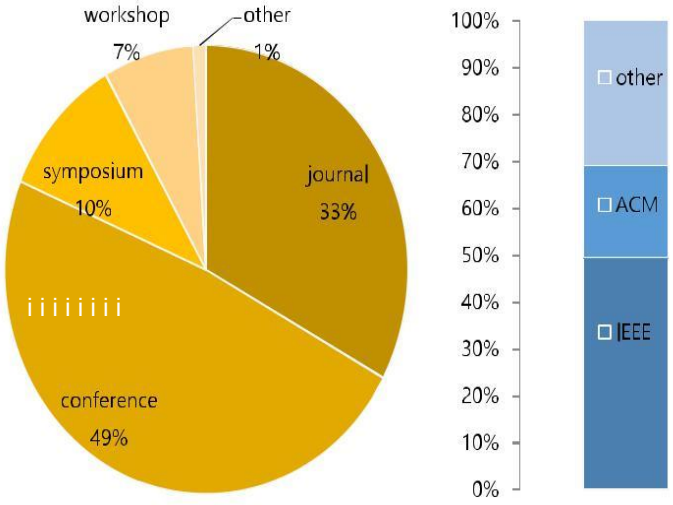

Fig. 3*. Distribution of publication venue*

For simulations studies, COOJA simulator [27] using ContikiOS/ContikiRPL implementation was the dominant method – 62.9% of all the simulation studies used COOJA simulator, as shown in Fig. 2. Runners-up were the NS-2/NS-3 [28], 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 simula-tions, 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 [29]. 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. By Publication Venue and Demographics: Fig. 3 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.

2.2 Literature Survey:

(There have been extensive studies on load balanced routing in a large-scale LLN environment. Depending on the routing structure, existing approaches can be classified into hierarchi-cal or flat routing model. Hierarchical routing approaches [3]– typically organize the network into clusters, where sensor nodes within one cluster can directly communicate with each other. Workload balance among sensor nodes usually focuses on either the arrangement of clusters [5] or the selection of cluster heads [3]. Because of the restriction of one-hop distance between nodes within a cluster, such a hierarchical approach may not be suitable for large-scale LLNs. On the contrary, flat routing model [7] typically requires packet trans-mission in a multi-hop fashion, and allows sensor nodes to make routing decisions by themselves. Although this approach is more suitable for large-scale LLNs, load balancing is more challenging due to the lack of global information. Routing algorithms proposed in [8]–[10] leverage the distributed, multi-hop feature of the flat routing model, and construct a logical tree structure to facilitate routing and load balancing. The establishment of the routing tree may rely on different metrics, e.g., [11], [12]. The tree structures may be dynamically changed according to the load distribution. To detect load imbalance, a threshold based approach is proposed in [10]. In this paper, we adopt the tree routing structure construction procedure by RPL protocol, and dynamically adjust routing paths according to workload distribution. Instead of relying on a predefined fixed threshold, we perform load imbalance detection in a distributed fashion.

Due to the lossy nature of wireless communications, multi-path based data collection approaches have been studied. To ensure reliability, data packets are forwarded through multiple paths towards the data collector [13],[14],[15]. In [13], a ran-domized forwarding mechanism is proposed to facilitate load balancing. However, simply spreading out workload among all neighbors may cause significant packet loss over low quality links. He et al. [14] proposed a multi-path geographic routing protocol, called SPEED. Although it achieves load balancing and reliability through multiple routing paths, the information of each node's geographic location may not be available for many application scenarios. Yan et al. [15] proposed a similar load balanced routing approach like our LB-RPL protocol. However, since they do not have a distributed load imbalance signaling mechanism, simply spreading the load among all parent nodes may not be effective. That is, for a parent node with more children, its workload will still be heavier than others. In LB-RPL, a sensor node with heavy workload can signal its status by delaying transmission of DIO packets. In this way, all the neighbors can get the signal, and fewer nodes will select it as next hop for packet forwarding. Therefore, its heavy workload can be alleviated.) Tsiftes et al. [8] first evaluated the performance of ContikiRPL and Ko et al. [9], [40] first evaluated that of TinyRPL, which show that both the two representative RPL implementations provide reliable upward packet deliv-ery. Especially in [9] and [30], Ko et al. showed that TinyRPL provides upward packet delivery performance that is compa-rable to CTP [2]. Kim et al. [28] deployed a TinyRPL-based multihop network in an

urban marketplace, which confirmed the reliability of TinyRPL's upward packet delivery. Ancillotti et al. [15] 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 nodesSome authors designed RPLca+ that includes a fast link quality update of each neigh-bor based on DIS unicasting and priority-based neighbor table management [16], and evaluated its performance through both COOJA simulations and testbed experiments. Dawans et al. [31] evaluate ContikiRPL's performance in a large-scale testbedKhelifi et al. [33] 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 natureOliveira and Vazão [17] survey research on RPL-based mobility support. In addition, the authors evaluate the four routing protocols presented in [34]–[36] and [37] 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 proto-cols with fewer control traffic provide better packet delivery performance. Vucini˘c´ et al. [38] compare the performance of RPL and LOADng through COOJA simulations. Elyengui et al. [39] 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. Mayzaud et al. [40] address topological inconsistency attacks, which maliciously trigger local repairs (i.e., fre-quent resets of TrickleTimer). 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 consump-tion. They evaluate AT through COOJA simulations.

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. First of all, none of RPL's own security mechanisms are implemented in either TinyRPL or ContikiRPL. 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' [14], [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. Even before the RPL standardization, Hui and Culler [32] propose to forward data traffic through alternative routes tem-porarily to probe link qualities of a diverse set of nodes, which is not implemented in TinyRPL nor ContikiRPLIt 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 exhaus-tive enough to state that these features were 'never' imple-mented. However, our position is that there are too many 'optional' features in RPL.

3. PROBLEM STATEMENT & PROPOSED METHODOLOGY

3.1 PROBLEM STATEMENT

To select a primary parent for packet forwarding, the most commonly used routing metrics in RPL typically belong to a specific network layer. For example, RSSI captures the physical layer signal quality for communication; ETX represents the aggregated link layer communication quality. However, for data collection in LLNs, not only the transmission qualities between a sender and a receiver should be considered, but also the channel contention and resource limitations at the receiver side should be considered. The notations used to formulate the problem are summarized in Table I.

TABLE I
NOTATION SUMMARY

Symbol	Definition
p^b	packet drop probability due to buffer limitation
λ	packet arrival rate
μ	service rate
q_r	buffer size at a sensor node
$_pc$	packet drop probability at node i due to channel condition
p^c_{ij}	packet drop probability from node i to node j due to channel condition
d^s	overall packet delivery probability
$^s_{ij}$	number of packets forwarded from node i to node j
$^f_{ij}$	forwarding probability from node i to node j
S	one realization of workload distribution
T_i	timer value for node i
T_0	constant number to assist timer calculation

Consider the scenario where there are multiple sensor nodes transmitting packets to a single relay node, as depicted in Figure 3. Assume the aggregated number of packets generated by all source sensor nodes follows a Markov process. Due to the MAC layer contention, we assume the processing of the packets, i.e., forwarding of the packets, at the relay node is also a Markov process. Therefore, the system can be modeled as an M/M/1/K queue, where K denotes the buffer size at the relay node. According to the finite queue analysis in [49], From this equation we observe that the number of packet sources plays a critical role in determining the ratio of the packet arrival and departure. In addition, when the ratio equals to one the buffer size at the relay node also significantly affects the packet drop rate.

$$T_i = T_0 \times \text{Buffer Utilization Counter.} \quad (1)$$

Load balanced data forwarding: Unlike RPL, where a single sensor node in the parent set is selected as preferred parent according to a single metric, in our LB-RPL, the top k parent nodes are all considered as a potential next hop for data forwarding. The probability of node i to forward data packet to a particular parent node j is calculated as[49]

$$f_{ii} = \frac{(1 - p^c)}{\sum_{i=1}^{k} (1 - p_{ii}^c)} \quad .....(2)$$

There are many channel condition metrics that can be used in routing. Here, we adopt the overall packet delivery ratio to quantify the communication quality. Denote the packet drop rate that is determined by channel condition as pci. Combined with the buffer related packet loss, we can conclude that the

3.2 Proposed Algorithms

Algorithm 1 Sensor Node Initialization Procedure

Initialize parent set and buffer utilization counter

Update the latest received version number

Insert the DIO message source into parent set according to the message arrival time

Calculate its own rank value
Set timer value T_i according to Equation 1

Generate a DIO message with its own rank number and the latest version number
When timer T_i expires, broadcast a DIO packet with current rank and version number

Algorithm 2 LB-RPL: Load Balanced RPL Routing Protocol

A sensor node listens to the radio channel

Once a message M arrives, check the type of the message

if M is a DIO message then
if New version of DIO then
Invoke Sensor Node Initialization Procedure
else
if Current DIO version then
if Rank value carried in the message is less than current node's rank then
Insert the DIO message source to parent set according to message arrival time
end if
else
Discard this message
end if
end if
else
if M is a DAO message then
Process it according to RPL
else
if M a is data message then
if the data message comes from a child of current node then
increase workload counter
end if
Forward this message by choosing the first two parent nodes from parent table, and selecting one as next hop with probability Equation 2
end if
end if
end if

4. SIMULATION ENVIRONMENT

4.1 Cooja Simulator

Contiki is an operating system with a focus on low power IoT devices. Cooja is the Contiki network simulator. Cooja allows the large and small networks of Contiki motes to be simulated. This article takes the reader through the process of programming IoT with Contiki and Cooja. We are all surrounded by a number of gadgets, mobile devices, smartphones, wireless nodes and many other objects that are digitally connected in real-time. The Internet of Things (IoT) enables real world objects to interact with each other. These objects can share information and communicate in real-time to deliver better performance as well as security. IoT works by developing and integrating smart objects that can be controlled using remote network infrastructure. The term, the Internet of Things, was coined by Kevin Ashton in 1999. The implementation of IoT is now widespread because of high performance wireless technologies. Radio frequency identification (RFID) tags and sensors are the most important components of IoT. The RFID tags can be embedded

in real world devices and objects, which can be monitored remotely using software based applications. RFID readers can be used to locate, read and sense RFID implanted objects. Very small, micro-sized transmitting and receiving chips are integrated with RFID to help them communicate with distant objects[48].

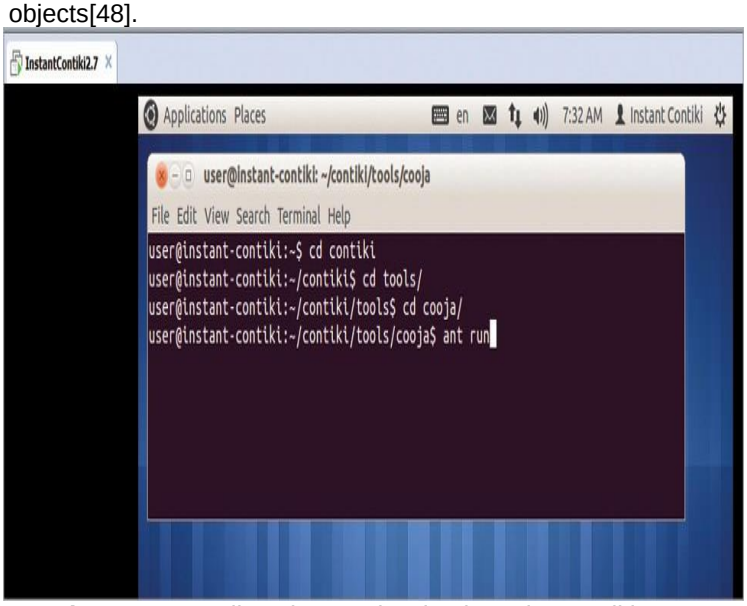

Figure 1: *Loading the Cooja simulator in Contiki*

behaviour of wireless motes in Cooja. Once the basic layout and working environment is ready, you need to import the RFID tags, sensor nodes or any other wireless devices that are to be connected and have to communicate over the IoT. In wireless networking and IoT, these are known as motes. There are many types of motes in Cooja that can be programmed. Physical motes, too, can be connected by using ports on the system so that real-time interfacing can be done. Every mote, with the base properties and programming APIs, is specified in the C source code at the back-end of Cooja. These C source code files can be customised and recompiled to get the new or desired output from these motes.

After compiling C code, a number of virtual motes can be imported in the simulation area so that the transmission of radio signals can be viewed and analysed. In this simulation of the IoT network, the scenario of a dynamic key exchange between the motes is done. Here, the dynamic security key is generated and authenticated for communication. In the IoT, for reasons of security, it is necessary to devise and implement the protocols and algorithms by which the overall privacy and security in communication can be enforced to avoid any intrusions. As the IoT can be used for military applications, it becomes mandatory to work on highly secured key exchange algorithms with the dynamic cryptography of security keys. Window, the log data can be copied and further analysed using data mining and machine learning tools for predictive analytics.

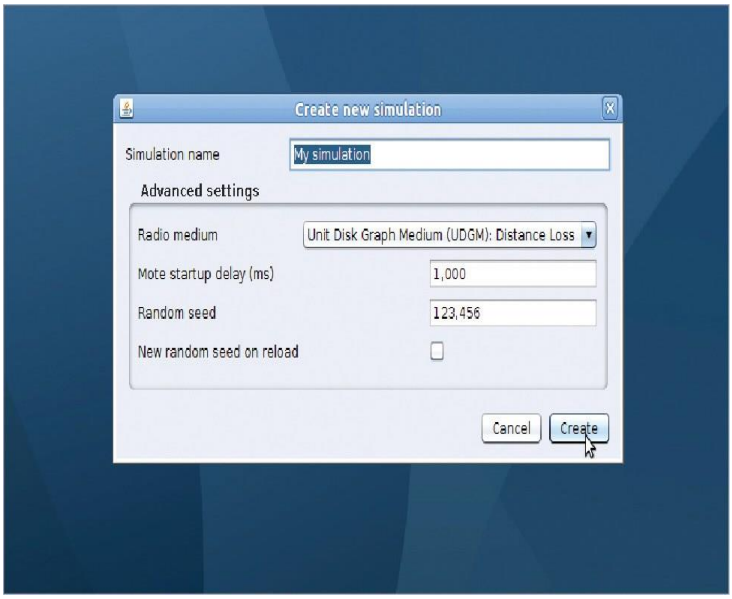

Figure 2*: Set-up of basic simulation properties in Cooja*

5. SIMULATION RESULTS

In order to consider LB-RPL in an exact manner, this routing protocol is simulated by Contiki COOJA simulator. Contiki was the first operating system that provided IP communication and is implemented in C programing language. The COOJA simulator is a Java-based sensor network simulator [29, 30].

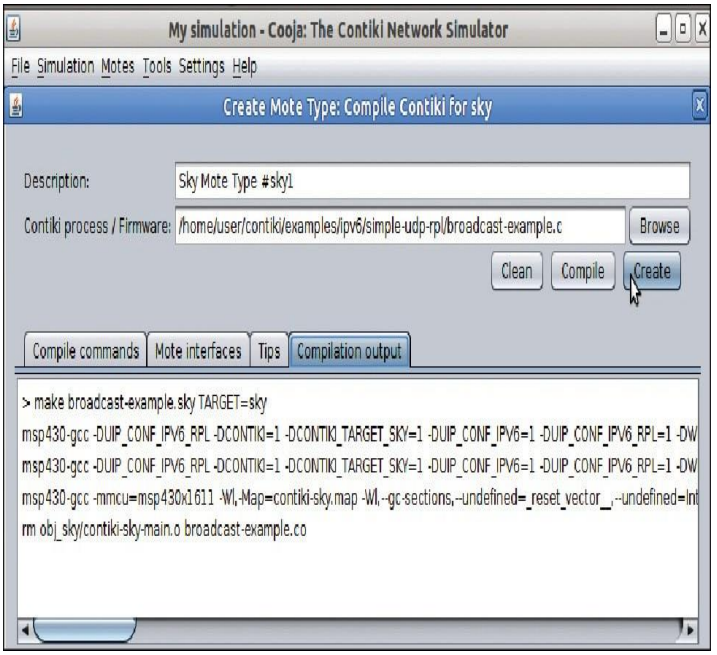

Figure 3 *: Compiling C source code for the desired*

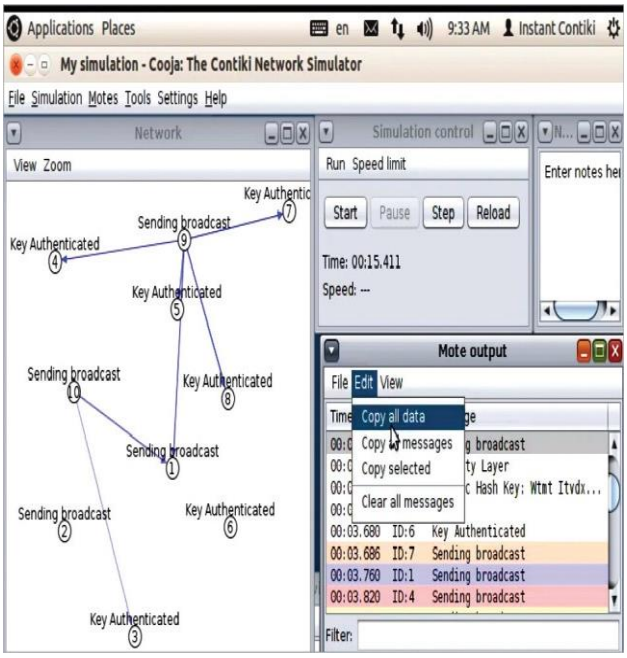

Figure 4: *Running simulation and behaviour analytics of motes.*

In the simulation, a total number of 250 static nodes are randomly deployed in a 500 by 500 square meters area and one LBR is located at the top. Simulation is run for 3 hours or 10800 seconds. In order to obtain more accurate results, nodes begin to send UDP packets after 7 minutes of starting time. In the simulation scenario, all nodes send packets with a length of a hundred bytes to the root node.

Table 2: *Simulation parameters*

Parameter	Value
Simulation time	10800 seconds
Number of nodes	250
Traffic Type	UDP
Packet length	100 bytes
Area	500*500 m^2
Simulator	Contiki COOJA
Repetition	5 times
Buffer size	8 UDP packets

Each node sends two packets with an interval of 30 seconds per minute. Each node has a buffer which is enable to hold maximum 8 UDP packets. In order to get more accurate results, the simulation scenario is repeated 5 times with different random seeds. We evaluated LB-RPL through packet delivery ratio, AVG end-to-end delay and root node throughput. Table 2 summarizes simulation parameters. Figure 5 illustrates packet delivery ratio versus simulation time. This figure proves that LB-RPL performs better than RPL in terms of packet delivery ratio with a 90% confidence. The simulated network consists of a great number of nodes and congestion occurrence is completely probable in such networks. RPL is not optimized for large scale networks and cannot balance network load in order to prevent performance reduction. This leads to noticeable packet loss. LB_RPL performs better in this case since this protocol distributes load and directs data traffic among multiple paths. Therefore LB-RPL provides higher PDR. Figure 6 illustrates AVG end-to-end delay versus simulation time. RPL performs better than LB-RPL in terms of AVG end-to-end delay and it is proven with 90% confidence. RPL makes routes based on the metrics and constraints formulated in OF. The OF used in RPL routing protocols is defined based on communication link quality. This results in better Rank for nodes providing a route toward the root with lower time. RPL provides better AVG end-to-end delay. In LB-RPL packets are distributed through the network, which results in longer paths. Packets may travel many nodes and links that are not completely in an optimized path toward the root. As LB-RPL does not use OF, it cannot formulate a delay in the routing process in order to decrease packets arrival time[44]. Figure 7 depicts root node throughput versus time. It can be seen that LB-RPL performs better than RPL with 90% confidence. LB-RPL prevents from intermediate nodes buffer over-utilization and lets packets to reach to destination through alternate routes even though they are longer. This decreased packet loss since buffer overflow is prevented. RPL selects routes with better overall Rank resulting in congestion in a part of network. Simulation result analysis was performed according to [42, 43]. Initial data deletion was performed and the replication process through techniques presented in Tables 3 to 5 present the simulation result report and confidence intervals [44].

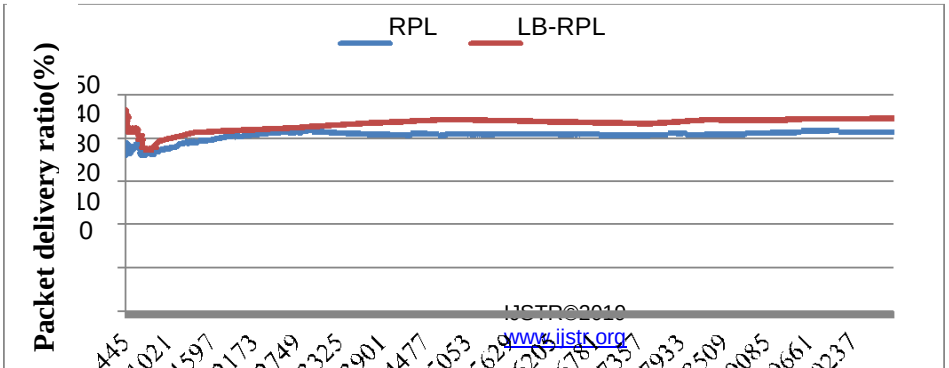

Fig. 5: Packet delivery ratio comparing graph.

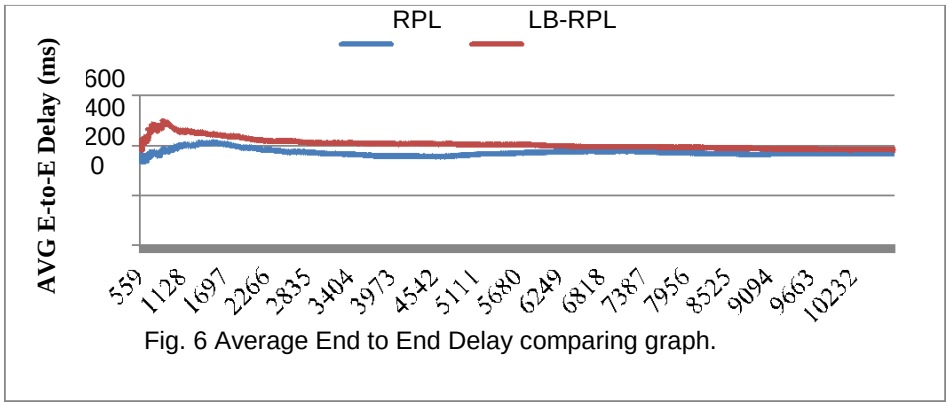

Fig. 6 Average End to End Delay comparing graph.

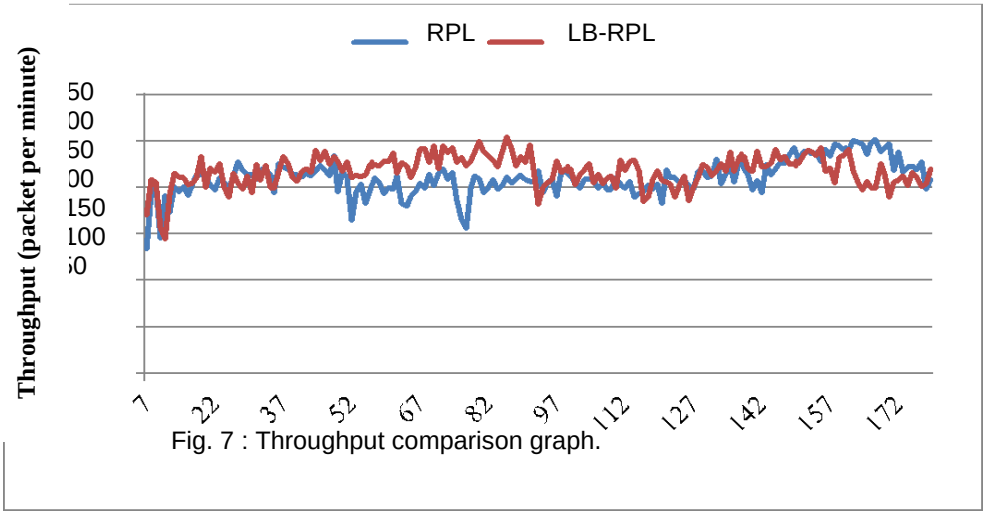

Fig. 7 : Throughput comparison graph.

Table 3: *The mean performance of the approaches for packet delivery ratio*

Method	Mean	CI
RPL	41.005	(39.052 , 42.958)
LB-RPL	43.432	(41.272 , 45.592)

Table 4*: The mean performance of the approaches for average end to end delay*

Method	Mean	CI
RPL	372.532	(360.978 , 384.086)
LB-RPL	398.484	(390.507 , 406.461)

Table 5: *The mean performance of the approaches for throughput of root node*

Method	Mean	CI
RPL	210.543	(204.403 , 216.683)
LB-RPL	217.751	(212.196 223.306)

ACKNOWLEDGMENT

RPL routing protocol is the best choice for routing in low-power and lossy networks but Load balancing is not considered in RPL routing protocol standardization process by IETF. As RPL routing protocol and LLNs are used in large scale networks, Load balancing is an important issue to be considered. Several methods have been proposed to alleviate the problem of load balancing. But they did not consider all aspects of RPL or some of them were not computationally efficient. LB-RPL is a new modification of RPL routing protocol, designed in order to tackle RPL load balance problems. This routing protocol distributes data traffic through the network and performs better than RPL in congestion prevention. In this paper, RPL and LB-RPL were evaluated in terms of packet delivery ratio, AVG end-to-end delay and network throughput. LB-RPL performs better in terms of packet delivery ratio and network throughput. Although LB-RPL prevents congestion, it is not the mature solution for RPL load balance problem cannot formulate application routing requirements and does not support advantages that IETF provided in RPL design process. A new version of RPL that saves IETF standardized design specifications and balances load especially in large scale networks is completely required.

REFERENCES:

[1] T. Winter et al., “RPL: IPv6 routing protocol for low-power and lossy networks,” Internet Eng. Task Force, Fremont, CA, USA, RFC 6550, Mar. 2012.

[2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol,” in Proc. ACM Int. Conf. Embedded Netw. Sensor Syst. (SenSys), Berkeley, CA, USA, 2009, pp. 1–14.

[3] S. Dawson-Haggerty, A. Tavakoli, and D. Culler, “Hydro: A hybrid routing protocol for low-power and lossy networks,” in Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm), Gaithersburg, MD, USA, Oct. 2010, pp. 268–273.

[4] T. Watteyne, Ed., et al., “Routing requirements for urban low-power and lossy networks,” Internet Eng. Task Force, Fremont, CA, USA, 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,” Internet Eng. Task Force, Fremont, CA, USA, RFC 5673, Oct. 2009.

[6] A. Brandt, J. Buron, and G. Porcu, “Home automation routing require-ments in low-power and lossy networks,” Internet Eng. Task Force, Fremont, CA, USA, 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,” Internet Eng. Task Force, Fremont, CA, USA, RFC 5867, Jun. 2010.

[8] N. Tsiftes, J. Eriksson, and A. Dunkels, “Low-power wireless IPv6 routing with ContikiRPL,” in Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN), Stockholm, Sweden, 2010, pp. 406–407.

[9] J. Ko et al., “ContikiRPL and TinyRPL: Happy together,” in Proc. Workshop Extending Internet Low Power Lossy Netw. (IP+SN), Apr. 2011.

[10] J. P. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, “Routing metrics used for path calculation in low-power and lossy networks,” Internet Eng. Task Force, Fremont, CA, USA, RFC 6551, Mar. 2012.

[11] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, “The Trickle algorithm,” Internet Eng. Task Force, Fremont, CA, USA, RFC 6206, Mar. 2011.

[12] P. Thubert, “Objective function zero for the routing protocol for low-power and lossy networks (RPL),” Internet Eng. Task Force, Fremont, CA, USA, RFC 6552, Mar. 2012.

[13] . Gnawali and P. Levis, “The minimum rank with hysteresis objective function,” Internet Eng. Task Force, Fremont, CA, USA, RFC 6719, Sep. 2012.

[14] J. Ko et al., “DualMOP-RPL: Supporting multiple modes of downward routing in a single RPL network,” ACM Trans. Sensor Netw., vol. 11, no. 2, pp. 1–20, Mar. 2015.

[15] E. Ancillotti, R. Bruno, and M. Conti, “The role of the RPL rout-ing protocol for smart grid communications,” IEEE Commun. Mag., vol. 51, no. 1, pp. 75–83, Jan. 2013.

[16] E. Ancillotti, R. Bruno, and M. Conti, “Reliable data delivery with the IETF routing protocol for low-power and lossy networks,” IEEE Trans. Ind. Informat., vol. 10, no. 3, pp. 1864–1877, Aug. 2014.

[17] A. Oliveira and T. Vazão, “Low-power and lossy networks under mobility: A survey,” Comput. Netw., vol. 107, pp. 339–352, Oct. 2016.

[18] H.-S. Kim et al., “MarketNet: An asymmetric transmission power-based wireless system for managing e-price tags in markets,” in Proc. ACM Int. Conf. Embedded Netw. Sensor Syst. (SenSys), Seoul, South Korea, Nov. 2015, pp. 281–294.

[19] Connected Grid Networks for Smart Grid—Field Area Network, Cisco, San Jose, CA, USA, accessed: May 2017. [Online]. Available: http://www.cisco.com/c/en/us/solutions/industries/energy/external-utilities-smart-grid/field-area-network.html

[20] RIOT OS. Accessed: May 2017. [Online]. Available: http://riot-os.org/api/group__net__gnrc__rpl.html
RIOT RPL. Accessed: May 2017. [Online]. Available: https://github.com/RIOT-
[22] OS/RIOT/tree/master/sys/net/gnrc/routing/rpl
[23] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "TRAIL: Topology authentication in RPL," in Proc. Eur. Workshop Wireless Sensor Netw. (EWSN), Graz, Austria, 2016, pp. 59–64.
[24] 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 Proc. ACM Int. Workshop Pervasive Wireless Healthcare (MobileHealth), Paderborn, Germany, Jul. 2016, pp. 11–16.
[25] O. Balmau et al., "Evaluation of RPL for medium voltage power line communication," in Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm), Venice, Italy, Nov. 2014, pp. 446–451.
[26] Moteiv Corporation. Tmote Sky. Accessed: May 2017. [Online]. Available:
[27] http://www.moteiv.com/products/tmotesky.php
[28] M. P. Andersen, G. Fierro, and D. E. Culler, "System design for synergistic, low power mote/BLE embedded platform," in Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN), Vienna, Austria, Apr. 2016, pp. 1–12.
[29] N. Tsiftes et al., "A framework for low-power IPv6 routing simulation, experimentation, and evaluation," in Proc. ACM Conf. Appl. Technol. Archit. Protocols Comput. Commun. (SIGCOMM), New Delhi, India, Sep. 2010, pp. 479–480.
[30] L. Bartolozzi, T. Pecorella, and R. Fantacci, "Ns-3 RPL module: IPv6 routing protocol for low power and lossy networks," in Proc. Int. ICST Conf. Simulat. Tools Tech. (SIMUTOOLS), Mar. 2012, pp. 359–366.
[31] K. Iwanicki, "RNFD: Routing-layer detection of DODAG (root) node failures in low-power wireless networks," in Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN), Vienna, Austria, Apr. 2016, Art. no. 13.
[32] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of RPL and 6LoWPAN in TinyOS," in Proc. Workshop Extending Internet Low Power Lossy Netw. (IP+SN), Apr. 2011.
[33] N. Khelifi, S. Oteafy, H. Hassanein, and H. Youssef, "Proactive maintenance in RPL for 6LowPAN," in Proc. Int. Conf. Wireless Commun. Mobile Comput. (IWCMC), Dubrovnik, Croatia, Aug. 2015,pp. 993–999.
[34] J. W. Hui and D. E. Culler, "IP is dead, long live IP for wireless sensor networks," in Proc. ACM Int. Conf. Embedded Netw. Sensor Syst. (SenSys), Raleigh, NC, USA, Nov. 2008, pp. 15–28.
[35] S. Dawans, S. Duquennoy, and O. Bonaventure, "On link estimation in dense RPL deployments," in Proc. IEEE Conf. Local Comput. Netw. Workshops, Clearwater, FL, USA, Oct. 2012, pp. 952–955.
[36] A. E. Korbi, M. B. Brahim, C. Adjih, and L. A. Saidane, "Mobility enhanced RPL for wireless sensor networks," in Proc. 3rd Int. Conf. Netw. Future (NOF), Gammarth, Tunisia, Nov. 2012, pp. 1–8.
[37] C. Cobârzan, J. Montavont, and T. Noël, "Analysis and performance evaluation of RPL under mobility," in Proc. IEEE Symp. Comput. Commun. (ISCC), Funchal, Portugal, Jun. 2014, pp. 1–6.
[38] K. C. Lee et al., "A comprehensive evaluation of RPL under mobility," Int. J. Veh. Technol., vol. 2012, Mar. 2012, Art. no. 904308.
[39] O. Gaddour et al., "Co-RPL: RPL routing for mobile low power wire-less sensor networks using corona mechanism," in Proc. IEEE Int. Symp. Ind. Embedded Syst. (SIES), Pisa, Italy, Jun. 2014, pp. 200–209.
[40] M. Vucini˘c,´ B. Tourancheau, and A. Duda, "Performance compari-son of the RPL and LOADng routing protocols in a home automation scenario," in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Shanghai, China, Apr. 2013, pp. 1974–1979.
[41] S. Elyengui, R. Bouhouchi, and T. Ezzedine, "LOADng routing proto-col evaluation for bidirectional data flow in AMI mesh networks," Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 6, pp. 37–43, 2015.
[42] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "Mitigation of topological inconsistency attacks in RPL-based low-power lossy networks," Int. J. Netw. Manag., vol. 25, no. 5, pp. 320–339, 2015.
[43] Hyung-Sin Kim, Jeonggil Ko, David E. Culler, "Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 19, NO. 4, FOURTH QUARTER 2017
[44] Parsaei MR, Rostami SM, Javidan R. (2016) A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset. International Journal of Advanced Computer Science and Applications, 7(6):20-25.
[45] Parsaei MR, Javidan R, Kargar NS, Nik HS. (2017) On the global stability of an epidemic model of computer viruses. Theory in Biosciences, 136(3-4):169-178. Doi: 10.1007/s12064-017-0253-2.
[46] MOHAMMAD REZA PARSAEI, AHMAD REZA PARNIAN, SAMANEH MIRI ROSTAMI AND REZA JAVIDAN, "RPL LOAD BALANCING IN INTERNET OF THINGS" IIUM Engineering Journal, Vol. 18, No. 2, 2017"
[47] Taheri R, Parsaei MR, Javidan R. (2016) A New Method for Optimizing Energy Consumption in Wireless Sensor Networks Using Enhanced LEACH Protocol. Journal of Engineering and Applied Sciences, 100(3):576-581.
[48] Deering S, Hinden R. (1998) Internet protocol, version 6 (IPv6) specification
[49] Shelby Z, Bormann C. (2011) 6LoWPAN: The wireless embedded Internet. The wireless embedded Internet [BOOK],

[50] https://opensourceforu.com/2017/06/programming-internet-things-using-contiki-cooja/
[51] U. Bhat, An introduction to queueing theory: modeling and analysis in applications. Birkhauser, 2008.
[52] Manish mishra, piyush shukla, Rajeev pandey., A Survey on Different Tools Used for Simulation of Routing Protocol for low-Power and Lossy Networks (RPL), IJCSE, Vol.7(7), pp. 359-363