Future Generation Computer Systems 82 (2018) 200–219

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A survey on routing protocols supported by the Contiki Internet of things operating system

Yousaf Bin Zikria [a], Muhammad Khalil Afzal [b], Farruh Ishmanov [c], Sung Won Kim [a], Heejung Yu [a,*]

[a] Department of Information and Communication Engineering, Yeungnam University, South Korea
[b] Department of Computer Science, Comsats Institute of Information Technology, Wah Campus, Pakistan
[c] Department of Electronics and Communication Engineering, Kwangwoon University, Seoul, South Korea

HIGHLIGHTS

- Provides an overview of the Contiki OS file structure and briefly explains Contiki OS network stack (netstack).
- Classifies the Contiki routing protocol research.
- Summarizes the literature according to the categorization.
- Lists the potential future work.
- Provides open research issues and directions.

ARTICLE INFO

Article history:
Received 27 October 2017
Received in revised form 2 December 2017
Accepted 24 December 2017
Available online 4 January 2018

Keywords:
Contiki OS
ContikiRPL
Routing protocols
Internet of things

ABSTRACT

Standardization and technology advancements have helped the realization of the Internet of things (IoT). The availability of low-cost IoT devices has also played a key role in furthering IoT research, development, and deployment. IoT operating systems (OSs) provide integration of software and hardware components. The availability of standard protocols, heterogeneous hardware support, ease of development, and simulation or emulation support are desirable features of IoT OSs. Contiki OS is one of the contenders for future IoT OSs. It was proposed in 2003, and since then, it has been continually under development and upgraded by professionals, academia, and researchers. Contiki OS supports open source, Internet standards, power awareness, dynamic module loading, and many hardware platforms. The diverse applications of IoT, including smart homes, smart health, smart cities, require efficient network connectivity and demand intelligent routing protocols that can handle heterogeneous, mobile, and diverse networks. Subsequently, designing routing protocols for memory- and central processing unit (CPU)-constrained IoT devices is a very challenging task. Therefore, this paper surveys the state-of-the-art routing protocols of Contiki OS. To the best of the authors' knowledge, this is the first study to classify the Contiki OS routing protocol literature and list the potential challenges and future work.

1. Introduction

The Internet of things (IoT) [1] is an emerging topic that connects durable goods, cars and trucks, industrial and utility components, and sensors to the Internet with data analytics capabilities. IoT promises to offer a fully connected smart world to transform everyday work, play, and life. IoT devices integrate the different objects through the interaction of software along with wireless enabling technologies. IoT operating systems (OSs) provide the environment for software interaction, and without it, IoT devices would just be non-functioning devices. The IoT OS provides various flexible features that facilitate integration of electronic products and technologies.

IoT resource-constrained devices possess low-power embedded computational and network-enabled devices to keep production costs low. These IoT devices offer low storage and memory capabilities. Thus, the IoT OS should have the following properties: small memory footprint, support for heterogeneous hardware, energy efficiency, network connectivity, real-time capabilities, and security.

* Corresponding author.
E-mail addresses: yousafbinzikria@ynu.ac.kr (Y.B. Zikria), khalilafzal@ciitwah.edu.pk (M.K. Afzal), farruh@kw.ac.kr (F. Ishmanov), swon@yu.ac.kr (S.W. Kim), heejung@yu.ac.kr (H. Yu).

https://doi.org/10.1016/j.future.2017.12.045
0167-739X/

Table 1
Comparative overview of existing reviews on Contiki OS.

Discussed features/Routing protocols	This review	[7]	[9]	[8]
RPL	√	√		
LOADng	√	√		
Opportunistic routing	√			
Secure routing	√			
Context aware routing	√			
IPV6 neighbor discovery protocols	√			√
Open research issues and recommendations	√	√		

Contiki OS [2] is an open-source, lightweight, and event-driven IoT OS designed for resource-constrained embedded systems. Contiki OS requires at least 2 kB random access memory (RAM) and 30 kB read only memory (ROM). Contiki OS offers both multithreading and optional preemptive multithreading. It is based on lightweight stack threads called protothreads [3]. Contiki OS achieves resource efficiency using an event-driven kernel and preemptive multithreading. It is designed to run on low-power battery-driven IoT platforms, which are intended to run for many years without human intervention. It supports standard wireless communication stacks. Further, it supports a range of hardware from different companies such as Texas Instruments (TI), Advanced RISC Machine (ARM), and Atmel. Moreover, it provides the simulation software to run, debug, and test the development before flashing the program into real IoT devices.

Contiki OS further provides a shell, file system, database system, runtime dynamic linking, cryptography libraries, and a fine-grained power-tracing tool. It includes complete testing facilities such as unit, regression, and full system integration. The primary programming language of Contiki OS is C. However, Java [4] and Python [5] can be used in the runtime environment. Along with Contiki OS, Tiny OS [6] is also a popular and widely used OS. It also targets very constrained devices. Tiny OS uses nesC language primitives that make it hard to learn and it lacks a large and active developer community. Therefore, Contiki OS becomes the choice for IoT research and commercial systems. It is actively developed by the professionals, researchers, and academia.

1.1. Motivation

Contiki OS is extensively developed and used by the research community. There are many proposals for each layer of transmission control protocol/Internet protocol (TCP/IP). However, to the best of the authors' knowledge, no previous study has provided a detailed research summary for each layer. Hence, this is the first paper to provide a comprehensive survey of proposed routing protocols, classification, and future research directions. This should help academia, researchers, and developers to kick start their work by learning the Contiki OS routing protocol research performed so far. Table 1 summarizes the contribution of this survey as drawn from the recent literature in the field. To the best of author's knowledge, there is no detail survey on the topic. The survey [7] mainly focuses on the RPL, The authors of [8] only discussed IPV6 neighbor discovery protocol. However, in contrast to [7,8], this survey paper pay attention to the comprehensive classification that covers related papers up to the most recent 2017 literature, and detail open research recommendations and directions for each classification for the routing protocols supported by Contiki IoT OS.

1.2. Related work

Hahm et al. [9] defined three design categories of IoT OS. The first category includes multithreaded OSs. In a multithreading OS, each thread runs in its own context and manages its own stack. Hence, it introduces memory and runtime overhead. The second category is event-driven OSs. An event-driven OS waits for an external interrupt and processes it according to its handler. This approach is memory efficient and yields low complexity. However, it imposes certain restrictions on programmers as it is not easy to express all programs in a finite state machine (FSM). The final category is real-time OSs. It guarantees to meet the time deadline of any process and imposes strict constraints to the developers. Thus, these strict restrictions result in an inflexible OS and porting to other hardware platforms is rather difficult. It is anticipated that the future IoT devices will be smaller, cheaper, and much more energy efficient. However, they are unlikely to have more memory or CPU power [10].

Farooq and Kunz [11] provided a classification framework to compare the existing OSs with regards to core OS features. They defined architecture, reprogramming, execution model, and scheduling as the core OS features. Further, in evaluation they also considered power management, simulation support, and portability. They analyzed different applications and suggested the ideal OS for those applications to draw a clear line for an application developer. Reusing [12] laid out the main requirements for an OS based on limited resources, concurrency, flexibility, and low power. Following this, he compared TinyOS [6] and Contiki OS. He concluded that TinyOS was better suited when resources were really scarce. In comparison with TinyOS, Contiki OS was the best choice when flexibility was the main concern.

Willmann [13] provided a valuable insight into Contiki OS. He found that an event-driven system over preemptive multithreading reduces the memory footprint. However, this feature can be added to those processes that explicitly require it. Al-Fuqaha et al. [14] presented an overview of enabling technologies, protocols, applications, and the recent research in IoT. They briefly discussed Contiki OS features such as language support, minimum memory requirements, programming model, dynamic memory allocation, Codo [15], the filesystem-level security, and routing attacks detection using an intrusion detection system (IDS) [16]. Ranjan et al. [17] studied Contiki OS in terms of the programming model, memory management, and architecture. Further, they discussed the aforementioned issues with the various OSs for wireless sensor networks (WSNs) [18–20].

1.3. Main contributions

The network layer is responsible for the correct delivery of data received from the upper layer to its destination and reception of data received from the lower layer. It is evident from the related work that the literature lacks a detailed survey on the routing protocol supported by Contiki OS. Therefore, this paper is the first work, to the best of the authors' knowledge, to provide an overview of the Contiki OS routing protocol and contributions of this paper are as follows:

- Provides an overview of the Contiki OS file structure;
- Briefly explains Contiki OS network stack (netstack);
- Classifies the Contiki routing protocol research;
- Summarizes the literature according to the categorization;
- Lists the potential future work;
- Provides open research issues and directions.

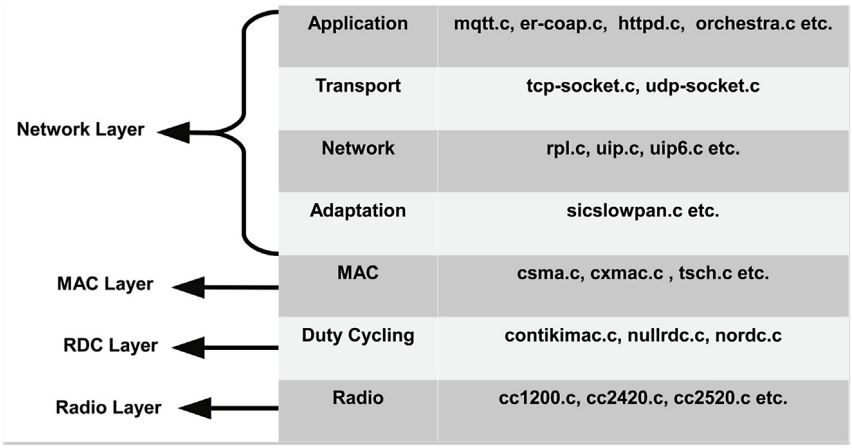

Fig. 1. The Contiki OS network stack.

1.4. Paper organization

Table 2 lists the abbreviations used in this paper. The rest of the paper is organized as follows. Section 2 provides a general overview of Contiki OS file structure and discusses the Contiki OS netstack. Section 3 classifies and summarizes the Contiki OS supported routing protocols literature. Section 4 lists future research recommendations and directions. Finally, Section 5 concludes the paper.

2. Contiki OS overview

OS is software to manage and control hardware and software resources of a device, in which OS is used [3,11]. Because of the factors affecting the design of OS, different types of OS are proposed and in use today. Although those different types of OS share common components and basic operations, the differences lie in resource allocation for OS operations, implementations of operations, existence of new kinds of operations, absence of some basic operations, etc. One of the widely-used OSs today is an embedded OS [12]. Recent developments in micro-electro-mechanical systems (MEMS) lead to the wide use of the embedded OS in different applications and devices. Compared to a computer system OS, the embedded OS can be highly customizable, application-specific and suitable for real-time applications [11]. Another important type of OS is a network OS, which is oriented to computer networking. These kinds of OS can be used in servers, routers, switches, and so on to support networking operations. Although the computer system OS and embedded OS can support communication protocol stack and networking, but a level and type of the support, and functions for networking operations make difference between the former two OSs and the network OS. Another key difference is that network OS encompasses not only a single device, but it also encompasses several devices and coordinates, manages, and controls networking operations and network resources among these devices.

Contiki OS contains app, cpu, dev, platform, core, tools, doc, regression-tests, and examples directories. Contiki OS app directory contains supported applications. It contains IoT supported applications ranges from coap, mqtt, etc., to the webserver. The cpu directory contains specific microcontroller (MCU) files. The dev directory lists all the external chips and devices. The platform directory includes all the specific libraries and drivers. The core directory contains all the Contiki OS core files and libraries. The tools directory contains flashing tools, debugging, and simulation. The doc directory encompasses the self-generated doxygen documentation. The regression-tests directory incorporates all the nightly regression tests. The examples directory consists of comprehensive ready to build examples. Table 3 lists the Contiki OS file structure.

There are four fixed layers in Contiki OS. Fig. 1 shows the Contiki OS netstack. It contains radio, radio duty cycle (RDC), medium access control (MAC), and network layers. Contiki OS automatically forms a wireless Internet protocol version6 (IPv6) [21] network with the help of the routing protocol. The network layer is further subdivided into the upper IPv6 layer and the lower adaptation layer. The function of the adaptation layer is to provide IPv6 and user datagram protocol (UDP) header compression and fragmentation to transport IPv6 packets with a maximum transmission unit (MTU) [22]. MAC provides the functionality of collision avoidance and backoff. RDC is essential to achieve energy efficiency while maintaining network communication. It saves energy by allowing a node to keep the radio transceiver off most of the time. The radio layer receives bytes or full packets via interrupt handlers. The incoming data is buffered and the process is polled. This polling mechanism causes the process to pass on to the special event and then finally it is passed to the upper layers.

3. Contiki OS supported routing protocols

Routing is a crucial operation in network and communication systems, which deals with packet delivery from source to destination [7,8]. The choice of OS and design of protocol stack have direct effects on the performance of routing protocols [9]. For example, if OS does not support real-time applications, real-time related issues cannot be addressed efficiently by a routing protocol or network stack alone [9]. Moreover, IoT devices are considered to be under resource constraints, especially in terms of energy and storage [13]. Hence, OS should manage efficiently resources of the devices by exploiting trade-off between resource utilization efficiency and protocol performance. Another aspect of OS, which can affect performance of a routing protocol, is availability of programming interface to easily update/upgrade of protocols. This is important for various reasons. For example, over the time bugs or pitfalls of protocol implementation might be detected and the upgrade might be needed to improve the performance of the protocol. In such cases, OS provides smooth update/upgrade. Hence, in this section we examine proposed protocols from different aspects including impacts of Contiki OS.

The IoT constitute IoT devices that can store data on the servers or clouds. The IoT devices are battery operated and need to be energy efficient. Thus, they may run for many years without human intervention. The network can be very large and nodes messages have to travel multi-hop before reaching the central or sink node.

Table 2
List of abbreviations.

Symbol	Description
3DES	Triple Data Encryption Standard
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ACO	Ant Colony Optimization
AES	Advanced Encryption Standard
AES-CMAC	AES Cipher-based Message Authentication Code
AH	Authentication Header
AMI	Advanced Metering Infrastructure
AT	Adaptive Threshold
BMRF	Bidirectional Multicast RPL Forwarding
CBC–CS	Cipher Block Chaining–Cipher Stealing
CCI	Channel Check Interval
CMAC	Cipher-based Message Authentication Code
CORB	Context-aware Opportunistic Resource-based
CTI	Cross Technology Interference
DAO	Destination Advertisement Object
DAO-ACK	DAO Acknowledgment
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination-Oriented Directed Acyclic Graph
EBAR	Energy Balancing Adaptive Routing Protocol
ECC	Elliptic Curve Cryptography
EE	Energy Consumption
ESP	Encapsulating Security Payload
E-TRICKLE	Enhanced Trickle
ETX	Expected Number of Transmissions
FSM	Finite State Machine
iACK	Implicit Acknowledgment
IDS	Intrusion Detection System
IKE	Internet Key Exchange
Imin	Minimum Interval
IoT	Internet of Things
IP	Internet Protocol
LLN	Low-power and Lossy Networks
LOADng	Lightweight On-demand Ad-hoc Distance Vector Routing Protocol — Next Generation
MAC	Medium Access Control
ME-TRICKLE	Modified E-Trickle
MIB	Management Information Base
MRHOF	Minimum Rank with Hysteresis Objective Function
MTU	Maximum Transmission Unit
ND	Neighbor Discovery
netstack	Network Stack
OCB	Offset Codebook Mode
OF	Objective Function
opt-TRICKLE	Optimized Trickle
ORPL	Opportunistic RPL
ORPL-LB	ORPL Load Balancing
OS	Operating System
PA	Precision Agriculture
PDR	Packet Delivery Ratio
pro-RPL	Proactive RPL
PRR	Packet Reception Ratio
QoS	Quality of Service
RAM	Random Access Memory
RDC	Radio Duty Cycle
ROM	Read Only Memory
RPGM	Reference Point Group Mobility
RPL	Routing Protocol for Low-Power and Lossy Networks
RRP	Ripple Routing Protocol
RWM	Random Waypoint model
SCAR	Sensor Context-aware Routing
SINR	Signal-to-interference-plus-noise ratio
SMRF	Stateless Multicast RPL Forwarding
SNMP	Simple Network Management Protocol
TI	Texas Instruments
TM	Trickle Multicast
WSN	Wireless Sensor Network
μIP	MicroIP

All the communication within the IoT devices uses wireless technology. This raises many problems and designers must propose very effective and reliable solutions that can run for long periods of time. The network layer focuses on how packets are routed towards the destination. Hence, each node takes a decision to route the packet to the next hop. All these decisions are made by the routing protocol. The IoT devices are very limited in terms of hardware resources and, hence, many IoT OSs use MicroIP (μIP) stack for the communication. The MicroIP stack supports IPv6 along with IPv6 over low-power wireless personal area networks (6LoWPAN) protocols. There are many routing protocol proposals for Contiki OS. Fig. 2 depicts the Contiki routing protocol proposal date.

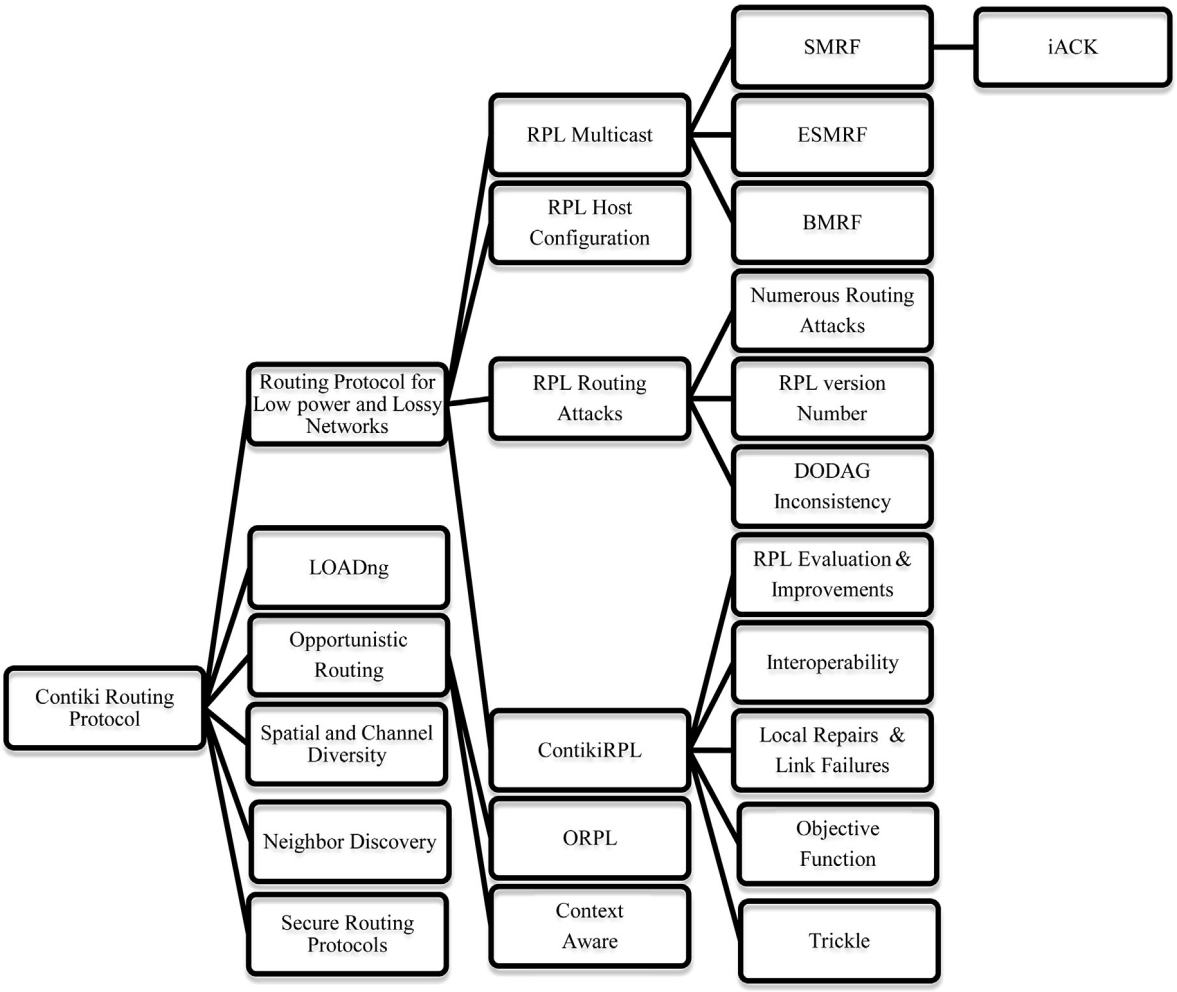

Fig. 2. Contiki OS routing protocols proposed in the literature.

Table 3
Contiki OS directory structure.

Directory	Description
app	Contiki OS supported applications.
cpu	MCU files
dev	External chip and devices
platform	Specific files and platform drivers
core	Contiki core files and libraries
tools	Flashing, debugging, simulation
doc	Self-generated doxygen documentation
regression-tests	Nightly regression tests
examples	Ready to build examples

3.1. Routing protocol for low-power and lossy networks

Low-power and lossy networks (LLNs) consist of processing-, memory-, and energy-constrained devices. Hence, traditional protocols such as open shortest path first (OSPF) [23], optimized link state routing protocol (OLSR) [24], routing information protocol (RIP) [25], ad hoc on demand distance vector (AODV) routing protocol [26], and dynamic source routing (DSR) protocol [27,28], etc., cannot be used in LLN. Routing protocol for LLNs (RPL) [29] is an effective solution for constrained IoT devices. RPL was preliminary designed to connect billions of IoT devices in the future. It is based on the famous distance-vector (DV) routing approach and run on top of many link layer mechanisms, including the IEEE 802.15.4 standard. RPL is highly scalable and adaptive to network conditions. Whenever default routes are inaccessible, it provides alternative routes to the destination. It disseminates information using Trickle [30] over dynamic network topology.

RPL builds a destination-oriented directed acyclic graph (DODAG) in which all leaf nodes have one route to the root node. Initially, each node sends DODAG information object (DIO) as an advertisement presenting itself as a root node. This message is disseminated in the network and eventually the whole DODAG is built. When a node wants to communicate to the other node, it sends a destination advertisement object (DAO) to its parents, the parents direct this message to the root node and eventually root node routes it to the destination. New node joins the network by sending a DODAG information solicitation (DIS) request to the root node and root node confirms the joining by sending back a DAO acknowledgment (DAO-ACK). RPL nodes can be stateless or stateful. A stateless node keeps track of its parent only. In this case, only root node has the complete knowledge of DODAG and, hence, all the traffic goes through the root node. In the case of a stateful node, it keeps the information of its children and parents. Hence, it manages all the communication within the subtree of DODAG and forwards the rest of the traffic to the root node. The objective function (OF) is used to obtain the optimized routes towards the DODAGs. The OF ranks the node which is the approximate distance from the node to the DODAG root node. It also tells nodes how to select the parent nodes. Fig. 3 represents an RPL node rank diagram.

3.1.1. ContikiRPL

3.1.1.1. RPL evaluation and improvements. This section covers basic the ContikiRPL implementation, evaluation in various scenarios,

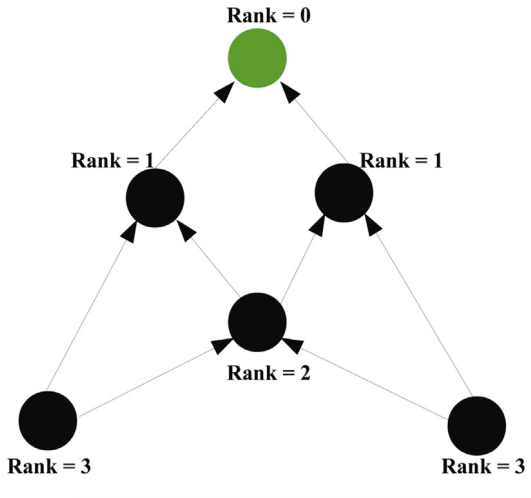

Fig. 3. RPL node rank.

impact of mobility, and improvements such as reliability, throughput, network lifetime, and load balancing. Tsiftes et al. [31] designed and implemented RPL inside the uIPv6 stack [32]. They verified the implementation using simulation on a testbed of 13 nodes in an office environment. They ran ContikiRPL on top of the Contiki MAC protocol and measured the power consumption of the system. They used power efficiency and implementation complexity as the performance metrics. The testbed experiments were performed for one night and the simulation ran for 10 000 s. The nodes generate 40 packets per minute. In simulations, they obtained a 100% packet delivery. However, in experiments they incurred 10% packet loss. The duty cycle of leaf nodes is around 0.5%–0.8% and that of routing nodes is 1%–3%. These results reveal that it is highly energy efficient and nodes can last for many years. Further, ContikiRPL is also memory efficient and code can reside in only 6%–8% of ROM and RAM.

Chen et al. [33] evaluated the basic behavior of ContikiRPL in precision agriculture (PA) using simulation. They evaluated time to find the first source–destination (src-dst) pair, time to complete convergence, and time to achieve a completely stable network. The authors concluded that the first two metrics are not affected by the number of nodes in the network. However, the metric of time to achieve a completely stable network depends on the total number of nodes. The drawback of this paper is that they proposed the theoretical PA architecture without any simulations or testbed results. Hence, the performance of the proposed architecture is questionable.

Acillotti et al. [34] first found the drawbacks of ContikiRPL and then proposed an improved RPL to increase reliability. They conducted experiments using a 90-mote testbed and 100-node simulation evaluation. Initially, they pointed out the problems of ContikiRPL and summarized these as follows. ContikiRPL shows very high packet losses. Packet losses are not related to the number of hops. Once node choses its parent with a bad link, the Contiki OS link estimation technique does not allow the node to switch to a better parent. Afterwards, they proposed a cross-layer design approach to increase the link quality estimation capability and manage neighbor information adaptively and efficiently. Their results depict a clear improvement in packet delivery rates, decreases in end-to-end delay, and increased energy efficiency. Furthermore, they found that asynchronous duty cycling decreases the packet delivery ratio (PDR) due to increased network contention and this resulted in more packet collisions. Chemate1 and Pingat [35] proposed a cross-layer design approach to enhance the reliability of data transmissions. They discussed the coordinated policies to oversee RPL and IP neighbor tables. However, the authors did not validate the proposed scheme at all. Therefore, the effectiveness of the proposal cannot be assumed.

Lamaazi et al. [36] analyzed the performance characteristics of RPL in different network scenarios. They considered the number of received and lost packets, the number of hops, the routing metric (Rtmetric), the expected transmission count (ETX), and power consumption. All these metrics increase with the number of nodes. They also evaluated mobility by using a random waypoint model (RWM) and random walk model [37]. The mobility model directly affects the packets reception ratio. Therefore, it also consumes more energy. The number of sink nodes also plays a crucial role in energy consumption: as the number of sink nodes increases, overall energy efficiency also increases. This work needs testbed evaluation. Further, it also needs evaluation using reference point group mobility (RPGM), nomadic, and self-similar least action walk [38–41].

Khan et al. [42] proposed a sink-to-sink coordination framework by utilizing RPL periodic route maintenance messages. The authors aimed to enhance the overall throughput and network lifetime. The sink node dynamically adjusts the network size for load balancing. Hence, it achieves higher throughput. They simulated random and grid multi-sink topologies. The results portrayed improved throughput compared with RPL. However, this approach requires further testbed evaluation and an investigation of whether the proposed scheme is scalable with an increasing number of nodes. Moreover, how the proposed scheme performs in the case of sink failure should be checked.

Banh et al. [43] developed a radio module method to estimate the mote energy consumption (EE) in choosing the path to relay the data to the sink. They combined ETX and EE to achieve load balancing. They used a 25-mote simulation scenario to verify the results. The results depict improved energy balance with improved energy efficiency and PDR. Nevertheless, this scheme needs testbed evaluation. Further, it needs validation in a dense network topology.

Kim et al. [44] proposed a mechanism to provide multi-hop routes to downward traffic-centric applications. They performed the experiment on real testbed using 31 TelosB motes. In the test bed environment, the proposed DT-RPL showed better performance in terms of packet delivery ratio, overheads, and duty-cycle. Riker et al. [45] provide energy-harvesting efficiency in a multi-hop network using a neutral operation approach. They used simulation-based 40 nodes and implemented the proposed solution (RAME) on ContikiRPL. RAME uses the information of lowest energy node to find an optimal path. The simulation-based result showed the energy consumption of minimum energy node is controlled in an efficient manner. However, for better verification, the testbed implementation is required. Similarly, another research is being done to provide a better congestion mechanism [46]. The proposed mechanism (OHCA) considered 25 nodes and utilized three metrics including buffer occupancy, expected transmission count and queuing delay. The mathematical analysis along with simulation shows OHCA improves performance in terms of overall throughput, average weighted fairness index, end-to-end delay, energy consumption and a number of lost packets by an overall average of more than 28.36%, 28.02%, 48.07%, 31.97% and 90.35%, respectively. These results show that it is very efficient to improve overall performance. However, the proposed solution introduces significant overheads in the network. Sheeraz et al. [47] provide a better parent selection mechanism for RPL scenario. They combined the ETX and Hop count metric and proposed a hybrid solution to utilize the good aspects of both matrices. The simulation-based results obtained from 20 nodes network scenarios are compared with ETX, ELT and HOP metric. The results show

poor performance of the proposed solution in terms of a number of parent changes, and a number of control packets. EC-MRPL is an energy efficient mechanism proposed by Maha et al. EC-MRPL uses a mobility aware routing mechanism for RPL network. The proposed mechanism maintains the mobile nodes connectivity. The simulation-based study shows that signaling cost has been reduced by 24 times. Similarly, the handover delay is about 156.25 ms and 60.65 ms for the proposed protocol compared to 56640 ms for RPL. However, the better handover delay also leads to more power consumption. For mobile sensing motes, another mechanism is proposed called BRPL [48], which support time-varying data and mobility. The detailed study is carried out using experimental and simulation environment. They considered 100 nodes in both experimental and simulation setup. The results are evaluated based on packet loss rate, end-to-end delay and communication overheads. However, the energy consumption information is missing and which is required for detailed analysis.

Latib et al. [49] proposed other mobility strategies for the IoT network. To evaluate a mobility of RPL, they used 25 nodes and analyzed three scenarios, i.e., sink and sensor nodes are static, static sink and the mobile sensor node, sink and sensor nodes are mobile. Based on these scenarios they proposed three strategies for a better performance of RPL in mobility scenarios. First, in mobility scenarios must operate in lower packet rates. Second, it should use higher duty-cycle rate. Lastly, sink node must position in the center of mobile nodes. They provide the outcome of these three scenarios in terms of packet delivery ratio, power consumption and radio duty cycle. However, they did not compare it with the previously proposed mechanism.

Harith et al. [50] proposed mechanism for dynamic mobility to improve end–end delay and energy consumption. The solution is based on received signal strength identification. The considered metrics are packet delivery ratio, end-to-end delay, and energy consumption. They used 25 mobile nodes and one static sink node. The nodes move randomly at 0–5 m/s with a maximum pause of the 30 s. The simulation-based result shows that the proposed mechanism provides 78% packet delivery ratio.

IRPL is an energy efficient routing protocol for IoT networks [50], which is based on clustering technique. The simulation results show that the IRPL balance the energy consumption more efficiently, compared with the original RPL protocol. They used 50 nodes in a simulation environment that are distributed randomly. Energy consumption and network lifetime are used to measure the proposed algorithm performance.

3.1.1.2. Interoperability. IoT networks consist of heterogeneous devices, and interoperability is necessary for network operations. Unfortunately, most research considered only homogeneous nodes operating with the same OS. The implementation of different protocols, stacks, specifications, and non-standard nodes are the major hurdles in interoperability. This section discusses interoperability implementations.

Ko et al. [51] investigated the interoperability and performance of Contiki OS and TinyOS RPL implementations. This was the first work towards fully interoperable sensor networks. They used simulations to evaluate the performance of both protocols. They used the packet reception ratio (PRR) and number of parent changes as performance metrics. Their results revealed that subtle difference in underlying layers affects the performance. Furthermore, the simulator must also support the fine-grained timing and interoperability requirements on underlying layers. They only considered the RPL with objective function zero (OF0) that considers hop count as a metric for evaluation and cannot test it with the minimum rank with hysteresis objective function (MRHOF) that selects a route based on ETX due to the implementation complexity and code bugs. Moreover, this paper lacked a testbed evaluation to confirm the results.

Parasuram et al. [52] discussed the pros and cons of RPL and the proposed interoperable RPL-Lite. They summarized the issues that included redundant and unused features of RPL, unsupported RPL applications and potential features beneficial for RPL. Thereafter, they proposed the RPL-Lite version. However, it was only a theoretical proposal. Thus, it requires implementation, testing, and validation to fully realize its effectiveness.

3.1.1.3. Local repair and link failures. Local repair is triggered to find the alternative path without considering the optimal path. When the parents are unreachable, the node first tries to associate itself with a new root from the siblings to reach the parents. In case of no response from the siblings, it either waits for the next periodic sending of a DIO by its neighbors or sends a DODAG DIS message requesting a new DIO from the DODAG. Hereafter, the node finds the best parent and rejoins the network. This section briefly elaborates on local repairs literature.

Korte et al. [53] studied the ContikiRPL local repairs. They evaluated it using a testbed of six motes in a linear topology. This was the first work to analyze and test the ContikiRPL local repair process efficiency. Further, they designed and implemented an RPL management information base (MIB) using the Contiki simple network management protocol (SNMP) agent [54,55]. The RPL-MIB helps in studying the local repair process of ContikiRPL. They ran a comprehensive test and calculated the fallback parent, DIO timer, route lifetime, fallback sibling, greediness of nodes, poisoned tree, and rebuild build memory. The results showed that the ContikiRPL repair mechanism works well. They also found that the time taken to switch to a new parent and build a subtree is under the maximum time allowed for an RPL DODAG repair in the case of node failure or movement. However, route timeouts and global repairs consume extra energy. They evaluated the performance on a very simple linear topology although it is not realistic. Hence, it requires further evaluation considering a denser mote-deployment scenario.

Khelifi et al. [56] presented a robust and proactive RPL (pro-RPL) mechanism to mitigate the RPL nodal or link failures. They first provided an analysis and then validated it using simulation. Pro-RPL monitors node conditions to ensure timely detection of failures. They introduced a dynamic tunable suffering index to detect the chances of failure and trigger the pro-RPL on time to reconstruct the DODAGs according to the parents. The simulation results show that pro-RPL delivers more packets compared with ContikiRPL as the network size increases. Further, it saves energy by detecting failures in advance using the node suffering index and dynamically performs repairs before the failure happens. Thus, it offers a viable failure mitigation protocol and increases the lifetime of the network. However, pro-RPL requires further validation using a testbed. Future research directions include identifying the mobility patterns and probabilistic trajectory of a floating DODAGS node. Furthermore, inclusion of a dynamic load distribution in pro-RPL to prolong the network lifetime is also a very intrinsic problem. Alvi [47] proposed another mechanism for route and recovery maintenance. They only used seven nodes for the simulation study. The performance is measured using reconnection probability, reconnection time, and a number of packets. The proposed scheme provides lower reconnection time and route recovery mechanism but again it incurs the overhead problem.

3.1.1.4. Objective function. The OF specifies how the nodes pick and optimize routes within an RPL instance. The OF is used for the selection of a DODAG to join, the rank of each node within the DODAG, an ordered list of parents in the DODAG, and to select and optimize routes. Contiki OS by default supports OF. This section describes OF-related work.

The MRHOF uses ETX to make the network more reliable; however, it induces high routing latency. Kamgueu et al. [57] designed

and implemented a new RPL OF by combining several metrics to optimize the network performance. This was the first work to obtain a unique value using a fuzzy inference system to merge ETX, delay, and the nodes' remaining power. They evaluated the proposed scheme using a testbed. The results show that the proposed approach performs better than the traditional ETX-based [58] routing in terms of packet loss, routing stability, energy efficiency, and end-to-end delay. However, this work lacks multiple traffic flow with different quality of service (QoS) requirements.

Belghachi and Feham [59] proposed an OF-based ant colony optimization (ACO) [60]. First, they provided the mathematical analysis and later verified it using simulations. They used energy- and delay-aware routing metrics to make RPL more energy efficient. Simulation results showed improvement in energy efficiency, and as a result increased network lifetime. Further, the proposed method can achieve improved transmission precision and decrement in end-to-end delay in comparison with the ContikiRPL. However, it requires additional verification in a realistic testbed scenario.

A queue-state-based algorithm is proposed for the IoT network [61]. The algorithm is designed that utilize packet loss alert and then select parent based on it. They used 24 nodes in Cooja simulation. The algorithm achieves 47.1% and 75.0% lower packet loss ratios compared with QU-RPL and RPL with OF0. The testbed verification is required to understand the proposed solution in detail. Shakya et al. proposed smart energy efficient objective function [62]. It combined electricity, gas, and water smart meter meshed network using open IEEE/IETF protocols in line with a Wi-SUN IoT solution for smart metering and utility networks. The performance is measured in terms of energy efficiency and network lifetime. Simulation results show that up to 27% improvement is attained in the network lifetime. Although simulation-based results show improved energy consumption, however, for results that are more realistic, we need to utilize more nodes in the simulation environment. Design and analysis of RPL objective functions for multi-gateway ad-hoc low-power and lossy networks are studied by Muhammad et al. [63]. They utilized 75 TelosB nodes. The result shows 25% higher PDR, however, the total number of control packet transmissions is 95% higher for the available bandwidth-based protocol compared to the other protocols.

3.1.1.5. Trickle. The Trickle timer algorithm is used to control the construction and updating of DODAG. The DODAG contains information about how to forward the information received by every node. The Trickle controls the injection of routing traffic in the form of DIOs messages. Further, it also specifies the node listen time for new information and how often it sends out new information to its neighbors. This section outlines previous work related to Trickle. Benson and Kinicki [64] evaluated RPL comprehensively under different network topologies and node densities. They vary the RPL parameters and Trickle algorithm to evaluate its performance. Further, they focused on the Trickle timer algorithm to verify the number of routing packets generated for each node in the network. They ran 300 distinctive simulation tests for almost 400 h. They use four variants of Trickle; basic Trickle algorithm [6], optimized Trickle (opt-Trickle) [65], enhanced Trickle (E-Trickle) [66] and modified E-Trickle (ME-Trickle). All variants are sensitive to changes in configuration and network density. The results did not reveal a clearly dominant variant. However, ME-Trickle performed well for a lower number of hops in terms of PDR and lower setup time. Further, ME-Trickle performance degrades with the increasing number of hops. Therefore, an optimization is desirable for ME-Trickle to perform well with any number of hops. Further evaluation using a testbed, by varying packet reception probability, and with mobility is required.

Ghaleb et al. [67] proposed Trickle-Plus to enhance the Trickle algorithm to improve network convergence time and reduce power consumption. The simulation results show a reduction in network convergence time with a slightly increased power consumption or traffic overhead as compared with the Trickle algorithm. Nevertheless, it is just a preliminary result and requires testbed evaluation. Further, they evaluated the performance with a fixed redundancy value and, hence, it requires validation by varying redundancy values.

Yassein et al. [68] presented a dynamic Trickle timer algorithm to cater to a listen-only-period problem. They verified the proposed method using simulations. The results indicate an improvement in overall network convergence time and energy consumption. There was no significant improvement in PDR. The proposed scheme requires a testbed evaluation. Further, it needs to be evaluated in different network densities and topologies. Moreover, it requires verification with multiple RPL OFs.

Trickle timer is one of the major components of RPL parent selection mechanism, Muneer et al. [69] proposed Trickle timer strategy for IoT devices. They evaluated a network with different network densities (20, 40, 60, 80) using the simulation experiments. The performance is analyzed through packet delivery ratio (PDR), the convergence time, and the power consumption. The result shows 35% less convergence time in 20 nodes scenario, 62% less convergence time in 40 nodes, and about and when the network is made up of 60 nodes, around 70% less convergence time is consumed. Similarly, 76% less time when there are 80 nodes. Studying this algorithm in testbed environment would be an interesting future research direction.

Table 4 outlines ContikiRPL protocols analysis and future research directions.

3.1.2. ContikiRPL multicast

Network layer multicast forwarding provides the functionality of a one-to-many communications model. The services such as service discovery and network management use the multicast functionality. The multicast forwarding with trickle algorithm provides IPv6 multicast. It does not rely on topology information. Further, it uses the Trickle algorithm to control the number of packet exchanges and to avoid flooding.

3.1.2.1. Stateless multicast RPL forwarding. Oikonomou and Phillips [73] presented a stateless multicast RPL forwarding (SMRF) algorithm. SMRF takes advantage of the basic RPL construct, DODAG, and populated routing tables to perform multicast forwarding without any further control messages. They compared it with Trickle multicast (TM) [74] using simulations. TM is highly susceptible to changes in minimum interval (Imin) timer value and results in unpredictable performance and energy consumption. Therefore, SMRF copes with these issues and is faster and more energy efficient than the TM. SMRF makes multicast groups of data-interested nodes and forwards data to them instead of forwarding datagrams to all nodes. It does not have any control message overhead as it uses RPL parent information and multicast group information. Through simulations, it is shown that SMRF delivers packets more efficiently in terms of time and energy consumption than the TM. However, the optimal Imin is an open issue and it depends on an underlying duty cycling algorithm. Moreover, it requires investigation of the number of out of order packets arrival per hop basis, performance in dense networks, and testbed evaluation. Oikonomou et al. [75] extended SMRF for IPv6-based WSNs. They performed simulations using a testbed topology with 21 nodes and 11 motes. They considered PDR, end-to-end delay, out-of-order datagram delivery ratio, and energy consumption. By keeping Imin values low in the scenarios without employing duty cycling, better performance is achieved. However, in a realistic scenario with duty cycling, the value of Imin should be higher than the channel check interval (CCI) to deliver more packets in a shorter time. Higher Imin values give optimal performance and lower Imin value results

Table 4
ContikiRPL protocols analysis and future research directions.

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Protocols used	Future research directions
[31]	–	✔	✔	41/13	Power efficiency, Implementation complexity	ContikiRPL	Testbed verification on dense network
[33]	–	✔	–	200	Time to, find the first src-dst pair, complete convergence, achieve complete stable network	ContikiRPL	PA architecture implementation and testbed validation
[34]	–	✔	✔	100/90	Packet loss ratio, End-to-end delay, Neighbor table, Link quality estimation, Energy consumption	RPLc, RPLcs, RPLca, RPLca+	Use network coding, opportunistic transmissions, and data compression in conjunction with RPL to increase PDR
[35]	–	–	–	–	–	–	Requires implementation and validation using testbed and simulation
[36]	–	✔	–	20,40,60	Number of receive and lost packets, number of hops, Rtmetric, ETX and power consumption	ContikiRPL with static and mobile nodes	Needs testbed evaluation and validation using reference point group mobility (RPGM), nomadic and self-similar least action walk
[42]	–	✔	–	164	Throughput, end-to-end latency, energy consumption	RT-SSCF, RT-RPL, GT-SSCF, GT-RPL	Requires testbed evaluation, proposal scalability, proposal performance in case of sink failure
[43]	✔	✔	–	25	Individual and total network energy consumption, PDR	ContikiRPL-ETX, e2eEnergy, ETX+single Energy, ETX+e2eEnergy	Testbed evaluation and validation in dense network topology
[51]	–	✔	–	40	Packet reception ratio, Number of parent changes	TinyRPL	Testbed evaluation and by using different RPL objective functions
[52]	–	–	–	–	–	–	Requires implementation, testing and validation to fully realize its effectiveness.
[53]	–	–	✔	6	Fallback parent, DIO timer and fallback parent, route lifetime, fallback sibling, greediness of nodes, poisoned tree, rebuild build memory	ContikiRPL	Realistic and dense network testbed validation
[56]	✔	✔	–	31	Packet loss, average energy consumption, lifetime of the network	ContikiRPL, Pro-RPL	Testbed evaluation, identify the mobility patterns and probabilistic trajectory of floating DODAGS node, inclusion of dynamic distribution of load in pro-RPL
[57]	✔	–	✔	28	Packet loss ratio, Routing stability, Average remaining power	ContikiRPL-ETX, Fuzzy	Lacks multiple traffic flow with different QoS requirements
[59]	✔	✔	–	20	Energy efficiency, Transmission performance, End-to-End delay	ContikiRPL-ETX, QoSRPL	Requires additional verification in realistic testbed scenario
[64]	–	✔	–	40	Network convergence time, power consumption	ContikiRPL with original, optimized, E, ME Trickle	Testbed evaluation, ME-Trickle optimization, assessment by varying PRR and with mobility
[67]	–	✔	–	50	Network convergence time, number of sent control traffic messages, Power consumption	ContikiRPL with trickle 4, 14 and plus	Requires testbed evaluation, requires validation by varying redundancy values
[68]	–	✔	–	20,40	Convergence time, Power consumption, PDR	ContikiRPL-Standard, Dynamic	evaluation in different network topologies, with multiple RPL objective function, testbed

(*continued on next page*)

in more out-of-order packet delivery and more packet loss. SMRF has low complexity and requires lower memory. Hence, it is more suitable to many IoT constrained devices. However, it requires dense realistic simulation and testbed evaluation.

(i) Implicit acknowledgment SMRF: Tharatipayakul et al. [76] proposed iACK, an implicit acknowledgment in SMRF, to make transmission or delivery more reliable. TM uses retransmissions that result in high delivery but it increases delay and congestion.

Table 4 (*continued*)

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Protocols used	Future research directions
[44]	–	–	✔	31	Downward and upward link quality	ContikiRPL, DT-RPL	Mobility support
[45]	–	✔	–	40	Energy consumption	ContikiRPL, RAME	Testbed verification
[46]	✔	✔	–	25	Packet loss ratio, End-to-end delay, Energy consumption	ContikiRPL, OHCA, DCCC6,QU-RPL	Testbed verification
[47]	–	✔	–	20	No. of parent changes, Energy consumption	ContikiRPLwith HOP, ETX, ELT, HYBRID	Reduce control overhead
[48]	–	✔	–	7	Reconnection probability, Reconnection time at different DIS intervals, DIO transmission at various trickle timers.	ContikiRPL	Energy consumption validation and comparison
[70]	–	✔	–	14	Energy consumption, Handover delay, Packet delivery Ratio The	ContikiRPL, MRPL, EC-MRPL	Reduce power consumption
[49]	✔	✔	✔	100/100	Packet loss rate, End–end delay, Communication Overhead	ContikiRPL, BRPL	Reduce energy consumption
[50]	–	✔	–	25	Packet delivery ratio, Energy consumption, Duty cycle	ContikiRPL	Comparison with previous studies
[71]	–	✔	–	26	Packet delivery ratio, Energy consumption	ContikiRPL, mRPL, D-RPL	Testbed verification
[72]	✔	✔	–	50	Power consumption, Packet loss rate	ContikiRPL, IRPL	Testbed verification
[61]	–	✔	–	24	Packet loss ratio, Number of DIO messages	ContikiRPL, QU-RPL, QSPS	Testbed verification
[62]	–	✔	–	18	Energy consumption	ContikiRPL with MRHOF and SEEOF	Dense network evaluation
[63]	–	✔	–	75	Hop counts, Packet delivery ratio, Delay, Total retransmission	ConntikiRPL	Dense network evaluation
[69]	–	✔	–	20, 40, 60, 80	Power consumption, Convergence time	ContikiRPL with standard and elastic TRICKLE	Testbed verification

SMRF is fast but lacks reliability. iACK treats the child nodes rebroadcast as an implicit acknowledgment. They simulated it by using a 20-node topology and compared it with TM and SMRF. Simulation results showed that iACK increases the PDR and has a lower delay than the TM but a slightly higher delay than the SMRF. iACK can be adjusted to select the tradeoff between delay and PDR. Moreover, it requires more memory. Further, it requires performance evaluation with simulations under realistic and dense network topology.

3.1.2.2. Extended SMRF. The sending limitation of SMRF is due to the lack of complete visibility of DODAG at the intermediate nodes. Therefore, AbdelFadeel and Elsayed [77] introduced ESMRF, an extension to SMRF, to support a multicast-on-behalf scheme, which allows the nodes to send multicast traffic up and down the RPL tree. The multicast-on-behalf scheme allows intermediate nodes within the DODAG to send the multicast traffic to the root. Further, it retains low memory overhead. They use simulation topology with 20 nodes to compare ESMRF with TM and SMRF. They use network PDR and end-to-end delay as performance metrics. Simulation results show that if the root of RPL is the multicast source, then SMRF and ESMRF show the same performance. However, ESMRF performs better in cases when the highest rank node is the source. In a random topology, ESMRF performs best among all the compared protocols. The performance with different traffic patterns when the number of nodes and hops increase needs to be demonstrated.

3.1.2.3. Bidirectional multicast RPL forwarding. SMRF lacks dynamic group registrations, downwards forwarding, and additional delay to mitigate the collisions. Henceforth, Lorente et al. [78] proposed a new multicast protocol called bidirectional multicast RPL forwarding (BMRF). BMRF offers configurable forwarding, bidirectionality, delivery disorder avoidance, multi-sourcing, and dynamic group registration. They evaluate BMRF using a 51-node simulation topology and compared it with SMRF. They calculated PDR, the number of packet transmissions, the number of radio transmissions, energy consumption per delivered packet, and end-to-end delay. Experimental results show that BMRF mixed mode gives the best result for link layer broadcast and unicast. However, in a random topology, it gives good results for channel check rates higher than 8 Hz. BMRF unicast provides the best reliability but at the expense of higher delay and energy consumption. In general, BMRF takes more memory space, which is sometimes crucial for IoT constrained devices. Moreover, comparison should be made with ESMRF. It further requires a testbed evaluation to confirm the results.

Table 5 provides a summary of ContikiRPL multicast proposal analysis and future research directions.

3.1.3. RPL host configuration

The hierarchical structure addressing allows prefix aggregation, which enables contiguous IPv6 addresses to be aggregated into a single prefix. However, it introduces the address space insufficiency problem. Therefore, instead of using prefixes, Peres and Goussevskaia [79] proposed MHCL, a multihop host configuration strategy as a RPL subroutine to assign IPv6 addresses to nodes. The goal was to study the top-down multihop host configuration. They evaluated the performance of MHCL through simulation in a topology with 169 nodes. MHCL is based on cycle-free network structures to generate and assign the IPv6 addresses. Their aim

Table 5
ContikiRPL multicast proposal analysis and future research directions.

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Protocols used	Future directions
[73]	–	✔	–	21	Packet delivery ratio, end-to-end delay, energy consumption	SMRF, TM	Optimal Imin, out of order packets arrival per hop basis, performance in dense network and testbed evaluation
[75]	–	✔	✔	21/11	Packet delivery ratio, End to end delay, Out of order arrivals, energy consumption, code sizes, memory requirements	SMRF, TM	Requires dense realistic simulation and test bed evaluation
[76]	–	✔	–	20	Data receive ratio, packet delay, packet receive count	iACK, SMRF, TM	Requires evaluation in dense, realistic simulation and testbed topology, evaluation of memory usage in different network topologies.
[77]	–	✔	–	20	Network packet delivery ratio, end-to-end delay	ESMRF, SMRF, TM	Requires further evaluation with denser and realistic simulation and testbed topology, performance with different traffic pattern with increasing number of nodes and hops
[78]	✔	✔	–	51	PDR, number of packet transmissions, number of radio transmissions, energy consumption per delivered packet, end-to-end delay	BMRF, SMRF	Code optimization, comparison with ESMRF, testbed evaluation

Table 6
ContikiRPL host configuration proposal analysis and future research directions.

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Protocols used	Future directions
[79]	✔	✔	–	169	Network setup time, Number of DIO messages, Number of DAO messages, Addressing success rate, Top-down routing success rate	Contiki-RPL, MHCL	Requires dense network and testbed evaluation

was to keep the memory footprint low and achieve efficient, reliable top-down data traffic. They proposed two address allocation algorithms: greedy (MHCL-G) and aggregation (MHCL-A). MHCL-G assigns the addresses from root to leaves, whereas in MHCL-A, nodes compute or identify their descendants and send this to the root. Afterwards, the root allocates the address range according to the size of the subtree of each child node. Each node, after receiving the range from its parent node, assigns IPv6 addresses to its children. The simulation results show that it is memory efficient and takes less time to adjust according to the network dynamics. Further, the number of DIO control messages is the same as those of RPL and there are significantly fewer DAO messages than RPL. In addition, in MHCL the downward PDR is higher than in the RPL protocol in all the simulated network scenarios. This study requires dense network testing and a testbed evaluation. Table 6 summarizes the RPL host configuration protocol.

3.1.4. RPL routing attacks

The outcome of routing attacks results in one or all three possible actions: packet drop, packet alteration, and routing disruption. A malicious node may use different techniques to launch a particular attack; however, it results in any or all of aforementioned actions. Therefore, attack avoidance methods should be more effective if they consider the outcome of attacks rather than the attack feature or characteristics.

3.1.4.1. Numerous routing attacks. Wallgren et al. [80] presented a study of routing attacks and countermeasures in RPL. They also proposed and implemented a heartbeat protocol to cater to selective-forwarding attacks [81,82]. They used a simulation topology with 25 nodes. The study shows that RPL is susceptible to selective-forwarding, sinkhole, HELLO flood [83], Wormhole [84], Clone ID [85–87], and Sybil [88] attacks. The authors proposed IDS placement in the network to mitigate these attacks. Moreover, the proposed heartbeat protocol performs well for selective-forwarding attack and is energy efficient. However, the heartbeat protocol is fully effective if it is used in conjunction with IPsec and ESP. Thus, it eventually increases the network complexity. Further, this study requires a testbed evaluation.

Pongle and Chavan [89] proposed real-time intrusion and wormhole detection in IoT. RPL is prone to wormhole attacks. The purpose of a wormhole attack is to disrupt the network and traffic flow. They used simulation topologies with 8, 16, and 24 nodes. The authors' lightweight IDS successfully detected two types of wormhole attacks: packet relay and encapsulation. The proposed IDS required location information to efficiently detect the wormhole attack. This location information required less overhead and resulted in a high true-positive detection rate. Moreover, IDS is memory efficient and achieved a 94% detection rate. It requires an extension to make it generalized IDS. Following this, it requires evaluation to verify it against Sybil, clone ID, RPL version number, and local repair attacks.

3.1.4.2. RPL version number. Version number is typically used to ensure loop- and error-free topology. An attacker can modify the number version associated with the topology and resulting in the entire routing topology being rebuilt. This rebuilding of the topology results in increased overhead, deterioration of energy resources, unavailability of channel, and sometimes loops in the routing topology. This section highlights the literature related to version number attack.

Mayzaud et al. [90] studied the effects of version number attack on RPL. The RPL uses the version number to control the global repair process. Every DIO message includes the version number

and the receiving nodes recalculate their rank and update their version number. The authors used a grid topology with 20 nodes for simulation. The simulation results revealed that version number attacks significantly increase the control overhead. Consequently, they reduce the PDR and double the end-to-end delay. The position of the attacker also plays a crucial role in increasing the overhead: highest overhead occurs when the attacker is far from the root. Hence, researchers should focus on techniques to secure the RPL version number. However, as IoT devices are constrained, these mitigation techniques need to be very simple and memory efficient. Moreover, this research work requires further evaluation and verification in a realistic dense network topology.

Aris et al. [91] studied RPL version number attacks in detail. They used a network topology with 44 nodes to study the attacks. The network contains static and mobile nodes. The results show that control packet overhead is highly related to the attacker's location. However, the attacker's location only has a minor effect on the packet delay and power consumption. The analysis of attacking probability, p, shows that the network degrades with increasing p. However, it also increases the chances of detecting an attacker. The attacker needs to use a low value of p to remain undetected. The mobile attacker nodes have the same impact on the network as the attacker nodes that are far from the root. This research study lacks the solutions to mitigate these attacks.

3.1.4.3. DODAG inconsistency. The RPL uses IPv6 header options to keep track of topological inconsistencies. A malicious node or attacker can manipulate the IPv6 header options to launch DODAG inconsistency attack. This can lead to control overhead and, as a result, wasting of precious energy resources of IoT resource-constrained devices. Further, an attacker can use DODAG attack by sending an appended packet to next hop neighbors to force them to drop these packets and this can result in a black hole attack. The black hole attack leads to the denial of service attacks.

Mayzaud et al. [92] proposed an adaptive threshold (AT) mechanism to mitigate DODAG inconsistency attacks. The AT relies on a set of parameters and can lead to inadequate optimization. Their evaluation used a five-node chain and ten-node random network topology. They extended the AT to derive the appropriate threshold for counteracting DODAG inconsistency attacks. The simulation results indicated that the AT approach performs better than the default RPL with the fixed threshold. The AT approach mitigates blackhole DODAG inconsistency attack and, hence, it avoids channel congestion and achieves high resource utilization. The AT relies on a set of parameters and it can lead to inadequate optimization. Therefore, the fully dynamic AT is calculated by using network characteristics. The dynamic AT performs better than the AT because AT requires pre-deployment constants that need to be calculated empirically. In contrast, the dynamic AT derives all parameters from the network neighborhood size. In the case of an aggressive approach, AT and dynamic AT show similar results. However, in all other scenarios the dynamic AT gives a better result than the AT. Their proposed approaches reduce overhead by 20%–50% and energy by 50%. Further, in the case of blackhole attacks, the dynamic AT achieves 99% PDR in comparison with 33% for the default RPL. The network size for the case study was small for an IoT scenario; therefore, it requires a dense network evaluation. Moreover, the proposed approaches also require verification using a testbed.

Table 7 exhibits ContikiRPL routing attacks summary and future research direction.

3.2. Lightweight on-demand ad-hoc distance vector routing protocol

6LoWPAN is a low-power wireless mesh network where every node has its own IPv6 address, allowing it to connect directly to the Internet using open standards. The 6LoWPAN lightweight on-demand ad-hoc distance vector routing protocol — next generation (LOADng) is a routing protocol for low-power WSNs. LOADng is a simplified on-demand reactive routing protocol derived from the AODV routing protocol, and intended for use in IEEE 802.15.4 devices in 6LoWPANs and LLNs. The operation of LOADng is based on multi-hop routing between devices to establish and maintain on-demand routes in 6LoWPANs [93]. The RPL is a DV routing protocol that states how to make a DODAG with a defined OF and a set of metrics and constraints. The RPL routing protocol uses a proactive approach. The performance of RPL and LOADng routing protocols was compared in a home automation system [94] using the Contiki OS and the well-known simulator Cooja [95]. RPL shows shorter delay, less control overhead, and lower memory requirement than the LOADng routing protocol. The main limitation of the LOADng routing protocol is the delay in the route discovery process. During the route discovery process, outgoing packets are buffered, which may cause losses in memory-constrained devices. Moreover, flooding is highly energy inefficient; therefore, nodes may suffer from energy depletion. Another issue is related to the collisions of control messages due to flooding, which may lead to unnecessary retransmissions.

To extend the network lifetime and to optimize traffic over 6LoWPAN resources, the design of an efficient routing protocol is crucial. Yonga et al. [96] proposed an energy balancing adaptive routing protocol (EBAR). In the proposed protocol, they considered the dynamic traffic and constrained energy in 6LoWPAN. EBAR adaptively updates paths between different source–destination pairs and balances the energy in 6LoWPAN. Their results show that EBAR improves 6LoWPAN performance compared with LOAD and RPL. The main limitation of RPL is the traffic congestion near the root node of DODAG. Furthermore, RPL provides the longer paths than the available shorter paths.

3.3. Opportunistic routing

Most of the IoT devices with the limited resources employ the Contiki OS and support a wide variety of routing protocols. As compared with conventional routing protocols for conventional networks, routing protocols for the IoT devices have been designed with consideration of the limited energy and other resources. The concept of opportunistic routing has been used to perform energy-efficient path planning operation of the IoT devices. The opportunistic path planning algorithms have already been extensively employed by the IoT devices [97,98]. The Opportunistic routing protocols supporting the Contiki OS for IoT devices have also been designed as an extension to RPL.

3.3.1. Opportunistic RPL

The RPL is considered a standard path planning protocol for low-power networks supporting tiny IPv6. However, the scalability of the RPL has been enhanced with the introduction of opportunistic routing. As discussed in [99], the Opportunistic RPL (ORPL) has been developed by using the Contiki OS. In ORPL, the tree topology of the RPL has been extended in conjunction with opportunistic routing. The ORPL has been implemented in Contiki OS and various features such as Contiki MAC with many-to-one and one-to-one traffic settings have been analyzed. The ORPL scales well to networks with a large number of nodes, and information in this large network is propagated using the bitmaps and bloom filters. Duquennoy and Landsiedel [100] have studied the ORPL with Contiki OS for networks with irregular traffic patterns. The upward and downward routing of data helps in supporting the irregular traffic patterns. Usually, in ORPL, traffic is routed upward along the gradient, and in downward routing, it goes away from the root and then ultimately leads to the destination. This upward and

Table 7
ContikiRPL routing attacks proposal analysis and future research directions.

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Attacks studied	Future directions
[80]	–	✔	–	25	Control overhead, Energy, Power	Selective forwarding, Sinkhole, Hello flood, wormhole, clone ID, Sybil, Intrusion detection system.	Reduce network complexity. Requires testbed evaluation.
[81]	–	✔	–	8, 16, 24	True positive detection rate, Energy consumption, Energy overhead for various events, Packet overhead, Memory Consumption.	Intrusion and wormhole detection	Requires testbed evaluation, extension to make it a generalize IDS
[90]	–	✔	–	20	Average incoming and outgoing packet overhead, per node outgoing packet overhead, average end-to-end delay, total number of loops detected per node, total number of inconsistencies detected per node.	RPL version Number	Simple and memory efficient mitigation techniques, Realistic dense network and testbed evaluation
[91]	–	✔	–	44	Packet delivery ratio, Average delay, Control packet overhead, Average power consumption	RPL version number	Mitigation techniques and testbed evaluation
[92]	✔	✔	–	5, 10	Per node outgoing packet overhead, Total control message overhead by per node, Fixed and Adaptive threshold time-lines, Fixed and dynamic threshold time-lines, Delivery ratio, Energy Consumption for adaptive and dynamic threshold, Energy consumption for control message overhead.	DODAG inconsistency	Requires dense network and testbed evaluation

downward movement follows the ORPL routes data traffic using DODAG. This traffic pattern is then tested with Indriya testbed running the Contiki OS.

To achieve load balancing during the sensing operation, the tiny sensing nodes also use the ORPL with Contiki OS [101]. In this study, the ORPL load balance (ORPL-LB) has been proposed to perform the load balancing and to avoid the network congestion. The ORPL-LB has been implemented using the Contiki OS. With the help of wake-up intervals, analysis has shown that the ORPL-LB not only performs the load balancing but can also reduce the duty cycle of the worst nodes. The proposed scheme shows improvement in PDR and throughput as compared with other state-of-the-art schemes.

Smart grids have gained much attention as a smart solution for existing power grids [102,103]. Contiki OS with ORPL performs the energy-efficient routing for the advanced metering infrastructure (AMI). AMI is a smart meter for transporting the data to the service provider. Gormus et al. [104], have used the ORPL as the routing solution for efficiently performing the routing decisions for the AMI. The Contiki OS with ORPL in this distributed network topology has been designed for the mesh networks. In addition to the ORPL, the extended version of the ORPLx has also been proposed and tested with Contiki OS for the AMI to minimize the retransmission.

The proposed ORPL shows the improvement in terms of scalability and throughput as a smart metering solution. The reliability of the AMI networks has been improved further with the help of the ORPL in [105]. This Contiki OS supportive routing protocol selects the multiple relays to efficiently transmit the date to the destination nodes. As compared with the other existing routing approaches for the AMI networks, the ORPL requires less energy and memory and can work efficiently while conserving the scarce resources. ORPL has also been used to increase the utilization of the network resources for the AMI networks in [106]. To conserve the network resources and to improve the network performance, the anycast routing approach with ORPL has been tested with Contiki OS for AMI. This anycast routing approach also reduces the traffic and works well with limited resources for the mesh networks. The performance of the ORPL with the AMI has been compared with the RPL, and the proposed ORPL provides better performance in terms of throughput and link success rate.

Table 8 lists opportunistic RPL proposal analysis and future research direction.

3.3.2. Context-aware routing

In contrast to the ORPL, context-aware routing has also been well-supported by the Contiki OS in the IoT devices. When the context-aware routing [107,108] is implemented using the Contiki OS, the routing operation in the IoT seems to conserve more energy and can also support the delay sensitive traffic. The state-of-the-art context-aware routing schemes supported by the Contiki OS are described as follows.

Context-aware opportunistic resource-based routing protocol (CORB) [109] has been proposed for the WSNs that can support the intermittent connectivity of low-power tiny sensing nodes. Context-aware routing is usually used for mobile nodes, while the CORB has been specifically designed for stationary sensing nodes. The scarce resources and the context features are taken into consideration by the CORB for the intermittent connectivity. The proposed scheme with Contiki OS has been compared with the ripple routing protocol (RRP), and shows improved performance in terms of throughput and energy conservation.

As compared with CORB, which takes into consideration the stationary sensing nodes, the sensor context-aware routing (SCAR) [110] protocol has been designed for mobile nodes. The operation of the SCAR appears to be more efficient and smooth with the Contiki OS. This context-aware routing scheme has been specially designed for delay-sensitive traffic such as multimedia applications. To improve the reliability, multi-path routing has been introduced in the SCAR protocol that increases not just the reliability but also robustness for the IoT devices. The SCAR protocol has been specifically tested for its efficient routing operation with the Contiki OS. Pasztor et al. [111] have implemented the SCAR protocol in the Contiki OS. To efficiently transmit the delay-sensitive and time-critical data without much delay, the routing operation

Table 8
Opportunistic RPL proposal analysis and future research directions.

Citation	Mathematical analysis	Simulation	Testbed	Size of study	Performance metrics	Protocols used	Future directions
[99]	–	–	✔	Min = 3 nodes Max = 135 nodes	PDR Duty cycle Latency	ORPL RPL ORW CTP LWB	Impact on different applications such as smart cities and health care can be investigated
[100]	–	–	✔	137 nodes	Latency Duty cycle	ORPL	Trade-off between efficiency and correctness of the topology needs to investigate
[101]	–	–	✔	93 nodes	PDR Duty cycle Latency Average wake-up interval Network lifetime	ORPL RPL	Load balancing with different traffics
[102]	–	(Saturated and non-saturated Traffic)	–	30 nodes 70 nodes	PDR number of retransmission Cooperative overheads	ORPL RPL	Performance of ORPL in duty cycled mesh networks. Scheduling mechanisms that can reduce contention and interference within the AMI mesh networks.

should be more efficient and reliable. Therefore, the SCAR protocol has been tested extensively using the Contiki OS prior to its implementation in the real scenarios. The SCAR protocol shows the same functionality and performance with the theoretical analysis. Another demonstration of SCAR protocol functioning with Contiki OS has been discussed in [112]. For the evaluation purpose, the T-mote Sky sensing nodes with the Contiki OS have been used, and the data gathering and routing operation of this protocol have been analyzed. The main contribution of this real-time implementation of the SCAR protocol with the Contiki OS is to find the right carrier for the transmission of the data to the sink node. Usually, the selection of the sink depends on the buffer size of the sensing nodes, connectivity, and the residual energy of the sensing nodes.

The SCAR protocol has been used extensively for the tiny mobile sensor devices as discussed in [113]. The routing metrics of the SCAR protocol such as the mobility patterns and the residual energy of the tiny low-powered sensing nodes have been evaluated by implementing the SCAR protocol in the Contiki OS. The routing protocols with the Contiki OS have been implemented in the COOJA simulator. The SCAR protocol, in this case, has been compared with respect to the random choice-based dissemination.

3.4. OPPCAST routing protocol

Low-power wireless devices suffer greatly from cross-technology interference (CTI), e.g., WiFi, Bluetooth, and microwaves. CTI reduces the signal-to-interference-plus-noise ratio (SINR) and it causes high bit-error rates [114]. Mobashir [115] studied CTI in organized (educational institution, library, corporate offices) and unorganized environments (shopping malls, residential complexes). Mobashir proposed a synchronization mechanism, SYNCAST, and data-collection protocol, OPPCAST.

It is a challenge to maintain synchronization among synchronous transmitters because of software and hardware propagation delay. This may lead to the problem in scalability. Therefore, SYNCAST provide synchronization for dense networks with a large number of concurrent transmitters. Results in [115] show the reliability of SYNCAST in contrast to Glossy [116]. The OPPCAST routing protocol provides a solution to exploit spatial–temporal and channel diversity to achieve robust data collection in a dense urban environment. Mobashir implemented OPPCAST on Contiki OS and showed that it is highly robust, achieved a reliability of at least 98.55% during 255 h of experiments in four different urban deployments suffering from a large amount of unplanned CTI.

3.5. IPv6 neighbor discovery

Neighbor discovery (ND) defines the methods for addressing router, prefix, and parameter discovery, address auto-configuration, address resolution, neighbor unreachability detection, and duplicate address detection. The ND protocol uses the IPv6 protocol. However, ND uses multicast transmissions which make it inefficient for 6LoWPANs. Recently, the Internet Engineering Task Force (IETF) [117] proposed some amendments to the ND protocol to make it more suitable for 6LoWPANs. Seliem et al. [118] implemented and evaluated the proposed ND protocol as compared with the basic IPv6 ND protocol. The authors implemented the optimized ND protocol over Contiki OS version 2.6. The main features of optimization are router solicitation (RS) or router advertisement (RA) and neighbor solicitation (NS) or neighbor advertisement (NA). The RS/RA message optimization aims to reduce the overhead by avoiding the use of multicast flooding. However, the NS/NA message optimization focuses the interfaces between hosts and routers. The results show that the optimized ND protocol transmits fewer RS/RA messages as compared with the basic ND protocol.

3.6. Secure network (routing) protocols

Secure routing protects the packet from deliberate drops and alterations and unauthorized access [119]. Moreover, it aims to protect the routing infrastructure from various routing attacks. Confidentiality, authenticity, and access control mechanisms are proposed to secure the packet. However, these mechanisms cannot cope with packet drop attacks and attacks against the routing structure [120,121]. Hence, we analyze security aspects of the proposed schemes based on the above-mentioned attacks (see Fig. 4). Moreover, a limited capability of IoT devices in memory, energy, and computing power and Contiki OS puts restrictions on algorithms and protocols [122]. Hence, we consider these factors to discuss and present existing secure network protocols.

One of the most important research works is presented in [123], which is called ContikiSec. ContikiSec provides confidentiality, authentication, and integrity, which can be used selectively depending on the required security level in the network. To provide such security properties, the authors evaluate and compare existing security mechanisms in the modular sensor board (MSB-430) platform under Contiki OS. Security mechanisms are evaluated in terms of security level, memory usage, and energy consumption. ContikiSec works in three modes: ContikiSec-Enc, ContikiSec-Auth, and ContikiSec-AE. ContikiSec-Enc provides encryption only,

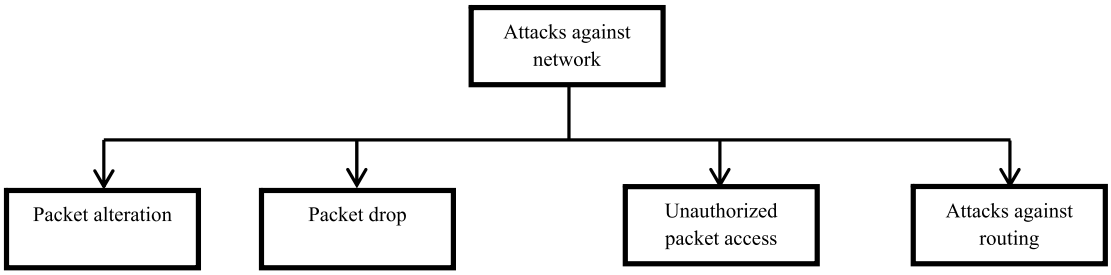

Fig. 4. Semantics of attacks against network protocols.

which is based on cipher block chaining–cipher stealing (CBC–CS) mode of operation with the advanced encryption standard (AES) [124] as the underlying block cipher. AES is found to be optimal among Skipjack [125], triple data encryption standard (3DES) [126], XTEA [127], RC5 [128], and Twofish [129] for confidentiality. ContikiSec-Auth gives more importance to authentication. Hence, AES cipher-based message authentication code (AES-CMAC) [130] is used for message authentication. Finally, ContikiSec-AE is designed for applications where the highest level of security is required. Hence, it provides confidentiality, authentication, and integrity. It uses offset codebook mode (OCB) mode [131] with AES as the underlying block cipher.

Although ContikiSec provides comprehensive research evaluating different security mechanisms on the MSB-430 platform, it would give more insight into the evaluation of security mechanisms of resource-constrained environment if it were performed on different boards. Moreover, the influence of compromised nodes in a secure network layer is not considered.

In [132], Raza et al. proposed the use of IPsec protocols to provide end-to-end secure communication in 6LoWPANs. Specifically, they proposed the use of two protocols of IPsec in 6LoWPANs for encryption and authentication, which are authentication header (AH) and encapsulating security payload (ESP). One of the main contribution of this work is to provide specification of IPsec for 6LoWPAN including definitions for AH and ESP extension headers. Moreover, the authors proposed the use of HC13 header compression to keep packet sizes reasonable in 6LoWPANs. HC13 defines context-aware header compression using IPHC for IP header compression and NHC for the next header compression. To evaluate the proposed scheme, it was implemented on Contiki OS. Evaluations were performed in terms of packet overhead, memory footprint, and energy overhead. Evaluations showed that AH and ESP fit in a tiny sensor node and they leave space for other applications. Moreover, it was shown that AES-CBC and AES-XBCMAC-96 [133] are optimal in terms of processing time and energy consumption. Although the proposed work contributes to the research field significantly, it would be better if effects of implemented mechanisms on routing performance were demonstrated. Moreover, as in previous work to encrypt the data, AES is used in this research work. However, key management issues are not covered in either research work.

As we discussed above, both schemes use AES, in which sender and receiver should have a symmetric key to encrypt and decrypt the data. However, in those research works, key distribution/exchange is not mentioned. In [134], Raza et al. proposed the use of elliptic curve cryptography (ECC) [135] as an asymmetric cryptographic system in the Diffie Hellman key exchange protocol. Moreover, they proposed a lightweight 6LoWPAN compression method for Internet Key Exchange version 02 (IKEv2). Currently RSA [136] is used for key exchange in IKE implementations, which is not suitable for resource-constrained IoT devices. Hence, the authors proposed the use of ECC for key exchange. A standardized ECC algorithm and NIST recommended elliptic curve and prime numbers were used in the ECC implementation. However, implementation details are not provided. To reduce the size of IEEE 802.15.4 link layer frames, the IKE header is compressed at the 6LoWPAN layer. The idea is to compress the IKE header along with UDP payload as it is UDP payload. The compressed IKEv2 header is recognized by the unique ID bits of 1101. Since the proposed research work was presented in a short paper, the implementation and description details were omitted.

In [137], Jutvik dealt with implementing some protocols of IPsec in Contiki OS. Hence, the author attempted to answer the following question: can IPsec and IKEv2 be implemented within the current hardware boundaries (in Contiki OS) while still being interoperable with other Internet hosts? After investigations and implementations, the answer was yes. Hence, IPsec and IKEv2 can be implemented in Contiki OS while providing interoperability with the vast majority of Internet hosts. As for IPsec ESP protocols, the following algorithms are implemented: AES-XCBC-MAC-96y, AES-Counter Mode (AES-CTR), and AES-CBC with 128 bit-key, NULL, 3DES-CBC, and HMACSHA1-96. In IKEv2 protocol implementation, the authors proposed the use of ECC as in [109], as it allows using smaller key sizes, without forsaking security. Overall, this research work contributes greatly to secure routing in Contiki OS implementing many protocols and algorithms of IPsec and demonstrating that it is feasible to use IPsec functionalities under Contiki OS. However, question of redundancy arises such as the need for a security mechanism in the network layer if security mechanisms are provided in the transport and data link layers. This is an important question that should also be answered by aforementioned research works. Table 9 portrays implemented and proposed network security mechanisms proposed for Contiki OS.

In brief, the proposed mechanism to secure network protocols focused on encryption and authentication mechanisms and their implementation, as demonstrated by Tables 9 and 10. Moreover, attacks such as packet drop and routing infrastructure are not considered in the proposed works. Since research on the secure network layer in Contiki OS is in an early stage, studies which focus on the combination of security mechanisms with routing algorithms have not yet been proposed.

4. Open research issues and recommendations

In this section, we briefly discuss challenges and open research issues.

ContikiRPL. Comprehensive studies are discussed in the previous section. The results indicated that there is always a difference in simulation and testbed outcomes. Thus, simulation requires further improvement to produce results closer to the testbed. Further, it lacks fine-grained timing. Most studies lack evaluation in a dense network topology to provide a detailed insight into protocol operation. Hence, it is of utmost importance as IoT constitutes billions of devices and any performance degradation hampers the

Table 9
Implemented and proposed security mechanisms to be used in the Contiki network layer.

Citation	Scheme	Encryption	Authentication	Implemented device
[123]	ContikiSec	AES	Cipher-based MAC	MSB-430
[132]	Securing Internet of Things with Lightweight IPsec	AES-CBC based on ESP protocol	MAC based on AH protocol	TmoteSky
[134]	Lightweight IKEv2	N/A	N/A	N/A
[137]	IPsec and IKEv2 for the Contiki operating system	AES-CBC	AES-XCBC-MAC-96	N/A

Table 10
Considered attacks and routing algorithms in the proposed schemes in the Contiki network layer.

Citation	Packet alteration	Unauthorized packet access	Packet drop	Routing infrastructure attacks	Considered routing algorithm
[123]	✔	✔	–	–	–
[132]	✔	✔	–	–	–
[135]	✔	✔	–	–	–
[137]	✔	✔	–	–	–

overall network performance. Many proposals suggest improvement in ContikiRPL; however, the effectiveness of these proposals is questionable until they are merged into the main Contiki OS tree. IoT may comprise many heterogeneous devices running different OSs and hence interoperability is another major concern. Extensive research is needed to make the protocols interoperable, scalable and energy efficient [138]. Recent improvements in IoT and cloud computing made it practically possible to use it in many applications such as health care [139] and smart life. The optimization of route timeouts and global repairs is required to make it energy efficient. The mobility is another intrinsic problem and in-depth work is required to identify the mobility patterns and likely trajectory of floating nodes. To sustain the network lifetime, further research needs to extend out to include dynamic distribution of load to prolong the network lifetime. The RPL OF requires additional research by considering fuzzy logic and ACO to make it more efficient. Trickle is another key area of research. It needs further research to reduce the control packet overhead and network convergence time, subsequently keeping power consumption low and improving energy efficiency.

ContikiRPL multicast. Further research on increasing transmission reliability, avoiding congestion, and incurring low delay is required. Moreover, more research is desirable on flexible and configurable forwarding, bi-directionality, delivery disorder avoidance, multi-sourcing, and dynamic group registration. Memory scarcity in IoT devices demands memory-efficient multicast proposals.

RPL host configuration. This area requires extensive research to assign dynamic address allocation with low memory footprint and achieve efficient and reliable top-down, bottom-up data traffic.

RPL routing attacks. ContikiRPL is prone to routing attacks such as selective forwarding, HELLO flooding, wormholes, clone ID, and Sybil. Moreover, it is also susceptible to version number attacks that result in control overhead and consequently increases delay and decreases PDR. RPL is also vulnerable to DODAG inconsistency attacks that cause congestion and high resource use. Therefore, IoT requires extensive research proposals that have to be lightweight but effective to mitigate these attacks.

Optimization of context-aware routing protocols for Contiki OS. The context-aware routing conserves more energy and support delay sensitive traffic while implemented on Contiki OS. However, an extensive research on the optimization of context-aware routing protocols with different topologies is required to gain the significant performance gain in real-life implementation.

Opportunistic routing protocol for dense lossy networks. Power consumption and reliable data delivery are important for IoT applications, especially for dense lossy networks. The concept of opportunistic routing has been used to perform energy-efficient path planning operation of the IoT devices. In dense lossy networks, the energy consumption during reception and idle listening can significantly affect the energy depletion. Therefore, scalable and energy-efficient opportunistic protocols need to be further investigated.

CTI-aware routing protocol. The number of IoT devices is increasing day-by-day. These tiny low-power wireless devices suffer greatly from CTI, which, as a result, increases the bit-error rate. Therefore, CTI-aware routing protocols are required for data collection in dense urban environments.

Secure network protocols. Research on secure routing protocols in Contiki OS is in its initial stage. There are a few research works that attempt to secure routing using existing security mechanisms. Future research directions and open research issues are as follows.

- **Intrusion detection and trust establishment.** Existing research focuses on cryptographic solutions only, which cannot provide full security alone. For example, data dropping attacks such as blackholes and grayholes or attacks directed at the routing infrastructure cannot be solved using cryptographic solutions. Moreover, compromised node problems might be serious problems in the network, which also cannot be mitigated with cryptographic solutions only. Hence, intrusion detection and trust establishment techniques can be used for these kinds of problems.
- **Key management.** One of the untouched research areas in secure routing in Contiki OS is key management. Key management is difficult problem in IoT security. The IKE protocol is used for manual or automatic key exchange in IPsec. Although manual key exchange methods are lightweight, they are less scalable and secure. On the other hand, automatic key exchange is heavier but scalable. Hence, research is required to discover optimal key exchange mechanisms which fit the requirements of IoT security and the resource capability of devices.
- **Impact of security mechanisms on routing performance and energy consumption.** It is important to consider the impact of the security mechanism on routing performance since it decides the performance of the whole network. Performance evaluation under different routing protocols, scenarios, and security mechanisms helps in finding optimal solutions. Moreover, energy is an important resource in IoT. Hence, one of the design goals of the security mechanism should be energy efficiency.
- **Interoperability of different security mechanisms.** Many types of IoT devices can use different types of security mechanisms, which might raise interoperability issues. Hence, standardizations of security mechanisms and their operation in IPsec in IoT is an important research direction.

5. Conclusion

The future IoT network size is enormous and it is rapidly modernizing everyday life. The IoT OS is the backbone to developing applications and protocols for heterogeneous devices. An effective method of finding devices to successfully deliver the data in a timely manner is crucial to the future IoT. The routing of data from source to sink is an essential part of IoT. Therefore, this survey paper presented an overview of Contiki OS-related routing protocols, issues, improvements, recent research, and future research directions. The aims of this survey were to lay down the foundation for researchers and professionals who are interested in working on IoT routing protocols. It also provides valuable insight on understanding past proposals, pros and cons of those proposals and recommendations for future directions. Traditional routing protocols are not suitable in LLNs. Therefore, a scalable, energy- and memory-efficient ContikiRPL has been proposed. To support the interoperability in IoT devices, RPL-Lite has also been proposed. Furthermore, to find an energy-efficient path, the concept of opportunistic routing has been introduced for IoT devices. Moreover, a discussion on context-aware routing has been presented that is well-supported by the Contiki OS in the IoT devices.

This survey paper has also discussed a secure network layer for the Contiki OS called ContikiSec. ContikiSec supports a configurable design, providing three security modes starting from confidentiality and integrity, and expanding to confidentiality, authentication, and integrity. In addition, the proposed security mechanisms and attacks in the Contiki network layer have been discussed.

We have thoroughly discussed all the proposals in each category, as well as issues and future directions. We finally presented comprehensive challenges, open research issues, and future research directions.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning, Korea), under the ITRC (Information Technology Research Center) support program (IITP-2016-R2718-16-0035) supervised by the IITP (National IT Industry Promotion Agency), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03030757) and the 2017 research grant of Kwangwoon University.

References

[1] M. Weiser, The computer for the 21st century, ACM SIGMOBILE Comput. Commun. Rev. 3 (3) (1999) 3–11.
[2] Contiki: The Open Source Operating System for the Internet of Things. [Online] Available: http://www.contiki-os.org/ (accessed on 03.07.17).
[3] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: Proc. 29th Annual IEEE International Conference on Local Computer Networks, Tampa, Florida, USA, Nov. 2004, pp. 455–462.
[4] N. Brouwers, K. Langendoen, P. Corke, Darjeeling, a feature-rich vm for the resource poor, in: Proc. International Conference on Embedded Networked Sensor Systems, ACM SenSys, Berkeley, CA, USA, Nov. 2009, pp. 169–182.
[5] S. Bocchino, S. Fedor, M. Petracca, PyFUNS: A python framework for ubiquitous networked sensors, in: Proc. European Conference on Wireless Sensor Networks, EWSN, Porto, Portugal, Feb. 2015, pp. 1–18.
[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, Tinyos: An operating system for sensor networks, in: Ambient Intelligence, Springer, Berlin, 2005, pp. 115–148.
[7] H.S. Kim, J. Ko, D.E. Culler, J. Paek, Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey, IEEE Comm. Surv. Tutor. 19 (4) (2017) 2502–2525 Fourthquater.
[8] A.S.A. Mohamed Sid Ahmed, R. Hassan, N.E. Othman, IPv6 neighbor discovery protocol specifications, threats and countermeasures: a survey, IEEE Access 5 (2017) 18187–18210.
[9] O. Hahm, E. Baccelli, H. Petersen, N. Tsiftes, Operating systems for low-end devices in the internet of things: a survey, IEEE Internet Things J. 3 (5) (2016) 720–734.
[10] L. Mirani, "Chip-makers are Betting that Moore's Law Won't Matter in the Internet of Things", 2014. [Online]. Available: http://qz.com/218514 (accessed on 03.07.17).
[11] M.O. Farooq, T. Kunz, Operating Systems for Wireless Sensor Networks: A Survey, Sensors 11 (6) (2011) 5900–5930.
[12] T. Reusing, Comparison of operating systems TinyOS and Contiki, Sens. Nodes-Oper. Netw. Appl. (SN) 7 (2012).
[13] D. Willmann, "Contiki - A Memory-Efficient Operating System for Embedded Smart Objects", Jan. 2009.
[14] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376 Fourthquarter.
[15] I.E. Bagci, M.R. Pourmirza, S. Raza, U. Roedig, T. Voigt, Codo: confidential data storage for wireless sensor networks, in: Proc. 2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems, MASS 2012, Las Vegas, NV, USA, Oct. 2012, pp. 1–6.
[16] S. Raza, L. Wallgren, T. Voigt, SVELTE: Real-time intrusion detection in the internet of things, Ad Hoc Networks 11 (8) (2013) 2661–2674.
[17] A. Ranjan, H.B. Sahu, P. Misra, A survey report on operating systems for tiny networked sensors, J. Adv. Res. Netw. Commun. Eng. 1 (2014) 1–12.
[18] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Comput. Netw. 38 (4) (2002) 393–422.
[19] J.N. Al-Karaki, A.E. Kamal, Routing techniques in wireless sensor networks: a survey, IEEE Wirel. Commun. 11 (6) (2004) 6–28.
[20] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, A taxonomy of wireless microsensor network models, ACM SIGMOBILE Mob. Comput. Commun. Rev. 6 (2) (2002) 28–36.
[21] S. Deering, R. Hinden, "Internet protocol, version 6 (IPv6) soecification", RFC 2460, Dec. 1998.
[22] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, "6L0WPAN: Transmission of IPv6 Packets over IEEE 802.15.4 Networks" RFC 4944, Sep 2007.
[23] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.
[24] T. Clausen, P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, Oct. 2003.
[25] C. Hedrick, "Routing information protocol," RFC 1058, Jun. 1988.
[26] C. Perkins, E. Belding-Royer, S. Das, "Ad Hoc on-demand distance vector (AODV) routing," RFC 3561, Jul. 2003.
[27] D.B. Johnson, Routing in ad hoc networks of mobile hosts, in: Proc. Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, Dec. 1994, pp. 158–163.
[28] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, Mobile Comput. 353 (1996) 153–181.
[29] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.P. Vasseur, R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012.
[30] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, "The Trickle Algorithm," RFC 6206, Mar. 2011.
[31] N. Tsiftes, J. Eriksson, A. Dunkels, Low-power wireless IPv6 routing with ContikiRPL, in: Proc. IPSN'10, Stockholm, Sweden, Apr. 2010, pp. 12–16.
[32] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, A. Dunkels, Making sensor networks IPv6 ready, in: Proc. 6th ACM Conf. Embedded Netw. Sensor Syst., Raleigh, North Carolina, USA, Nov. 2008, pp. 421–422.
[33] Y. Chen, J.-P. Chanet, K.M. Hou, RPL routing protocol a case study: Precision agriculture, in: Proc. First China-France Workshop on Future Computing Technology, Harbin, China, Feb. 2012, pp. 1–6.
[34] E. Ancillotti, R. Bruno, M. Conti, Reliable data delivery with the IETF routing protocol for low-power and lossy networks, IEEE Trans. Ind. Inform. 10 (3) (2014) 1864–1877.
[35] A.R. Chemate1, S.P. Pingat, Reliable data delivery using RPL, Int. J. Sci. Res. (IJSR) 4 (2) (2015) 646–648.
[36] H. Lamaazi, N. Benamar, M.I. Imaduddin, A.J. Jara, Performance assessment of the routing protocol for low power and lossy networks, in: Proc. 2015 International Conference on Wireless Networks and Mobile Communications, WINCOM, Marrakech, Morocco, Oct. 2015, pp. 1–8.
[37] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Mobile Computing, Kluwer Academic Publishers, 1996, pp. 153–181 (Chapter 5).
[38] X. Hong, M. Gerla, G. Pei, C. Chiang, A group mobility model for ad hoc wireless networks, in: Proc. 2nd ACM Int Workshop Model. Anal. Simul. Wireless Mobile Syst. Seattle, Washington, USA, Aug. 1999, pp. 53–60.
[39] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wirel. Commun. Mobi. Comput. (WCMC): Spec. Issue Mob. Ad Hoc Netw. Res. Trends Appl. 2 (5) (2002) 483–502.
[40] G. Lin, G. Noubir, R. Rajaraman, Mobility models for ad hoc network simulation, in: Proc. IEEE INFOCOM 2004, Hong Kong, Mar. 2004, pp. 454–463.

[41] K. Lee, S. Hong, S.J. Kim, I. Rhee, S. Chong, SLAW: self-similar least-action human walk, IEEE/ACM Trans. Netw. 20 (2) (2012) 515–529.
[42] M.M. Khan, M.A. Lodhi, A. Rehman, A. Khan, F.B. Hussain, Sink-to-Sink coordination framework using RPL: Routing protocol for low power and lossy networks, J. Sens. 2016 (2016) 11 Article ID 2635429.
[43] M. Banh, N. Nguyen, K.H. Phung, L. Nguyen, N.H. Thanh, K. Steenhaut, Energy balancing RPL-based routing for Internet of Things, in: Proc. 2016 IEEE Sixth International Conference on Communications and Electronics, ICCE, Ha Long, Vietnam, Jul. 2016, pp. 125–130.
[44] H.-S. Kim, H. Cho, H. Kim, S. Bahk, DT-RPL: Diverse bidirectional traffic delivery through RPL routing protocol in low power and lossy networks, Comput. Netw. 126 (2017) 150–161.
[45] A. Riker, M. Curado, E. Monteiro, Neutral operation of the minimum energy node in energy-harvesting environments, in: Proc. IEEE Symposium on Computers and Communications, ISCC, Heraklion, Greece. Sep. 2017, pp. 477–482.
[46] H.A.A. Al-Kashoash, H.M. Amer, L. Mihaylova, A.H. Kemp, Optimization-based hybrid congestion alleviation for 6LoWPAN networks, IEEE Internet Things J. 4 (6) (2017) 2070–2081.
[47] S.A. Alvi, F. u. Hassan, A.N. Mian, On the energy efficiency and stability of RPL routing protocol, in: Proc. International Wireless Communications and Mobile Computing Conference, IWCMC, Valencia, Spain, Jul. 2017, pp. 1927–1932.
[48] S.A. Alvi, A.N. Mian, On route maintenance and recovery mechanism of RPL, in: Proc. International Wireless Communications and Mobile Computing Conference, IWCMC, Valencia, Spain, Jul. 2017, pp. 1933–1938.
[49] Y. Tahir, S. Yang, J. McCann, BRPL: Backpressure RPL for high-throughput and mobile IoTs, IEEE Trans. Mob. Comput. 17 (1) (2018) 29–43.
[50] Z.A. Latib1, A. Jamil, N.A.M. Alduais, J. Abdullah, L.H.M. Audah, R. Alasi, Strategies for a better performance of RPL under mobility in wireless sensor networks, in: Proc. AIP, Sep. 2017.
[51] J. Ko, J. Eriksson, N. Tsiftes, S. Haggerty, A. Terzis, A. Dunkels, D. Culler, ContikiRPL and TinyRPL: Happy together, in: Proc. Workshop on Extending the Internet to Low power and Lossy Networks, IPSN 2011, Chicago, IL, USA, Apr. 2011, pp. 1–6.
[52] A. Parasuram, D. Culler, R. Katz, An Analysis of RPL Routing Standard for Low Power and Lossy Networks (Master Thesis), EECS Department, University of California, Berkeley, USA, 2016.
[53] K.D. Korte, A. Sehgal, J. Schönwälder, A study of the RPL repair process using ContikiRPL, dependable networks and services, in: Proc. 6th IFIP Proc. IFIP AIMS, in: Lecture Notes in Computer Science, vol. 7279, Springer International Publishing, 2012, pp. 50–61.
[54] S. Kuryla, J. Schönwälder, Evaluation of the resource requirements of SNMP agents on constrained devices, in: Proc. Managing the Dynamics of Networks and Services, Springer, 2011, pp. 100–111.
[55] A. Sehgal, V. Perelman, S. Kuryla, J. Schonwalder, Management of resource constrained devices in the Internet of things, IEEE Commun. Mag. 50 (12) (2012) 144–149.
[56] N. Khelifi, S. Oteafy, H. Hassanein, H. Youssef, Proactive maintenance in RPL for 6LowPAN, in: Proc. 2015 International Wireless Communications and Mobile Computing Conference, IWCMC, Dubrovnik, Croatia, Aug. 2015, pp. 993–999.
[57] P.O. Kamgueu, E. Nataf, T.N. Djotio, On design and deployment of fuzzy-based metric for routing in low-power and lossy networks, in: Proc. 40th IEEE Local Computer Networks Conference Workshops, New York, USA, Oct. 2015, pp. 789–795.
[58] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proc. of the 9th annual international conference on Mobile computing and networking, MobiCom '03, San Diego, Calif, USA, Sep. 2003, pp. 134–146.
[59] M. Belghachi, M. Feham, QoS routing RPL for low power and lossy networks, Int. J. Distrib. Sensor Netw. (IJDSN) 2015 (2015) 1–10.
[60] C. Blum, Ant colony optimization: Introduction and recent trends, Phys. Life Rev. 2 (2005) 353–373.
[61] J.-H. Kwon, H.-H. Lee, Y. Ko, J.J. Jung, E.-J. Kim, Ch. H. Seo, Queue state-based parent selection algorithm for large-scale wireless sensor networks, Sensors Mater. 29 (7) (2017) 977–982.
[62] N.M. Shakya, M. Mani, N. Crespi, SEEOF: Smart energy efficient objective function, in: Global Internet of Things Summit, GIoTS, Geneva, Switzerland, Jun. 2017, pp. 1–6.
[63] M.O. Farooq, . Cormac J. Sreenan, Kenneth N. Brown, T. Kunz, Design and analysis of RPL objective functions for multi-gateway ad-hoc low-power and lossy networks, Ad Hoc Networks 65 (2017) 78–90.
[64] D. Benson, R. Kinicki, A Performance Evaluation of RPL with Variations of the Trickle Algorithm (Bachelor thesis), WPI MQP, 2016.
[65] B. Djamaa, M. Richardson, Optimizing the trickle algorithm, IEEE Commun. Lett. 19 (5) (2015) 819–822.
[66] B. Ghaleb, A. Al-Dubai, E. Ekonomou, E-Trickle: Enhanced trickle algorithm for low-power and lossy networks, in: Proc. 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, Oct. 2015, pp. 1123–1129.
[67] B. Ghaleb, A. Al-Dubai, E. Ekonomou, B. Paechter, M. Qasem, Trickle-plus: Elastic trickle algorithm for low-power networks and Internet of Things, in: Proc. 2016 IEEE Wireless Communications and Networking Conference, Doha, Qatar, Apr 2016, pp. 1–6.
[68] M.B. Yassein, S. Aljawarneh, E. Masa'deh, B. Ghaleb, R. Masa'deh, A new dynamic trickle algorithm for low power and lossy networks, in: Proc. 2016 International Conference on Engineering & MIS, ICEMIS, Agadir, Morocco, Sep. 2016, pp. 1–6.
[69] M.B. Yassein, S. Aljawarneh, E. Masa'deh, A new elastic trickle timer algorithm for Internet of Things, J. Netw. Comput. Appl. 89 (2017) 38–47.
[70] M. Bouaziz, A. Rachedi, A. Belghith, EC-MRPL: An energy-efficient and mobility support routing protocol for internet of mobile things, in: Proc. 14th IEEE Annual Consumer Communications & Networking Conference, CCNC, Las Vegas, USA, Jul. 2017, pp. 19–24.
[71] H. Kharrufa, H. Al-Kashoash, Y. Al-Nidawi, M.Q. Mosquera, A.H. Kemp, Dynamic RPL for multi-hop routing in IoT applications, in Annual Conference on Wireless On-demand Network Systems and Services, WONS, Jackson, USA, Mar. 2017, pp. 100–103.
[72] W. Zhang, G. Han, Y. Feng, J. Lloret, IRPL: An energy efficient routing protocol for wireless sensor networks, J. Syst. Archit. 75 (2017) 35–49.
[73] G. Oikonomou, I. Phillips, Stateless multicast forwarding with RPL in 6LowPAN sensor networks, in: Proc. 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, Lugano, Switzerland, Mar. 2012, pp. 272–277.
[74] J. Hui, R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)," RFC 7731, Feb. 2014.
[75] G. Oikonomou, I. Phillips, T. Tryfonas, IPV6 multicast forwarding in rpl-based wireless sensor networks, Wirel. Pers. Commun. 73 (3) (2013) 1089–1116.
[76] K. Tharatipayakul, S. Gordon, K. Kaemarungsi, iACK: Implicit acknowledgements to improve multicast reliability in wireless sensor networks, in: Proc. 2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON, Nakhon Ratchasima, Thailand, May 2014, pp. 1–6.
[77] K.Q. AbdelFadeel, K. Elsayed, ESMRF: enhanced stateless multicast RPL forwarding for IPv6-based low-Power and lossy networks, in: Proc. 2015 Workshop on IoT Challenges in Mobile and Industrial Systems, IoT-Sys'15, ACM, New York, NY, USA, 2015, pp. 19–24.
[78] G.G. Lorente, B. Lemmens, M. Carlier, A. Braeken, K. Steenhaut, BMRF: Bidirectional multicast RPL forwarding, Ad Hoc Networks 54 (2017) 69–84.
[79] B. Peres, O. Goussevskaia, "MHCL: IPv6 Multihop Host Configuration for Low-Power Wireless Networks," http://arxiv.org/abs/1606.02674, 2016.
[80] L. Wallgren, S. Raza, T. Voigt, Routing attacks and countermeasures in the RPL-based Internet of things, Int. J. Distrib. Sens. Netw. 9 (8) (2013) 1–11.
[81] W. Xin-sheng, Z. Yong-zhao, X. Shu-ming, W. Liang-min, Lightweight defense scheme against selective forwarding attacks in wireless sensor networks, in: Proc. 2009 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Zhangijajie, China, Oct. 2009, pp. 226–232.
[82] S. Kaplantzis, A. Shilton, N. Mani, Y.A. Sekercioglu, Detecting selective forwarding attacks in wireless sensor networks using support vector machines, in: Proc. 2007 3rd International Conference on Intelligent Sensors. Sensor Networks and Information, Melbourne, Qld., Australia, Dec. 2007, pp. 335–340.
[83] M.A. Hamid, M.-O. Rashid, C.S. Hong, Routing security in sensor network: Hello flood attack and defense, in: Proc. IEEE ICNEWS 2006, Dhaka, Bangladesh, Jan. 2006, pp. 77–81.
[84] Y.C. Hu, A. Perrig, D.B. Johnson, Wormhole attacks in wireless networks, IEEE J. Sel. Areas Commun. 24 (2) (2006) 370–380.
[85] B. Parno, A. Perrig, V. Gligor, Distributed detection of node replication attacks in sensor networks, in: Proc. 2005 IEEE Symposium on Security and Privacy, S&P'05, Oakland, California, USA, May 2005, pp. 49–63.
[86] M. Conti, R. Di Pietro, L.V. Mancini, A. Mei, A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks, in: Proc. 8th ACM International symposium on Mobile ad hoc networking and computing, Montreal, Quebec, Canada, Sep. 2007, pp. 80–89.
[87] Z. Li, G. Gong, Randomly directed exploration: An efficient node clone detection protocol in wireless sensor networks. in: Proc. IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, Macau, China, Oct. 2009, pp. 1030–1035.
[88] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, in: Proc. First IEEE International Workshop on Sensor Network Protocols and Applications, Anchorage, Alaska, USA, May 2003, pp. 113–127.
[89] P. Pongle, G. Chavan, Real time intrusion and wormhole attack detection in Internet of things, Int. J. Comput. Appl. 121 (9) (2015) 1–9.

[90] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, J. Schönwälder, A study of RPL DODAG version attacks, in: Proc. of AIMS Conference, Brno, Czech Republic, Jun. 2014, pp. 92–104.
[91] A. Aris, S.F. Oktug, S.B.O. Yalcin, Version number attacks: In-depth study, in: Proc. 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, Apr. 2016, pp. 776–779.
[92] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, J. Schonwalder, Mitigation of topological inconsistency attacks in RPL-based Low-Power Lossy Networks, Int. J. Netw. Manag. 25 (5) (2015) 320–339.
[93] A.C. Martínez, T.H. Clausen, Implementation and Testing of LOADng: A Routing Protocol for WSN, (Bachelor Thesis), Universitat Politecnica De Catalunya, Paris, France, 2012, pp. 1–188.
[94] M. Vučinić, B. Tourancheau, A. Duda, Performance comparison of the RPL and LOADng routing protocols in a Home Automation scenario, in: Proc. 2013 IEEE Wireless Communications and Networking Conference, WCNC, Shanghai, China, Apr. 2013, pp. 1974–1979.
[95] [Online] Available: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja (accessed on 03.07.17).
[96] S. Yongan, S. Ziyu, L. Bin, An energy balancing adaptive routing strategy based on interference model in 6LoWPANs, Chin. J. Electron. 22 (3) (2013) 466–470.
[97] F. Singh, J.K. Vijeth, C.S.R. Murthy, Parallel opportunistic routing in IoT networks, in: Proc. 2016 IEEE Wireless Communications and Networking Conference, Doha, Qatar, Apr. 2016, pp. 1–6.
[98] I. Amdouni, C. Adjih, N. Aitsaadi, P. Muhlethaler, Experiments with ODYSSE: opportunistic duty cycle based routing for wireless sensor networks, in: Proc. 2016 IEEE 41st Conference on Local Computer Networks, LCN, Dubai, UAE, Nov. 2016, pp. 232–235.
[99] S. Duquennoy, O. Landsiedel, T. Voigt, Let the Tree Bloom Scalable Opportunistic Routing with ORPL, in: Proc. International Conference on Embedded Networked Sensor Systems, ACM SenSys 2013, Rome, Italy, Nov. 2013, pp. 1–14.
[100] S. Duquennoy, O. Landsiedel, Opportunistic RPL, in: Proc. EWSN 2013. Ghent, Belgium, Feb. 2013, pp. 1–4.
[101] M. Michel, S. Duquennoy, B. Quoitin, T. Voigt, Load-balanced data collection through opportunistic routing, in: Proc. 2015 International Conference on Distributed Computing in Sensor Systems, Fortaleza, Brazil, Jun. 2015, pp. 62–70.
[102] X. Fang, S. Misra, G. Xue, D. Yang, Smart Grid-the new and improved power grid: a survey, IEEE Commun. Surv. Tutor. 14 (4) (2012) 944–980 Fourth Quarter.
[103] V. Gungor, D. Sahin, T. Kocak, C. Buccella, C. Cecati, G.P. Hancke, Smart grid technologies: communication technologies and standards, IEEE Trans. Ind. Inf. 7 (4) (2011) 529–539.
[104] S. Gormus, F. Tosato, Z. Fan, Z. Bocus, P. Kulkarni, Opportunistic RPL for reliable AMI mesh networks, Wirel. Netw. 20 (8) (2014) 2147–2164.
[105] S. Gormus, Z. Fan, Z. Bocus, P. Kulkarni, Opportunistic communications to improve reliability of AMI mesh networks, in: Proc. 2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies, Manchester, UK, Dec. 2011, pp. 1–8.
[106] S. Gormus, M.Z. Bocus, Efficient cooperative anycasting for AMI mesh networks, in: Proc. 2013 IEEE Global Communications Conference, GLOBECOM, Atlanta, GA, USA, Dec. 2013, pp. 2661–2666.
[107] M. Musolesi, C. Mascolo, CAR: Context-aware adaptive routing for delay-tolerant mobile networks, IEEE Trans. Mob. Comput. 8 (2) (2009) 246–260.
[108] A. Betances, K.M. Hopkinson, M. Silvius, Context aware routing management architecture for airborne networks, IET Netw. 5 (4) (2016) 85–92.
[109] M. Elias, A. Khattab, K.M.F. Elsayed, CORB: Context-aware opportunistic resource-based routing for stationary wireless sensor networks, in: Proc. 2015 International Conference on Computing, Networking and Communications, ICNC, Garden Grove, CA, USA, Feb. 2015, pp. 166–170.
[110] C. Mascolo, M. Musolesi, SCAR: context-aware adaptive routing in delay tolerant mobile sensor networks, in: Proc. International conference on Wireless communications and mobile computing, IWCMC, Vancouver, British Columbia, Canada, Jul. 2006, pp. 533–538.
[111] B. Pasztor, C. Mascolo, M. Musolesi, Implementation of SCAR using Contiki, in: Proc. Contiki Workshop'07, Kista, Stockholm, Sweden, Mar. 2007, pp. 1–2.
[112] C. Mascolo, M. Musolesi, B. Pdsztor, Data collection in delay tolerant mobile sensor networks using scar, in: Proc. 4th Int Conf on Embedded Networked Sensor Systems, SenSys'06, Boulder, Colorado, USA, Oct.-Nov. 2006, pp. 1021–034.
[113] B. Pasztor, M. Musolesi, C. Mascolo, Opportunistic mobile sensor data collection with SCAR, in: Proc. 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, MASS 2007, Pisa, Italy, Oct. 2007, pp. 1–12.
[114] A. Hithnawi, S. Li, H. Shafagh, J. Gross, S. Duquennoy, CrossZig: combating cross-technology interference in low-power wireless networks, in: Proc. 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN, Vienna, Austria, Apr. 2016, pp. 1–12.
[115] M. Mobashir, Tackling self interference, cross-technology interference and channel fading in wireless sensor networks, in: Proc. SenSys'15, Seoul, South Korea, Nov. 2015, pp. 503–504.
[116] F. Ferrari, M. Zimmerling, L. Thiele, O. Saukh, Efficient network flooding and time synchronization with glossy, in: Proc. 10th Int'l Conf. Information Processing in Sensor Networks, IPSN'11, Chicago, IL, USA, Apr. 2011, pp. 73–84.
[117] S. Chakrabarti, Z. Shelby, E. Nordmark, Neighbor discovery optimization for IPv6 over low-power wireless personal area networks, 6LoWPANs, RFC 6775, Nov. 2012.
[118] M.A.M. Seliem, K.M.F. Elsayed, A. Khattab, Performance evaluation and optimization of neighbor discovery implementation over Contiki OS, in: Proc. 2014 IEEE World Forum on Internet of Things, WF-IoT, Seoul, South Korea, Mar. 2014, pp. 119–123.
[119] M. Amadeo, et al., Information-centric networking for the Internet of things: challenges and opportunities, IEEE Netw. 30 (2) (2016) 92–100 Mar.-Apr.
[120] F. Ishmanov, Y.B. Zikria, Trust mechanisms to secure routing in wireless sensor networks: current state of the research and open research issues, J. Sens. 2017 (2017) 1–16.
[121] K. Zhang, X. Liang, R. Lu, X. Shen, Sybil attacks and their defenses in the internet of things, IEEE Internet Things J. 1 (5) (2014) 372–383.
[122] A. Sehgal, V. Perelman, S. Kuryla, J. Schonwalder, Management of resource constrained devices in the Internet of things, IEEE Commun. Mag. 50 (12) (2012) 144–149.
[123] L. Casado, P. Tsigas, Contikisec: A secure network layer for wireless sensor networks under the contiki operating system, in: Identity and Privacy in the Internet Age, Springer Berlin Heidelberg, 2009, pp. 133–147.
[124] "(NIST), (ITL): Specification for the advanced encryption standard (AES)," Federal Information Processing Standards Publication 197 (FIPS PUB 197) Nov. 26, 2001, [Online] Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (accessed on 03.07.17).
[125] "Escrowed encryption standard (EES)," National Institute of Standards and Technology. Federal Information Processing Standards Publication 185, U.S. Department of Commerce, Feb. 1994.
[126] "Data encryption standard (DES)," National Institute of Standards and Technology. Federal Information Processing Standards Publication 46-3, U.S. Department of Commerce, Feb. 1994.
[127] D.J. Wheeler, R.M. Needham, TEA, a Tiny Encryption Algorithm, in: Lecture Notes in Computer Science, 1995, pp. 363–366.
[128] R.L. Rivest, The RC5 Encryption Algorithm, in: Lecture Notes in Computer Science, 1995, pp. 86–96.
[129] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, "TwoFish: A 128-bit block cipher," NIST AES Proposal, 15, Jun. 1998.
[130] J. Song, R. Poovendran, J. Lee, T. Iwata, The advanced encryption standard-cipher-based message authentication code-pseudo-random function-128 (AES-CMAC-PRF-128) algorithm for the Internet key exchange protocol (IKE), RFC 4615, Aug. 2006.
[131] P. Rogaway, M. Bellare, J. Black, T. Krovetz, OCB: A block-cipher mode of operation for efficient authenticated encryption, ACM Trans. Inf. Syst. Secur. 6 (3) (2003) 365–403.
[132] S. Raza, T. Chung, S. Duquennoy, D. Yazar, T. Voigt, U. Roedig, Securing Internet of Things with Lightweight IPsec, ABB Corporate Research, Västerås Sweden. Mar. 2011, pp. 1–26.
[133] S. Frankel, H. Herbert, "The AES-XCBC-MAC-96 algorithm and its use with IPsec," RFC 3566, Sep. 2003.
[134] S. Raza, T. Voigt, V. Juvik, Lightweight IKEv2: A key management solution for both the compressed IPsec and the IEEE 802.15. 4 security, in: Proc. IETF Workshop Smart Object Security, Paris, France, Mar. 2012, pp. 1–2.
[135] D. Hankerson, A. Menezes, S. Vanstone, Guide To Elliptic Curve Cryptography, Springer-Verlag, New York, NY, USA, 2004.
[136] B. Kaliski, J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0," RFC 2437, Oct 1998.
[137] V. Jutvik, "Ipsec and IKEv2 for the contiki Operating system," UPTEC IT, 2014, ISSN 1401-5749; 14 010.
[138] M. Tabassum, Md.A. Razzaque, Md.N.S. Miazi, M.M. Hassan, A. Alelaiwi, A. Alamri, An energy aware event-driven routing protocol for cognitive radio sensor networks, Wirel. Netw. 22 (5) (2016) 1523–1536.
[139] J.H. Abawajy, M.M. Hassan, Federated internet of things and cloud computing pervasive patient health monitoring system, IEEE Commun. Mag. 55 (1) (2017) 48–53.

Yousaf Bin Zikria is currently a PostDoctoral Fellow in the department of Information and Communication Engineering, College of Engineering, Yeungnam University, Gyeongsan-Si, South Korea. He has more than 10 years of experience in research, academia and industry in the field of Information and Communication engineering, and Computer science. He received his Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, Korea, in 2016. He obtained his M.S. degree in computer engineering from Comsats Institute of Information Technology (CIIT) Islamabad, Pakistan

in 2007. He received his B.S. in computer engineering from University of Arid Agriculture Rawalpindi (UAAR) in 2005. He was a research officer in horizon technology Pvt. Ltd Pakistan from 2007 to 2011. Later, he joined King Khalid University (KKU) Saudi Arabia as a lecturer from 2011 to 2012. His research interests include IoT, 5G, wireless communications and networks, opportunistic communications, wireless sensor networks, routing protocols, cognitive radio ad hoc networks, cognitive radio ad hoc sensor networks, transport protocols, VANETS, embedded system and information security. He is the editor FT/SI on Unlocking 5G Spectrum Potential for Intelligent IoT: Opportunities, Challenges and Solutions for IEEE Communications Magazine, Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques for Elsevier Future Generation Computer Systems (FGCS) and 5G Mobile Services and Scenarios: Challenges and Solutions for MDPI Sustainability. He is also serving as the reviewer of IEEE Communications, Surveys and Tutorials, IEEE Sensors Letters, IEEE ACCESS, IEEE IT Professional, Elsevier Future Generation Computer Systems, Elsevier Computer Standards and Interfaces, Springer The Journal of Supercomputing, Sage International Journal of Distributed Sensor Networks, and KSII Transactions on Internet and Information Systems. He is the recipients of the excellent paper award of ICIDB 2016 Conference and fully funded scholarship for Masters and Ph.D. He held the prestigious CISA, JNCIS-SEC, JNCIS-ER, JNCIA-ER, JNCIA-EX, and Advance Routing Switching and WAN Technologies certifications. orcid.org/0000-0002-6570-5306.

Muhammad Khalil Afzal (SM'16) received his B.S. and M.S. degrees in Computer Science from COMSATS Institute of Information Technology, Wah Campus, Pakistan in 2004 and 2007, respectively and his Ph.D. Degree from Department of Information and Communication Engineering, Yeungnam University, South Korea, in December 2014. He has served as lecturer from January 2008 to November 2009 in Bahauddin Zakariya University Multan Pakistan, and from December 2009 to June 2011 in King Khalid University Abha, Saudi Arabia. Currently, he is working as Assistant Professor in Department of Computer Science at COMSATS, Wah Cantt Pakistan. He is serving as a Guest Editor of Future Generation Computer Systems (Elsevier), IEEE ACCESS, Journal of Ambient Intelligence and Humanized Computing (Springer), **IEEE Communication Magazine** and reviewer for IEEE ACCESS, Computers and Electrical Engineering (Elsevier), Journal of Network and Computer Applications (Elsevier), FGCS, and IEEE transaction on Vehicular Technology. His research interest includes wireless sensor networks, ad hoc networks, Smart Cities, 5G, and IoT. orcid.org/0000-0002-6161-1310.

Farruh Ishmanov received his B.S. in Information Systems in 2007 from Tashkent State University of Economics, Uzbekistan. At this University, he studied and worked in the Multimedia Lab during the undergraduate years. He received his M.S. and Ph.D. degrees from the Department of Information and Communication Engineering, Yeungnam University, Korea, in 2009 and 2014, respectively. He was awarded Korean Government IITA Scholarship for pursing M.S. degree. In March 2015, he joined the Department of Electronics and Communication Engineering, Kwangwoon University, Korea, where he is currently an Assistant Professor. His research interests include resource management and security in wireless sensor networks and IoT. orcid.org/0000-0002-7378-8009.

Sung Won Kim received his B.S. and M.S. degrees from the Department of Control and Instrumentation Engineering, Seoul National University, Korea, in 1990 and 1992, respectively, and his Ph.D. degree from the School of Electrical Engineering and Computer Sciences, Seoul National University, Korea, in August 2002. From January 1992 to August 2001, he was a Researcher at the Research and Development Center of LG Electronics, Korea. From August 2001 to August 2003, he was a Researcher at the Research and Development Center of AL Tech, Korea. From August 2003 to February 2005, he was a Postdoctoral Researcher in the Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA. In March 2005, he joined the Department of Information and Communication Engineering, Yeungnam University, Gyeongsangbuk-do, Korea, where he is currently a Professor. His research interests include resource management, wireless networks, mobile networks, performance evaluation, and embedded systems. orcid.org/0000-0001-8454-6980.

Heejung Yu received his B.S. in radio science and engineering from Korea University, Seoul, Rep of Korea, in 1999 and his M.S. and Ph.D. in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea, in 2001 and 2011, respectively. He is currently an assistant professor with the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, Rep. of Korea. From 2001 to 2012, he was a senior researcher with ETRI, Daejeon, Rep. of Korea. He participated in the development of the IEEE 802.11 standardization, focusing on IEEE 802.11n, 11ac, and 11ah, to which he made technical contributions from 2003. His areas of interest include statistical signal processing and communication theory. Prof. Yu was the recipient of the Bronze Prize in the 17th Humantech Paper Contest and the Best Paper Award in the 21st Joint Conference on Communications and Information (JCCI) in 2011. orcid.org/0000-0001-8046-2376.