International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 3, June 2021, pp. 2569~2576
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i3.pp2569-2576

History-based consistency algorithm for the trickle-timer with low-power and lossy networks

Firas A. Albalas, Haneen Taamneh , Wail E. Mardini
Department of Computer Science, Jordan University of Science and Technology, Jordan

Article Info

Article history:

Received Aug 18, 2020
Revised Oct 7, 2020
Accepted Dec 5, 2019

Keywords:

Internet of things
Lossy networks
Low power
Trickle timer
Wireless sensor networks

ABSTRACT

Recently, the internet of things (IoT) has become an important concept which has changed the vision of the Internet with the appearance of IPv6 over low power and lossy networks (6LoWPAN). However, these 6LoWPANs have many drawbacks because of the use of many devices with limited resources; therefore, suitable protocols such as the routing protocol for low power and lossy networks (RPL) were developed, and one of RPL's main components is the trickle timer algorithm, used to control and maintain the routing traffic frequency caused by a set of control messages. However, the trickle timer suffered from the short-listen problem which was handled by adding the listen-only period mechanism. This addition increased the delay in propagating transmissions and resolving the inconsistency in the network. However, to solve this problem we proposed the history based consistency algorithm (HBC), which eliminates the listen-only period based on the consistency period of the network. The proposed algorithm showed very good results. We measured the performance of HBC trickle in terms of convergence time; which was mainly affected, the power consumption and the packet delivery ratio (PDR). We made a comparison between the original trickle timer, the E-Trickle, the optimized trickle and our HBC trickle algorithm. The PDR and the power consumption showed in some cases better results under the HBC trickle compared to other trickle timers and in other cases the results were very close to the original trickle indicating the efficiency of the proposed trickle in choosing optimal routes when sending messages.

Corresponding Author:

Firas A. Albalas
Department of Computer Science
Jordan University of Science and Technology
Irbid-Jordan
Email: faalbalas@just.edu.jo

1. INTRODUCTION

Recently, IoT has become one of the most important fields emerging in the area of wireless sensor networks (WSNs). IoT also called the internet of everything or the industrial internet can be thought of as a variety of "things" or "objects" such as sensors, actuators, devices, and mobile phones, which by using unique addressing schemes, are able to interact with each other and with the end-users to reach common goals; creating a new form of communication between objects and people and between objects themselves [1]. This concept has been extended to cover a wide range of fields and applications such as transport applications, healthcare applications, utility applications, and most importantly Industrial applications [2]. The term, IoT was first released in 1999 by Kevin Ashton, who worked in radio frequency identification (RFID) and emerging sensing technology field, at Auto-ID center at the Massachusetts Institute of Technology (MIT) [3, 4].

WSNs represent a major factor in constructing the framework of IoT [5]. In WSNs, the sensor nodes exchange the data with a base station, which collects and manipulates the received data. These nodes have constrained processing abilities, constrained memory capabilities, and restricted power capabilities. These constraints make the network design and energy consumption very important challenges of the WSNs, therefore, to enable these devices to participate in the IoT standardization and to be fully integrated into the overall Internet; organizations and the research community have defined many architectures and protocols that are efficient in resource management and usage of constraint devices [6].

Hence, the concept of 6LoWPAN emerged, standing for the IPv6 over low power wireless personal area network. The 6LoWPAN protocol [7] is used to enable IPv6 packets to be sent to and received from over the standard (IEEE 802.15.4) which is universally recognized [8, 10].

Routing protocol for low-power and lossy networks (RPL) [11, 12] has many advantages in enhancing the overall network lifetime, one of the techniques used in RPL is the trickle algorithm [13]. The deployment of the trickle algorithm in IPv6, RPL along with other routing protocols makes it a good mechanism in controlling and maintaining the routing traffic frequency caused by a set of ICMPv6 control messages, such as DODAG information object (DIO), DODAG information solicitation (DIS), and the DODAG destination advertisement object (DAO), which are distributed among the nodes to deliver the right information to each node to make them work efficiently within the whole network [13].

The trickle timer's ability to control the flow of the RPL messages makes it a good mechanism in saving the power consumption of the network. Trickle's main problem was the short-listen problem, where the nodes have no time to hear each other sufficiently. This was later solved by adding the listen-only period mechanism. Such a period enables trickle to robustly solve the short-listen problem at the expense of increasing the latency of the network [14]. In this paper, we introduce a new mechanism as an optimization to the trickle timer algorithm that can decrease the latency with no additional overhead.

The rest of the paper is organized as: Section 2 presents a background to RPL. In section 3, we present the related work. Section 4, presents the methodology of the proposed algorithm and the experimental design. Section 5, discusses the results. Finally, Section 6 has the concluding remarks and the future work.

2. BACKGROUND AND RELATED WORK

Since RPL routing protocol is mostly used for LLNs, a lot of research has been done on it, and mainly on the trickle timer algorithm as an important component of the RPL protocol that controls the transmission of the control messages and saves energy [14-17]. An optimized trickle was proposed by Djamaa *et al.* [14], which only focused on the inconsistent state in its modification. The idea was to choose random time t, from the interval [0, Imin) rather than the interval [I/2, I). Therefore, it eliminates the listen-only period from the first interval until reaching a consistent state in the node, while the consistent state will choose t from the interval [I/2, I).

Ghaleb *et al.* [15] proposed another trickle algorithm called "E-Trickle" in which they show that the reason behind minimizing the suppression mechanism is not mainly the absence of the listen-only period, but it exists when the nodes ignore to receive control messages from the randomly chosen time t until the end of the interval. Therefore, they proposed a new solution for the short listen problem without using the listen-only period in the trickle algorithm by suggesting three modifications. The first modification was on the random time t that was in the standard trickle randomly chosen from the interval [I/2, I] and will be chosen from [0, I]. The second modification was the value of counter c that will be set to zero only at the beginning of the first interval Imin rather than at the beginning of each interval. However, this will set unequal intervals among the nodes, therefore, some nodes will send more messages than others. As a solution they used a formula for the value of the redundancy factor k, which will be set depending on the interval size, since the interval with short length will have priority to transmit more than the long intervals in case of unequal nodes intervals. However, the unfairness will still appear since the formula will be applied rarely. The results showed that the convergence time decreased under a different number of nodes, loss rates and different values of k. However, they did not notice any change in power consumption or packet delivery ratio (PDR) under a different number of nodes, loss rates or different values of k.

The trickle timer suffers from two main problems, it might consume high power to form a network in a short time, or it takes long convergence time with low power consumption. Therefore, Ghaleb *et al.* [16] suggested a new algorithm called trickle-plus. Their goal was to build a network with low power consumption and short convergence time. To handle the previous problem another version of trickle was proposed in [17], called the "New elastic trickle". The goal of his study was to improve the convergence time and the latency of the network. The researchers noticed a relation between the number of neighbors and the listen-only period, and came out with the result that when there are small numbers of neighbors the listen-only period becomes shorter and when the neighbors increase the period increases as well. In Table 1, we

summarize the recent related work on the trickle timer showing their methodology, the parameters they take into consideration, the strength of their proposed algorithm and the drawbacks. We will explain the proposed algorithm in more detail in the next section.

Table 1. Summary for some recent related works

Name of the Trickle Algorithm	Idea of the trickle	Performance Measurements	Strength of the Algorithm	Drawback of the Algorithm
Optimized trickle [14]	Choose random time t, from the interval [0, Imin) rather than the interval [I/2, I) only for inconsistent state	Power consumption convergence time PDR	- Better convergence time than the standard trickle - No additional overhead - PDR not affected.	- Consumes energy
E-Trickle [15]	Chose random t from [0, I]. Set c to zero only at the beginning of the first interval Imin Use a formula for the value of the redundancy factor k	Power consumption convergence time PDR	- Convergence time decreased under different number of nodes, loss rates and different values of k - Less consumed power because of reduction in the probability of collisions. - PDR not affected.	- Set unequal intervals among the nodes, therefore, some nodes will send messages more than others
Trickle-plus [16]	Permit the algorithm to start from min time, move ahead to a specific interval needed to be reached without moving during unnecessary intervals	Power consumption convergence time Load balancing	- Less convergence time	- Load balancing and high traffic overhead problems
Elastic Trickle [17]	They noticed a relation between the number of neighbors and the listen only period	Convergence time Traffic overhead Power consumption	- Better convergence time	- Increase in the traffic overhead - Increase in power consumption
Trickle-F [18]	Added the priority factor s to the original algorithm that gives higher priority of transmission for nodes who spend a long time waiting to transmit in the previous interval.	Power consumption convergence time Load balancing	- Better routes were discovered with less number of nodes and the same average of power consumption. - Solved the load balance problem	- Long Convergence time
Adaptive Trickle [19]	Make a dynamic trickle timer parameter that meets the needs when the power is very low and in safe mode	Power consumption convergence time	- Small values of Imin cause high power consumption and sending rates with low convergence - Higher values of Imin found better in terms of convergence - When the k value increases the convergence time decreases and the power consumption increases	- Used test bed methods with high cost
FL- Trickle [20]	The flexible trickle algorithm (FL-trickle) based on the transmission time and the intervals that specify the period of data delivery.	Convergence time Overhead Energy consumption Network lifetime.	- fixed transmission time to I/2 - decrease the delay of control messages transmission and to make the algorithm faster in terms of convergence time	- Short simulation time - Minimum number of nodes used

3. METHODOLOGY AND EXPERIMANTAL DESIGN

In this work, an optimized algorithm for the trickle timer was proposed called "history-based consistency (HBC) algorithm for the trickle-timer with low-power and lossy networks". The HBC algorithm was proposed to solve some of the problems that the standard trickle timer suffers from, in our case the listen-only period mechanism problem. The listen-only period in the standard trickle timer is the first half time of the interval where nodes keep listening and receiving messages from their neighbors having no ability to transmit, during the second half of the interval the nodes begin to transmit. However, using the listen-only period mechanism as a solution for the short-listen problem comes at the expense of increased delay in resolving the inconsistency. This will also affect the convergence time of the network [21]. In our proposed algorithm, we try to overcome this problem, by eliminating the time the node takes to wait until it can transmit which means we eliminate the listen-only period, however, this elimination is based on some conditions.

To understand the history-based consistency (HBC) algorithm we will explain how it works:

- First, it starts by setting the interval (I) of a node uniformly to a random value within the range [Imin, Imax], usually it sets the first interval to a value of (Imin).
- At the beginning of the interval (I), trickle resets the counter (c) to 0 for each node, which is used to keep track of the number of receiving consistent messages within the current interval (I).
- Then it assigns a random value to the time (t) within the range [I/2, I].
- Whenever a new consistent or redundant message is received, the counter (c) is incremented by one and the node is in a consistent state, and a new variable "hC" is incremented by one to count the number of consistent states of the node, and a variable called the "history_Counter" is also incremented by one to count the number of consistent and inconsistent states.
- Else if the message is old or new to the node's data, then the node will be in an inconsistent state, and a variable "hInc" is incremented by one to count the number of inconsistent states, and the "history_Counter" variable is also incremented by one. Then the interval (I) is reset to (Imin), and a new interval starts from step 2.
- In time (t), if the counter (c) is greater than or equal to the redundant constant (k), trickle suppresses the transmission; otherwise the message will be transmitted.
- When the interval (I) expires, trickle doubles the size of the interval (I = I*2), if the size exceeds the maximum interval (Imax), trickle sets (I) to (Imax) and re-executes the steps from step 2.
- All the above steps are repeated for each node until the variable "history_Counter" reaches the value of "10", therefore, the above steps are repeated for ten times. We have experienced more than one value (5, 7, 10, 15, 20), and based on experience the value ten gave us the best results.
- When reaching step 2 after the tenth time, the trickle resets the counter (c) to 0.
- Then trickle checks if the value of (hC) is greater than the value of (hInc) then the node was mostly in a consistent state, and so the time (t) will be chosen randomly from the range [0,I], so we eliminate the listen-only period and the node will transmit immediately without listening.
- However, if the value (hC) is smaller than the value (hInc) then the node was mostly in an inconsistent state, therefore, the time (t) will be chosen randomly within the range [I/2, I]. Hence, to keep track of the inconsistent messages we keep the listen-only period of the first half of the interval.

The flow of the HBC algorithm is shown in Figure 1. We modified the standard trickle timer algorithm where "+" means a new code was added to the standard algorithm, and each "-" means the old code was eliminated from the standard algorithm.

History – Based Consistency (HBC) algorithm
Input : Imin, Imax and K
Output: Deciding to transmit or suppress, to control the flow of the messages based on the consistency states of the nodes.

1. Initialization
   I ← Imin
2. Start a new Interval
   I ← I * 2; c ← 0
   if Imax <= I then I ← Imin End if
   - t ← random [I/2, I)
   + if history_Counter ==10
   + if hC >= hInc then + t ← random [0, I) else t ← random [I/2, I) End if
   + End if
3. Received a Consistent Transmission
   c ← c + 1
   + hC ← hC + 1
   + history_Counter ← history_Counter + 1
4. Received an Inconsistent Transaction
   I ← Imin
   + hInc ← hInc + 1
   + history_Counter ← history_Counter + 1
5. Random Timer Expires
   If c<= k then Transmit the message else Suppress the message end if

Figure 1. History-based consistency (HBC) algorithm

In our experiments we used the Cooja [22, 23] simulator, it provides a real environment to build our WSN with different types of motes, such as Tmote Sky, MicaZ and others, different sizes of networks could be implemented, and it also allows to upload the code to any mote to make it function different than other motes in the same network. The experiments consisted of 25, 50, 80, 100, 120 nodes, classified as sparse, moderate and dense network respectively. The nodes were placed in a random topology under (100 m*100 m) dimensions. Random topology means that the nodes are placed randomly with different distances between them.

The simulation for each experiment lasted 900 seconds (15 minutes) and each experiment was repeated 3 times with different seeds to get accurate results of the performance measurements as used in [16]. The transmission (TX range) and interference ranges (INF range) of each node were 50 m. The transmission success ratio (TX ratio) was 100% with different loss rate (RX ratio) values (0%, 20%, 40%, 60% and 80%). The RX value gives an indication of the loss rate of the sent messages, therefore, if we need to study the network under 60% loss rate, for example, the RX value must be 40%, because $100\% - 60\% = 40\%$. The objective function used was the default the minimum rank with hysteresis objective function (MRHOF) in selecting the preferred parent to build routes of high quality links. These configuration parameters of the simulation are clarified in Table 2.

Table 2. COOJA simulation parameters [16]

Parameters	Values
Operating system	CONTIKI 3.0 [24]
Simulator	COOJA
Nodes type	Tmote Sky
MAC/adaptation layer	ContikiMAC/6LowPAN
Routing protocol	RPL
Radio environment	Unit disk graph medium (UDGM)
Number of nodes	25, 50, 80, 100, 120
Simulation duration	900 seconds (15 minutes)
Data packet rate	60 seconds
Transmitting success ratio (TX)	100%
Simulator speed limit	Unlimited
Imin/Imax	$2^{12}/2^{20}$
Network topology	Random/Grid
Objective function	MRHOF

We used different topologies in our work, in these topologies, the network nodes are positioned in fully random topologies to analyze the convergence time, energy consumption and PDR in each of them, in order to evaluate the performance of the proposed trickle from different aspects:

3.1. Convergence time

The convergence time is the most important metric in our study, it indicates the time when the last node joins the network, and it is measured in seconds (s) or milliseconds (ms) and achieved once all the routing protocol information has been distributed to all the nodes participating in the network.

3.2. Power consumption

The power consumption represents the consumed energy of the nodes within some period of time to accomplish a specific task [25]. It is measured in (MW). The lower the consumed power is, the better the performance of the network. Figure 2 (in Appendix) shows the simulation scenarios.

4. RESULTS AND DISCUSSION

In our experiments, we compared the proposed algorithm (HBC) with three exited algorithms, which are the original trickle timer [14], the E-Trickle [15], and the optimized trickle [16]. In Figure 3 shows comparison between the four trickle algorithms has been made, but using the grid topology for the nodes. We can notice that here also the original trickle timer has the highest convergence time, and therefore it is the slowest in building up the DODAG. The lowest convergence time was given by our proposed HBC trickle algorithm with an average improvement of 48.38% compared to the original trickle algorithm, and the E-Trickle gave better results than the optimized for all the node densities.

In Figure 4 we did the same experiments, but for random topology, and it is clear that the Power consumption for all trickle algorithms was very similar to the original trickle. As explained before, the similarity in power consumption indicates the efficiency of the proposed trickle algorithms in choosing optimal routes just as the original trickle but in less time. We also notice that for 80 nodes the HBC trickle gave slightly better results. It is also clear that increasing the number of nodes increases the average energy consumption; this is resulted by the increment of node neighbors, which increases the traffic overhead and the probability of collisions, which as a result increases the power consumption by retransmission.

In Figure 5 we show a comparison between the four algorithms: the original, the optimized, the E-Trickle and our proposed HBC algorithm, in terms of Power consumption in a grid topology. It was noticed that for 50 and 25 nodes the power consumption was very similar to the original trickle. The fact that

they are identical in terms of power consumption indicates that the proposed algorithms were very efficient in choosing optimal routes just as the original trickle but in less time. We also noticed that for 80, 100 and 120 nodes the HBC trickle consumes slightly less energy. This is because when the number of nodes increases in the same area size, the communication becomes faster and the same message is sent from more than one neighbor, causing the counter c value to become more than the value of k in a faster time. It is also clear that increasing the number of nodes increases the average energy consumption; this is resulted by the increment of node neighbors, which increases the traffic overhead and the probability of collisions, which as a result increases the power consumption by retransmission.

In Figure 6, we did the same experiments but for random topology, and it is clear that the Power consumption for all trickle algorithms was very similar to the original trickle. As explained before, the similarity in power consumption indicates the efficiency of the proposed trickle algorithms in choosing optimal routes just as the original trickle but in less time. We also notice that for 80 nodes the HBC trickle gave slightly better results. It is also clear that increasing the number of nodes increases the average energy consumption; this is resulted by the increment of node neighbors, which increases the traffic overhead and the probability of collisions, which as a result increases the power consumption by retransmission.

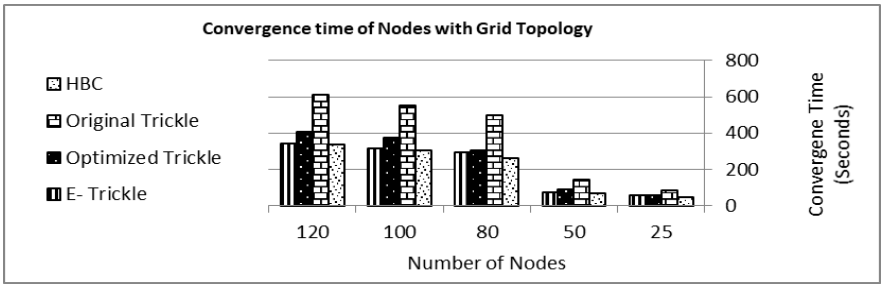

Figure 3. A comparison of the convergence time for (25, 50, 80, 100 and 120) nodes, between the original trickle, the optimized trickle, the E-Trickle and the HBC trickle, for the random topology.

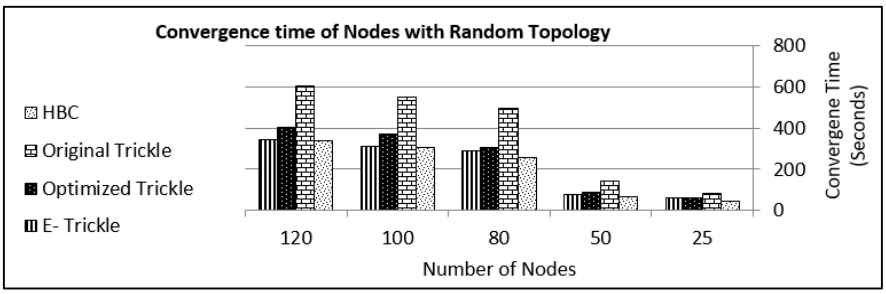

Figure 4. A comparison of the power consumption of (25, 50, 80, 100 and 120) nodes, between the original trickle, the optimized trickle, the E-Trickle and the HBC trickle, for random topology

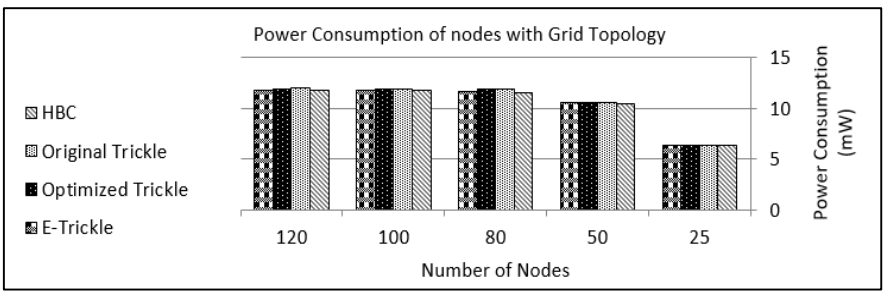

Figure 5. A comparison of the power consumption for (25, 50, 80, 100 and 120) nodes, between the original trickle, the optimized trickle, the E-Trickle and the HBC trickle, for the grid topology

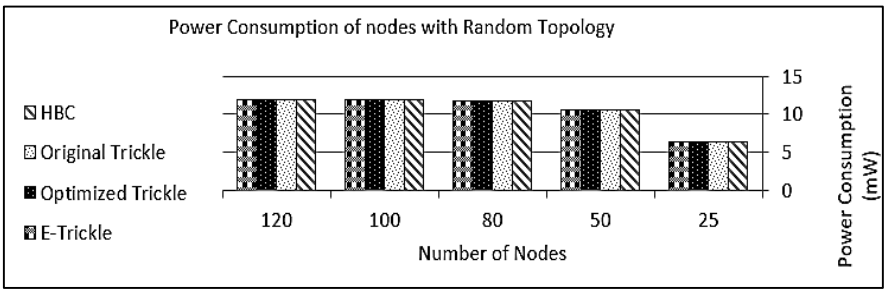

Figure 6. A comparison of the power consumption for (25, 50, 80, 100 and 120) nodes, between the original trickle, the optimized trickle, the E-Trickle and the HBC trickle, for random topology

5. CONCLUSION

In our work, we proposed an optimized trickle timer algorithm (HBC). And it mainly focused on handling the listen-only period based on the consistency and inconsistency of the network. We tested the algorithm under three measurement metrics which are the convergence time of the network and the power consumption. We compared our work with the original trickle, the optimized trickle and the E-Trickle algorithms, the simulation result showed that our proposed algorithm gave better results in terms of the convergence time of the network with an average improvement of 48.38% for random topologies. As for the power consumption the proposed algorithm gave the same results as the original algorithm for medium density networks, but for high density networks it even gave slightly better results

The same experiments and comparison for grid topology for different loss rates. The best performance was always gained when the loss rate was low. In terms of convergence time, the HBC gave an average improvement of 49.14%. In terms of power consumption, the results were in most cases better for the proposed trickle.

APPENDIX

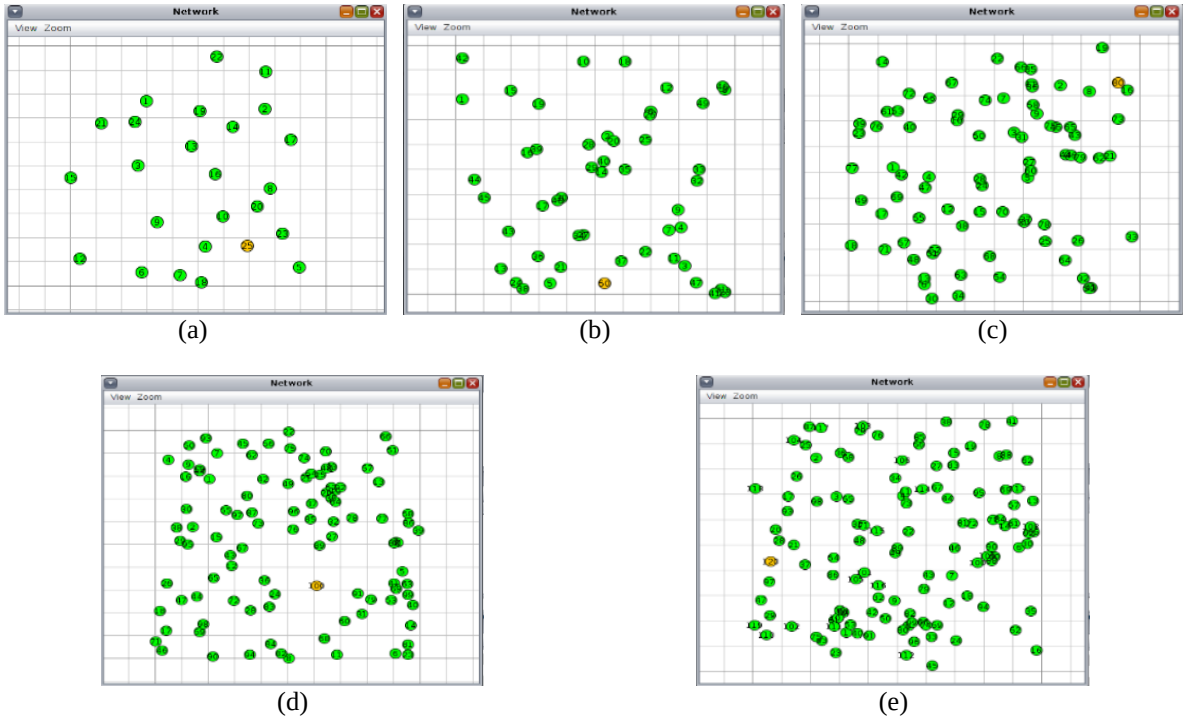

Figure 2. Simulation scenarios, (a) Random topology with 25 nodes, (b) random topology with 50 nodes, (c) random topology with 80 nodes, (d) random topology with 100 nodes, (e) random topology with 120 nodes

REFERENCES

[1] Hanane Lamaazi, et al., "Challenges of the Internet of Things: IPv6 and Network Management," *Proceedings of the 8th International IEEE Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 328-333.

[2] Iera A., Floerkemeier C., Mitsugi J. and Morabito G., "The internet of things," *IEEE Wireless Communications*, vol. 17, pp. 8-9, 2010.

[3] Gubbi J., Buyya R., Marusic S., Palaniswami M., "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[4] Ashton K., "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97-114, 2009.

[5] Qasem M., Al-Dubai A. and Yassien M. B., "A Dynamic power tuning for the constrained application protocol of Internet of Things," *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 2015, pp. 1118-1122.

[6] Fischione C., Park P., Di Marco P. and Johansson K. H., "Design principles of wireless sensor networks protocols for control applications," *Wireless Networking Based Control*, pp. 203-238, 2011.

[7] Hui, J., and Pascal Thubert, "Compression format for IPv6 datagrams in low power and lossy networks (6LoWPAN)," *draft-ietf-6lowpan-hc-15 (work in progress)*, 2011.

[8] Lee I and Lee K., "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431-440, 2015.

[9] Winter T., et al., "RPL: IPv6 routing protocol for low-power and lossy networks," *RFC 6550, Internet Engineering Task Force RFC 6550*, pp. 1-157, 2012.

[10] "RFC 4919-IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," [Online]. Available: https://tools.ietf.org/html/rfc4919.

[11] Musaddiq, A., Zikria, Y. B., Zulqarnain, Kim, S. W., "Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network," *EURASIP Journal on Wireless Communications and Networking*, no 21, pp. 1-23, 2020.

[12] Wang, F., Babulak, E., Tang, Y., "SL-RPL: Stability-aware load balancing for RPL," *Transactions on Machine Learning and Data Mining*, vol. 13, no. 1, pp. 27-39, 2020.

[13] Zhang T. and Li X., "Evaluating and analyzing the performance of rpl in contiki," *Proceedings of the first international workshop on Mobile sensing, computing and communication, ACM*, 2014, pp. 19-24.

[14] Djamaa B. and Richardson M., "Optimizing the Trickle Algorithm," *IEEE Communications Letters*, vol. 19, no. 5, pp. 819-822, 2015.

[15] Ghaleb B., Al-Dubai A. and Ekonomou E., "E-Trickle: Enhanced Trickle Algorithm for Low-Power and Lossy Networks," *Proceedings of the 14th IEEE International Conference on Ubiquitous Computing and Communications (IUCC)*, pp. 1123-1129, 2015.

[16] Ghaleb B., Al-Dubai A., Ekonomou E., Paechter B., Qasem M., "Trickle-Plus: Elastic Trickle Algorithm for Low-Power Networks and Internet of Things," *IEEE Wireless Communications and Networking Conference WCNC, IEEE Workshop on Mobile Edge Computing and IoT*, 2016, pp. 1-7.

[17] Yassein M. B., Aljawarneh S. and Masadeh E., "A new elastic trickle timer algorithm for Internet of Things," *Journal of Network and Computer Applications*, vol. 89, pp. 38-47, 2017.

[18] Vallati C. and Mingozzi E., "Trickle-F: Fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol," *2013 Sustainable Internet and ICT for Sustainability (SustainIT)*, Palermo, 2013, pp. 1-9.

[19] Meyfroyt T., Stolikj M., Lukkien J., "Adaptive Broadcast Suppression for Trickle-Based Protocols," *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, MA, 2015, pp. 1-9.

[20] Lamaazi, H. and Benamar, N., "RPL Enhancement Based FL-Trickle: A Novel Flexible Trickle Algorithm for Low Power and Lossy Networks," *Wireless Personal Communications*, vol. 110, pp. 1403-1428, 2020.

[21] O. Shahryrai, H. Pedram, V. Khajehvand and M. Fooladi, "Energy-Efficient and delay-guaranteed computation offloading for fog-based IoT networks," *Computer Networks*, vol. 182, pp. 45-62, 2020.

[22] Dunkels A., Gronvall B. and Voigt T., "Contiki-a lightweight and flexible operating system for tiny networked sensors," *29th Annual IEEE International Conference on Local Computer Networks*, Tampa, FL, USA, 2004, pp. 455-462.

[23] "Cooja Contiki-os.org," 2019. [Online]. Available: http://www.contiki-os.org/start.html#start-cooja.

[24] "Contiki: The Open Source Operating System for the Internet of Things," *Contiki-os.org*, 2018. [Online]. Available from: http://www.contiki-os.org.

[25] Albalas F., Mardini W. and Al-Soud M., "Aft: Adaptive fibonacci-based tuning protocol for service and resource discovery in the internet of things," *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, 2017, pp. 177-182.