Performance Evaluation of Source Routing Minimum Cost Forwarding Protocol over 6TiSCH Applied to the OpenMote-B Platform

Anderson Rocha Ramos[1], Fernando J. Velez [1] and Gordana Gardašević [2]

[1] Instituto das Telecomunicações and Universidade da Beira Interior, DEM, Faculdade de Engenharia, 6201-001 Covilhã, Portugal
[2] Faculty of Electrical Engineering, University of Banja Luka, Banja Luka, Bosnia and Herzegovina
anderson.ramos@ubi.pt, fjv@ubi.pt,
gordana.gardasevic@etf.unibl.org

Abstract. The aim of this work is the development of Source Routing Minimum Cost Forwarding (SRMCF) protocol over IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), evaluating the performance of these protocols for the Internet of Things (IoT), and different healthcare, medical monitoring and urban mobility applications. To perform this evaluation, this work is making use of the OpenWSN project platform, which implements IEEE 802.15.4e in an open source environment. The evaluation process is also being done in the most recent version of the OpenMote-B platform. Another goal of this research is to give contribution to the investigation of the applicability of quality of service (QoS) applied to the IEEE 802.15.4e standard. In the present stage of development, the efforts are concentrated on the programming of the required code, and the adaptation of the OpenWSN stack. Experimental results have shown that the proposed protocol is capable of reducing Packet Loss Ratio (PLR) and energy consumption in comparison to the Routing Protocol for Low Power and Lossy Networks (RPL). In the next steps the team will also investigate the possibilities to explore long range routing techniques using the OpenMote platforms, together with xBee, LoraWAN, Raspberry PI and Arduino platforms.

Keywords: IEEE 802.15.4e, Minimum Cost Forwarding, Sensor Network, 6TiSCH.

1 Introduction

This work intends to implement SRMCF protocol over 6TiSCH and evaluating its performance by using the OpenMote B platforms [http://www.openmote.com/]. It is based on the results of a short-term scientific mission (STSM) from COST CA 15104 IRACON during 2017 that took place at University of Banja Luka as a joint activity with researchers from University of Beira Interior (UBI).

This work has been partially supported and funded by COST CA 15104, UID/EEA/500008/2013, CONQUEST (CMU/ECE/0030/2017) and ORCIP. The authors acknowledge the fruitful discussions with Fardin Derogarian and Dragan Vasiljevic.

A routing protocol is needed to establish end-to-end data delivery. Recent researches have addressed the development of energy efficient protocols with the aim of finding routes in the networks that minimize energy consumption in nodes with small energy resources [1]. SRMCF is an extension of Minimum Cost Forwarding protocols (MCF) and it is proposed to reduce energy consumption and PLR expending the lifetime of devices in a Wireless Sensor Network (WSN).

The main idea behind MCF is to find the path with minimum cost in a large sensor network [2]. By performing this task, these protocols can compute the effects of delay, throughput and energy consumption in communications between a node and the sink. The operation of MCF protocols usually comprises two phases [3]. The first phase consists of setting up the cost values to all nodes by broadcasting a message from the sink node, which allows other nodes to adjust their costs according to the received messaged. Next, the source will broadcast the intended message to its neighbors and the nodes that receive the message add their transmission costs. Finally, the node checks the cost in the packet and if the remaining cost is not sufficient to reach the destination, the packet is dropped. Nowadays, WSNs are applied in a variety of scenarios including healthcare, medical monitoring with the so-called wireless body sensor networks (WBSNs) or Medical BANs [1], and urban mobility applications.

Low-power WSNs use a radio transmission technology based on the IEEE.802.15.4-2006 standard, which defines a physical layer and a MAC layer to control the access to wireless medium. The fundamental principle of this standard has in it an intrinsic flaw because it requires wireless devices to keep listening, since they do not know when their neighbors will transmit a message. The IEEE.802.15.4e standard tries to deal with this flaw by altering the MAC protocol while preserving the physical layer [4]. The latter standard implements a schedule that tells the devices in the network when they should transmit, receive or go to sleep.

Power consumption in sensor nodes is mainly due to the presence of a radio transceiver which plays a decisive role as IoT devices become increasingly more present in everyday applications. Also, recent advances in the production and miniaturization of electronic circuits are causing an increase on the deployment of WSNs.

The OpenWSN project implements the IEEE.802.15.4e Time Slotted Channel Hopping (TSCH) protocol [5] and allows the use of IPv6 communication. This project is considered in the present work to evaluate SRMCF alongside the most recent version of the OpenMote B platform.

The rest of the paper is structured as follows. Section II gives an overview of SRMCF while the OpenWSN stack is analyzed in section III. Section IV addresses the process of development and integration of the proposed protocol with the OpenWSN stack. Section V describes the OpenMote platforms and parameters settings. Section VI presents the preliminary results. Finally, section VII draws conclusions and discusses the possibilities for future work.

2 Source Routing Minimum Cost Forwarding

SRMCF is classified as a reactive protocol, a class of protocols that avoid saving information about network topology in the devices. One example of these protocols is the Gossiping algorithm [6], in which messages forwarded from nodes are sent with some probability, to avoid overhead. In general, in reactive protocols there are two types of information traffic. First, there is the traffic between base station (BS) and sensor nodes (SN), when nodes send acquired information to the sink node and receive information from it. Next, there is traffic between sensor node, when these nodes get information data, topology, connections etc.

In SRMCF protocols, sensor nodes use the MCF protocol to send packets to the base node. However, in this protocol the SN needs to keep information about minimum cost path. The idea of SRMFC is that packets generated at the BS contain information about the path, so that SN can use the path information present in the header to route packets, such as in Dynamic Source Routing Protocol (DSR) [7]. The BS is required to have a routing table containing path information that is sent from itself to SN [8] by adding this information to the headers of the packets that are sent in unicast messages.

As a consequence of the SRMCF topology, packets coming from the BS and packets coming from SN have different routing algorithms and also different packet headers. Intermediate nodes are required to identify the appropriate routing scheme based on the type of the message [9]. Data transmission always uses unicast messages. This protocol also supports broadcast messages to perform cost advertisement and cost request. Cost advertisement is generated when BS starts up or when a SN acquires a new cost value, while all SN can broadcast cost request messages [6].

The operation of SRMCF protocol starts with a setup phase, when nodes define their cost to communicate with the BS and the routing table is created. The first part is similar to minimum cost forwarding back-off algorithm, where each node sets its own initial cost as ∞ and the cost to communicate with the BS is equal to 0 [3]. When a node receives a cost value, it compares the received cost with its own cost. If the received value is larger than its own value, the current node value is updated with the new value, which is then broadcasted. This process is executed until all nodes can set up their values to the minimum.

Every time a node needs to change its initial value, in the previous first phase, it sends a packet with its ID and addresses it to a near-node. This node adds its own address to the payload and sends this payload to the next node, until the message finally reaches the BS that will now be able to identify all the nodes in the minimum cost path by their IDs and store this information. Fig. 1 shows the described setup phase.

The protocol developed during the STSM that inspired the present work is also capable of performing link failure recover. To performer this task, the algorithm keeps a watchdog timer in which each node monitors the activity of its near-node.

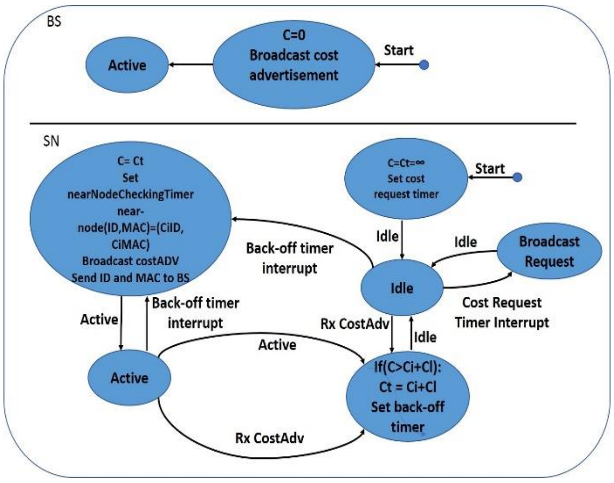

Fig. 1. Network setup phase in SRMCF protocols.

If a node does not respond during the requested time, its near node begins to broadcast cost request messages, so that the network can reestablish the routing information.

3 OpenWSN Protocol Stack

The OpenWSN project is an implementation of the IEEE.802.15.4e standard that allows users to investigate this standard in the context of low-power device-based networks. It is also heavily based on the concept of IoT, which means that it brings several IoT based standards, such as Routing Protocol for Low Power and Lossy Networks (RPL), Constraint Application Protocol (CoAP) and IPV6 over Low-Power Wireless Personal Area Network (6LoWPAN). All these standards and the fact that the project can be easily accessed by researchers make the OpenWSN the perfect tool for implementing and testing algorithms over the IEEE.802.15.4e standard. The structure of the OpenWSN stack is illustrated in Fig. 2.

The MAC sub-layer of the stack uses IEEE.802.15.4e TSCH. While this implementation preserves some characteristics of the IEEE.802.15.4 standard, such as Carrier Sense Multiple Access Collision Avoidance (CSMA/CA), which uses the concept of self-interference cancellation to detect collisions [10]. It also brings great improvements, such as the use of channel hopping, which allows persistent multi-path fading and increases interferences immunity [5]. The stack also brings the uRES protocol implemented in it. uRES is used to ensure that nodes can agree in message exchange that occurs in two-way communications, since the IEEE.802.15.4e standard by itself has no means to allow efficient data transport over multi-hop paths [5]. This part of the stack may be regarded as a Logical Link Control (LLC) layer. Also, there is a schedule mechanism implemented to allow nodes to identify actions to be taken based on the slots of the frame in which time is sliced in the IEEE.802.15.4e standard.

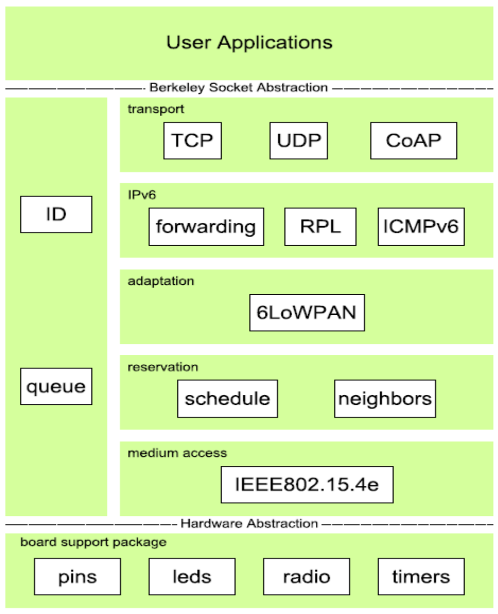

Fig. 2. OpenWSN protocol stack [11].

In the next layer, the stack implements 6LoWPAN as adaptation layer [12]. This layer allows the stack to compress the IPv6 headers. This procedure ensures that frames from the IEEE.802.15.4e standard have a length of 127 bytes, at most, and reduces the volume of information required for transmission [11]. Basically, the underling idea is to remove information that is not crucial from the headers while compressing other information, such as source and destination address. The stack also has a mechanism called Low-Power Border Router (LBR) which inflates the headers back to the standard size of the IPv6 headers [13].

RPL is implemented in the next level. It is responsible for the routing topology of the stack. This protocol is designed by the IETF ROLL working group and is especially developed to work with low-power networks. RPL uses the concept of Direct Acyclic Graph (DAG), where a node acts as the root of the topology [14]. Each node in the network receives a rank that keeps the position of that node in the network. These ranks increase in the downward direction. There are two types of messages within this concept. The first is the DAG Information Object (DIO), which is a control message that helps the protocol to build the DAG. The next message type is the Destination Advertisement Object (DAO), which is a unicast control message from nodes to its parents that allows intermediate nodes to recover information about the reverse path of the packet [15].

Finally, the OpenWSN stack implements CoAP [16]. This protocol is constructed as a header which is placed upon the User Datagram Protocol (UDP) and allows the implementation of applications that can be uploaded to motes to explore several features of hardware and software.

4 Protocol Implementation and Details

This section gives details about the process of code development and integration within the OpenWSN stack. The starting point was the analysis of the work previously developed, which was performed using the OMNeT++ simulator [17] and the C++ programing language. This code is composed of functions to deal with messages from upper layers, messages from lower layers, messages from SN to BS, messages from BS to SN, self-messages, control messages, and back-off timer.

The first step in this new phase was the development of the described functions in the C programming language. The OpenWSN stack uses the concept of modules. This means that every feature of the stack can be separately worked and then integrated with other layers, what helps to deal with the fact that C is not an object-oriented language. Considering this characteristic, the functions have been implemented in a source file and a header file was associated to hold prototype functions and variables to be accessed by other modules. Fig. 3 illustrates the placement of the SRMCF module inside the stack.

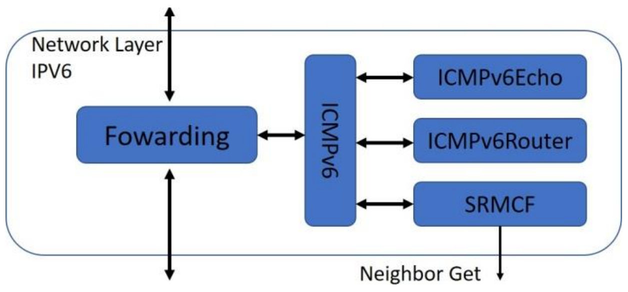

Fig. 3. SRMCF protocol inside OpenWSN stack.

The OpenWSN project is subdivided into firmware and software components [4]. The firmware sub-project contains code that run in the devices and it is where the main part of the implementation process was developed. The original RPL inside this sub-project can deal with message exchanges from BS to SN and from SN to BS in one source code. However, as in SRMCF BS and SN have different routing algorithms, it has been necessary to create two different source codes to be uploaded to each type of device. This fact creates the need for keeping two versions of the firmware, one to hold the routing algorithm to be applied in the SN, and another to deal with the BS.

The process of integration also requires modifications in several source files and modules of the stack. There is a need to create pointers and instances to the new module, and place them in certain modules while identifying functions that are specifically designed for RPL, since the goal is to replace the tasks performed by this protocol.

Once the source code is finished and compiled, one way to verify the exchange of message between nodes is using the Open Visualizer (OV). OV is part of the software sub-project and runs in the computer. It has several functionalities, such as the possibility to define a parent node, visualization of traffic between nodes and identifications of a node by its MAC address. Fig. 4 shows the Open Visualizer with two OpenMotes B connected to it. It is possible to observe how the OV identifies the motes by the MAC address from which one can obtain the IPv6 address.

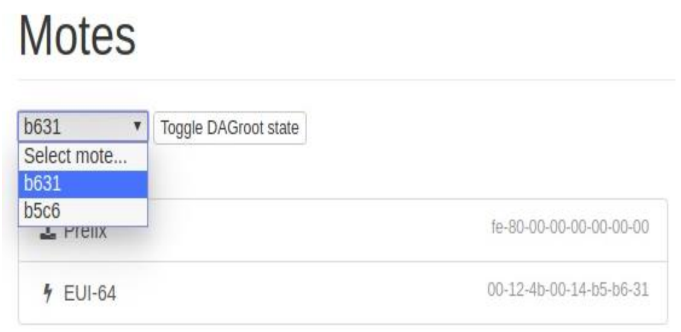

Fig. 4. OpenMote B devices connected to the Open Visualizer (identified by their MAC addresses).

5 Testbed and Settings

OpenMote is an open-source prototyping platform especially designed for the industrial Internet of Things and it is the latest generation of "Berkeley motes". The OpenMote-CC2538, which is probably the most well-known version of this platform, was, at first, composed of an entire ecosystem where the mote itself was the core of the system and it was followed by other devices, such as a base to allow communication during code development, an USB based interface designed for ease-of-use by end users and a base to provide power and some basic sensing capabilities through temperature and humidity sensors These motes features a 32 bit ARM Cortex-M3 microcontroller with a radio transceiver of -97 dBm sensitivity level and + 7 dBm power transmission [4].

The new versions of the OpenMote platform incorporate most of the above-mentioned features in one single device, what has made code development even more dynamic. These new motes have two radio transceivers that operate at 2.4 GHz and sub-GHz frequency bands. In the Telecommunications laboratory of Instituto de Telecomunicações/Universidade da Beira Interior there is a kit of ten units of these motes that are used in experimental research. These motes are shown in Fig. 5.

To evaluate the work and performance of the protocol developed, the code has been uploaded to the motes, with one mote receiving the code related to the BS whilst being connected to the host computer running Ubuntu 16.04.04 LTS via USB cable.

Fig. 5. OpenMote B devices from the Instituto de Telecomunicações (IT) of the Universidade da Beira Interior.

The first evaluation consists of sending data packets to a mote while waiting for the reply, to verify the correct transmission of information. Next, traffic is generated and recorded in the motes' network for different test cases. The traffic of information consists of payloads that are sent by the BS which waits for the reply. To perform the last test, firmware parameters are adjusted to create two scenarios of evaluation, as suggested in [5], as follows:

- Scenario 1: Slot frame length = 20 slots, number of active timeslots in slot frames = 14 slots.
- Scenario 2: Slot frame length = 9 slots, number of active timeslots in slot frames = 5 slots.

These test scenarios allow the measurement of PLR and throughput. The modifications in payload size and time of transmission require the recompilation of the firmware subproject in each test. As the proposed tests are similar to those performed in previous works found on the literature, using the OpenWSN stack and RPL [5] they allow for the comparison of the data obtained, what justify the chosen scenarios.

OpenWSN implements the IPv6 ping function, which makes possible to send a request to a specific device and observe the time it takes to reply. It is also possible to use a sniffer software, such as Wireshark, to analyze packet content, time of transmission and reply, as well as loss of packets.

6 Preliminary Analysis and Results

The network behavior depends on the number of hops. The results in this section considers a maximum of two hops. Tests with more hops have shown the occurrence of link failures that indicate the need to improve the failure recover mechanism.

The first phase of this analysis has consisted in verifying the correct behavior of the code. To perform this first evaluation, one mote received the algorithm for the BS and another mote received the algorithm for the SN. The BS was connected to the host computer. Next, the IPv6 ping was used to verify if the motes were able to receive packets and send responses. The early stages of code development had indicated that for the stack to work with the new version of the OpenMote B, the baud rate had to be adjusted in some of the modules.

Finally, test scenarios 1 and 2 have been applied. The original RPL of the OpenWSN stack has also been applied in order to evaluate the performance of the two protocols under the considered parameters. Results are shown below.

6.1 Test Scenario 1

This test considers hop distances of 1 and 2 with payload sizes of 20 bytes and 30 bytes, with 300 ms of inter-packet time. Tables 1 and 2 show PLR for the SRMCF while Tables 3 and 4 show throughput [kb/s] for test scenario 1.

Table 1. PLR for test scenario 1 and SRMCF

Hop Count	Payload size		
	20 bytes	30 bytes	40 bytes
1	0.11%	0.10%	0.51%
2	0.23%	0.67	0.70%

Table 2. PLR for test scenario 1 and RPL

Hop Count	Payload size		
	20 bytes	30 bytes	40 bytes
1	0.13%	0.10%	0.70%
2	0.40%	0.85%	0.88%

Table 3. Throughput [kb/s] for test scenario 1 and SRMCF

Hop Count	Payload size		
	20 bytes	30 bytes	40 bytes
1	0.60	0.68	0.82
2	0.65	0.80	0.88

Table 4. Throughput [kb/s] for test scenario 1 and RPL

Hop Count	Payload size		
	20 bytes	30 bytes	40 bytes
1	0.50	0.80	0.98
2	0.52	0.75	0.99

In scenario 1, it is possible to notice how PLR increases for higher hop counts. One aspect to be considered is how the OpenWSN firmware behaves with different inter-packet times, and how results can be analyzed when inter-packet times that are too small. For this test scenario, it can be observed that PLR is smaller for SRMCF in comparison with RPL. Also, higher PLR can be caused by mote queue overflow, since this queue has a limit, in this version of the stack.

6.2 Test Scenario 2

This test scenario considers hop distances 1 and 2 with payload sizes of 30 bytes and 300 ms of inter-packet time. Tables 5 and 6 show the PLR while Tables 7 and 8 show the throughput [kb/s] for the SRMCF and RPL protocols in this test scenario.

Table 5. PLR for test scenario 2 and SRMCF

Hop Count	Payload size = 30 bytes
1	0.30%
2	0.63%

Table 6. PLR for test scenario 2 and RPL

Hop Count	Payload size = 30 bytes
1	0.30%
2	0.70%

Table 7. Throughput [kb/s] for test scenario 2 and SRMCF

Hop Count	Payload size = 30 bytes
1	0.40
2	0.49

Table 8. Throughput [kb/s] for test scenario 2 and RPL

Hop Count	Payload size = 30 bytes
1	0.50
2	0.50

The analysis of the results presented on Tables 5 and 6 show that SRMCF and PRL present the same values of PLR for one hop.However, for distances of two hopes SRMCF presents a slightly better performance.

7 Conclusions and Future work

This work has just presented the analysis and practical results of the implementation of SRMCF in the OpenWSN stack. The performance analysis has shown the potential of the proposed protocol. It has also been possible to observe how the different characteristics of the studied stack may influence communication between motes. It was shown how slot frame length influences network performance, mainly because 6TiSCH schedule algorithm depends on the characteristics of active slots. However, the fact that the stack uses static scheduling makes the influence of characteristics such as slot frame length less significative in network performance. These facts indicate that, to maximize the performance of IoT base networks while using the OpenWSN stack, one must consider the size of the network and requirements for the transmission time.

As the proposed algorithm brings the potential for more efficient communications among wireless sensor motes, reducing packet collisions and energy consumption, it can enhance data traffic among IoT devices in healthcare, medical monitoring and urban mobility applications, whose deployment scenarios are characterized by the existence of interferers and obstacles that tend to increase the packet loss ratio. Besides, OpenWSN OS already implements IPv6, what brings advantages when one considers the context of IoT devices and the constant increasing in the number of devices connected to wireless networks.

The next steps in this research will be the development of functions more suitable to deal with link failure recovery and control data forwarding. Link recovery is a crucial part in SRMCF, because once motes in the network have suffered a failure, the network is required to recalculate path costs and send this new information to BSs, so that the cost table in it can be updated. The creation of a new graphical interface is also necessary so that one can execute simulations in the graphical environment. Once these features are fully implemented it will be possible to evaluate the performance of the proposed protocol in comparison to RPL implemented in the OpenWSN stack.

Another feature that can be explored in future work is the development of QoS upon the proposed stack by performing the necessary changes in the algorithm. The team will also make use of the developed protocol to explore long range communications. The main idea behind this part of the work is to use the OpenMote platform to interoperate with LoraWAN modules that allow long range communications, e.g., to control swarms of aerial or aquatic drones [18].

References

[1] F. J. Velez and F. Derogarian, *Wearable technologies and wireless body sensor networks for healthcare*, The Institution of Engineering and Technology, London, UK, 2019.

[2] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, no. 3. pp. 325–349, 2005.

[3] F. Ye, A. Chen, S. Lu, and L. Zhang, “A scalable solution to minimum cost forwarding in large sensor networks,” in *Proc.of Tenth International Conference on Computer. Communications and Networks (ICCCN)*, Scottsdale, Arizona, USA, Jan. 2001, pp. 304–309.

[4] T. Chang, P. Tuset-Peiro, X. Vilajosana, and T. Watteyne, “OpenWSN & OpenMote: Demo’ing a complete ecosystem for the industrial internet of things,” in *Proc. of 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking, (SECON),* London, UK, 1999.

[5] D. Vasiljević and G. Gardašević, “Performance evaluation of OpenWSN operating system on open mote platform for industrial IoT applications,” in *Proc.of 2016 International Symposium on Industrial Electronics (INDEL)*, Banja Luka, Bosnia Herzegovina, Nov. 2016, pp.1-6.

[6] C. Patra and N. Botezatu, “Effect of gossiping on some basic wireless sensor network protocols,” in *Proc. of 21st International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, 2017.

[7] S. Ahmad, I. Awan, A. Waqqas, and B. Ahmad, “Performance analysis of DSR & extended DSR protocols,” in *Proc.of 2nd Asia International Conference on Modelling and Simulation (AMS)* , Kuala Lumpur, Malaysia, May 2008, pp. 191–196.

[8] F. Derogarian, *Design of a body sensor network embedded in textiles for biomedical applications*, Doctoral dissertation, University of Porto, Porto, Portugal, 2015.

[9] F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “Analysis and evaluation of an energy-efficient routing protocol for WSNs combining source routing and minimum cost forwarding,” *Jornal of Ambient Wireless Communications and Smart Environmenst*, vol. 2017, no. 1, 2017.

[10] J. J. Garcia-Luna-Aceves, “Carrier-Sense multiple access with collision avoidance and detection,” in *Proc.of 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, Miami, Florida, USA, Nov. 2017, pp. 53–61.

[11] T. Watteyne et al., “OpenWSN: A standards-based low-power wireless development environment,” *Eur. Trans. Telecommun*., vol. 23, no. 5, pp. 480–493, 2012.

[12] J. Melorose, R. Perroy, and S. Careas, “Compression format for IPv6 datagrams in low power and lossy networks,” *Statew. Agric. L. Use Baseline 2015*, vol. 1, pp. 1–25, 2015.

[13] T. Winter et al., “RPL: IPv6 routing protocol for low power and lossy networks,” Work Progress), http//tools. ietf. org/html/draft-ietf-roll-rpl-19, no. July, pp. 1–164, 2011.

[14] D. Airehrour, J. A. Gutierrez, and S. K. Ray, “A Trust-Aware RPL routing protocol to detect blackhole and selective forwarding attacks,” *Aust. J. Telecommun. Digit. Econ.*, vol. 5, no. 1, p. 50, 2017.

[15] D. Wang, Z. Tao, J. Zhang, and A. A. Abouzeid, “RPL based routing for advanced metering infrastructure in smart grid,” in *Proc. of 2010 IEEE International Conference on Communications Workshops*, Capetown, South Africa, 2010.

[16] Z. Shelby, K. Hartke, C. Bormann, B. Frank, "Constrained application protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), 2013, [online] Available: http://tools.ietf.org/html/draft-ietf-core-coap-18.

[17] A. Köpke et al., “Simulating wireless and mobile networks in OMNeT++ the MiXiM vision,” in *Proc. of First International ICST Conference on Simulation Tools and Techniques for Communications Networks and Systems*, Marseille, France, Mar. 2008, pp.71-78.

[18] F.. J. Velez *et al*., "Wireless Sensor and Networking Technologies for Swarms of Aquatic Surface Drones," in *Proc. of IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Boston, MA, USA ,Sep. 2015.