IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 July 2022, accepted 24 July 2022, date of publication 5 August 2022, date of current version 11 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3196693

RESEARCH ARTICLE

Improved Trickle Algorithm Toward Low Power and Better Route for the RPL Routing Protocol

SSU-TING LIU AND SHENG-DE WANG, (Member, IEEE)
Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
Corresponding author: Sheng-De Wang (sdwang@ntu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant 109-2221-E-002-145-MY2.

ABSTRACT The IPv6 Routing Protocol for Low Power and Lossy networks (RPL) is a routing protocol standardized by the Internet Engineering Task Force. According to the specification of the protocol, the Trickle algorithm is adopted for the dissemination of route construction information among nodes in such networks. Since the algorithm is originally designed for code propagation and maintenance in Wireless Sensor Networks, when using the algorithm for the route formation of the network there exist some problems, such as the route convergence time, the fairness issue among nodes, and the amount of power consumption. Therefore, the paper proposes an improved Trickle algorithm, namely FI-Trickle, by taking a new approach to simultaneously reduce the unfairness among nodes and the power consumption and to improve the packet delivery ratio of the network. The performance of FI-Trickle is verified via simulation with extensive experiments over various network sizes, interference conditions, and network topologies. For comparison purposes, the extensive experiments are applied not only to FI-Trickle, but also to various Trickle variants such as Trickle, Trickle-F, I-Trickle, and Drizzle. The simulation results show that FI-Trickle can use less power to achieve a similar fairness level as compared to Trickle-F. Also, in terms of the packet delivery ratio, as the network size grows, FI-Trickle can increasingly outperform Trickle-F, Trickle, Drizzle, and I-Trickle by up to 1%, 3%, 3%, and 4%, respectively. This result implies that FI-Trickle can be a better Trickle candidate for the dissemination of RPL messages for large-scale networks.

INDEX TERMS Low-power and lossy networks (LLNs), power consumption, packet delivery ratio, RPL, trickle algorithm.

I. INTRODUCTION

The Trickle algorithm was first introduced in [1] for code propagation and maintenance in Wireless Sensor Networks. It has been proved that the algorithm can scale well with the network density and can rapidly disseminate packets with a low cost. For these features, the algorithm has gained popularity in recent years and has been standardized as RFC 6206 in [2] by the Internet Engineering Task Force (IETF). Particularly, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) utilizes the algorithm to disseminate route construction information among nodes in the network as stated in RFC 6550 [3].

The efficiency of the Trickle algorithm is based on two mechanisms: an adaptive sending rate and a suppression mechanism. The underlying consistency model of the algorithm guides the transmission policy of the nodes. When a node encounters an inconsistency, the node will communicate rapidly to resolve the situation. When a node does not encounter an inconsistency, the node will slow down its communication rate exponentially and the communication rate can go down to a minimum rate if the node keeps staying in a consistent state. While staying in a consistent state, if a node receives more than a certain number of packets during a period of time the transmission of the node will be suppressed. Instead of flooding packets in the network, the algorithm guarantees that nodes in the network can become consistent with an appropriate number of packets.

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

Fundamentally, the Trickle algorithm does not include a load balancing mechanism. However, the load distribution of RPL messages is crucial to the performance of the network. Some studies have shown that improving fairness among nodes with respect to the dissemination of RPL messages can enhance the route formation of the network [9], [10]. In addition to fairness, more RPL messages in the network can support more robust routes. However, a greater number of RPL messages can cause the nodes to increase their power consumption as well. Therefore, this paper proposes an improved Trickle algorithm that aims for the following goals:

- First, the proposed algorithm can achieve fairness among nodes with a new approach and can simultaneously increase the performance of the network in terms of the packet delivery ratio (PDR).
- Second, the proposed algorithm can also lower the number of RPL messages in the network without degrading the PDR of the network.

The rest of the paper is organized as follows: Section II gives a detailed description of the Trickle algorithm and the RPL protocol. Section III presents an overview of related work. Section IV describes the proposed algorithm, FI-Trickle. Section V gives a detailed description of the experiment setting and a detailed analysis of the simulation results. Finally, Section VI concludes the paper.

II. BACKGROUND OF THE TRICKLE ALGORITHM AND THE RPL PROTOCOL

This section gives the details about the Trickle algorithm and the RPL protocol.

A. THE TRICKLE ALGORITHM

The Trickle algorithm runs for a variable-length interval and has three configuration parameters:

- The minimum interval length, I_{min}, is defined in units of time, e.g., milliseconds, seconds.
- The maximum interval length, I_{max}, is described as a number of doublings of the minimum interval length. Therefore, the maximum length of the interval specified by I_{max} would be $I_{min} \times 2^{Imax}$.
- The redundancy constant, k, is an integer greater than zero.

In addition to these three parameters, the Trickle algorithm maintains three variables:

- I, the current interval length.
- t, a transmission time within the current interval.
- c, a consistency counter within the current interval.

The behavior of the Trickle algorithm can be described by the following six rules:

1) When the algorithm starts, it sets I to a value in the range of $[I_{min}, I_{max}]$ and begins the first interval.
2) At the beginning of the interval I, the algorithm resets c to 0 and sets t to a random point within the range of $[I/2, I]$. The first half of the interval I is so-called the listen-only period.
3) Whenever a node senses a consistent transmission, the algorithm increments the counter c.
4) At time t, the algorithm allows a node to transmit if and only if the consistency counter c is less than the redundancy constant k; otherwise, the transmission of a node is suppressed.
5) When the interval I expires, the algorithm doubles the interval length and starts a new interval as described in 2). Note that if this new interval length would be longer than the length of I_{max}, the algorithm sets the new interval length to be I_{max}.
6) Whenever a node senses an inconsistent transmission and I is greater than I_{min}, the algorithm sets I to I_{min} and starts a new interval as described in 2). If I is equal to I_{min} when a node senses an inconsistent transmission, the algorithm does nothing. In response to external events, the algorithm can also reset I to I_{min} and start a new interval as described in 2).

Note that the definitions of consistent states, inconsistent states, and external events depend on how a protocol uses the Trickle algorithm.

B. THE RPL PROTOCOL

The RPL protocol is a routing protocol designed for the low-power and lossy network (LLN), in which all the nodes are resource-constrained devices. The basic idea behind the protocol is to construct a Destination-Oriented Directed Acyclic Graph (DODAG) over a network so that the protocol can support both upward and downward routes. In general, a data collection node in the network is designated to be the root node of the DODAG and all the other nodes join the DODAG by learning a route towards the root node. Specifically, the DODAG is formed in accordance with an objective function (OF) of the protocol that guides each non-root node to select a preferred parent when processing the received DODAG Information Object (DIO) message. Several OFs have been documented in RFCs, such as the objective function zero (OF0) [4] and the Minimum Rank with Hysteresis Objective Function (MRHOF) [5].

In the context of the RPL protocol, the Trickle algorithm is used to regulate the sending rate of DIO messages for each node and the construction of the DODAG starts with the DIO message emitted by the root node. When a non-root node receives a DIO message, the node processes the received DIO message to select a preferred parent. In addition, the node joins the DODAG and starts to emit its DIO message just as the root node does after processing the first received DIO message. The DODAG is completely formed once all the nodes have joined the DODAG. After the complete formation of the DODAG, whenever a non-root node needs to send a packet to the root node, the non-root node sends the packet to its preferred parent, which will then forward the packet until the packet reaches the root node.

Since the RPL protocol uses the Trickle algorithm to disseminate the DIO message, it is up to the RPL protocol

to decide the consistency or inconsistency of the received DIO message. The transmission of the DIO message is non-stop and hence each non-root node could possibly change its preferred parent at some time in the future. When the preferred parent does not change after processing the received DIO message, the received DIO message is considered to be consistent. However, if the preferred parent changes, the received DIO message is considered to be inconsistent.

III. RELATED WORK

There are some problems while applying the Trickle algorithm to the RPL protocol, such as the route convergence time, fairness among nodes, and power consumption. Therefore, some variants of the Trickle algorithm have been proposed in the past to tackle the problems. For instance, Ghaleb *et al.* [6] proposed a variant called E-Trickle to shorten the route convergence time by eliminating the need of the listen-only period. According to the results, this variant improves the route convergence time but the power consumption and the PDR almost remain the same as the Trickle algorithm. Additionally, Djamaa and Richardson [7] proposed a variant called Opt-Trickle to shorten the route convergence time after an inconsistency occurred in the network. This variant can improve the route convergence time as well by only eliminating the need of the listen-only period on those intervals reset back to the minimum interval length I_{min} due to inconsistencies. However, this variant also introduces a little extra transmission cost. Instead of eliminating the listen-only period, Jeong *et al.* [8] proposed a variant called A^2-Trickle to shorten the route convergence time by utilizing three techniques. This variant applies an interval boundary alignment to the minimum interval I_{min} for each non-root node so that the propagation time can be shortened when the transmission begins. Subsequently, a tiling mechanism is established on the transmission period for all the intervals after the minimum interval in order to avoid the collision during the transmission and hence help in the message propagation. Moreover, the adaptive suppression scheme of this variant turns the redundancy constant k into a variable and adapts the value of k at run time by utilizing the neighbor information, and therefore bottleneck nodes can be released from the suppression for the message propagation. Based on the results, this variant improves the route convergence time and the reliability of the network.

A number of the variants have focused on improving fairness for the Trickle algorithm. At first, Vallati and Mingozzi [9] proposed a variant called Trickle-F, which addresses the fairness problem by utilizing the number of consecutive suppressions. This variant introduces a variable s to keep track of the number of consecutive suppressions. The rationale is that a node can get a higher transmission priority proportional to the number of last consecutive suppressions if the node has been suppressed for a long period of time. This variant guarantees fairness in the transmission of DIO messages as a consequence of prioritization. Subsequently, Meyfroyt *et al.* [10] proposed another variant called Adaptive-k to tackle the fairness problem by turning the redundancy constant k into a variable and dynamically adjusting the value of k based on the number of received DIO messages, i.e., the consistency counter c. That is, for each node, the value of k will be independently adjusted to a higher or lower value if the number of received DIO messages during an interval increases or decreases, respectively. Besides, in case of deadlock, this variant introduces k_{min} and k_{max} constants to be the lower and upper bounds for the adjusted value of k. The variant called Trickle-D proposed by Vučinić *et al.* [11] also utilizes the number of received DIO messages to dynamically adjust the value of k for each node, where the adjustment on k is scaled proportionally to the difference of the number of received DIO messages and the number of neighbors. In fact, both Adaptive-k and Trickle-D have to carefully select k_{min} and k_{max} constants for a given network in order to control the total number of DIO messages in the network.

Some other variants have focused on reducing unfairness among nodes and power consumption at the same time. For instance, Ghaleb *et al.* [12] and [13] proposed a variant called Drizzle, which eliminates the need of the listen-only period and utilizes the corresponding history of a node to decide the range of the random selection for the time t within the interval I. As one might notice, the decision on the random selection for the time t is a major difference between E-Trickle and Drizzle. The outcome of the decision for Drizzle is that nodes will compete with each other for the next transmission on the interval-level when they have the same number of cumulative transmissions and the same number of elapsed intervals since the minimum interval length I_{min}. To further distinguish between competing nodes for the fairness purpose, Drizzle also dynamically adjusts the reference value of the redundancy constant k for each node with an adaptive suppression mechanism based on the suppression history. One important thing to note here is that the sending rate will be adjusted to the lowest when Drizzle detects some inconsistent events. In addition, the variant called I-Trickle proposed by Goyal and Chand [14] also uses a similar policy to decide the range of the random selection for the time t within the interval I, but the decision is based on a variable s which keeps track of the number of consecutive suppressions. Overall, these two variants can cause the network to produce a smaller total number of DIO messages and hence reduce power consumption.

Table 1 briefly summarizes the Trickle variants described in this related work section.

IV. FI-TRICKLE

The proposed algorithm is named FI-Trickle in this paper and a pseudo-code of FI-Trickle is presented in Algorithm 1. In the Trickle algorithm, each of the functions in the pseudo-code is basically invoked according to the six rules presented in Section II. A related flowchart that illustrates the pseudo-code is presented in Fig. 1, where each of the blue

TABLE 1. Summary of different Trickle variants.

Trickle variant	Brief description	Improvement
E-Trickle [6]	Eliminating the need of the listen-only period	Improving the route convergence time
Opt-Trickle [7]	Eliminating the need of the listen-only period on those intervals reset back to the minimum interval length I_{min} due to inconsistencies	Improving the route convergence time
A^2-Trickle [8]	Aligning and tiling the transmission period boundaries with an adaptive suppression scheme	Improving the route convergence time
Trickle-F [9]	Utilizing the number of consecutive suppressions to guarantee fairness in the transmission of DIO messages	Improving fairness among nodes
Adaptive-k [10]	Turning the redundancy constant k into a variable and dynamically adjusting the value of k based on the number of received DIO messages, i.e., the consistency counter c	Improving fairness among nodes
Trickle-D [11]	Turning the redundancy constant k into a variable and dynamically adjusting the value of k based on the number of received DIO messages and the number of neighbors	Improving fairness among nodes
Drizzle [12] and [13]	Eliminating the need of the listen-only period and selecting time t within the interval I based on the transmission history, also turning the redundancy constant k into a variable and dynamically adjusting the value of k based on the suppression history	Improving fairness among nodes and power consumption at the same time
I-Trickle [14]	Eliminating the need of the listen-only period and selecting time t within the interval I based on the suppression history	Improving power consumption

colored rectangles and diamonds can find its corresponding function in Algorithm 1.

An apparent problem of the Trickle algorithm is the uneven load distribution of DIO messages among nodes. In practice, some nodes may emit more DIO messages than others, even though the algorithm provides each node with equal average transmission probability in the long run. This problem may also occur even if the inconsistent transmission does not happen in the network. Besides, this phenomenon becomes

Algorithm 1 FI-Trickle

```
Function Initialization()
    I ← random(I_min, I_max)
    c ← 0
    s_flag ← flase
Function IntervalBegins()
    t ← random(I/2, I)
Function ConsistentTransmissionReceived()
    c ← c + 1
Function InconsistentTransmissionReceived()
    I ← I_min
    c ← 0
    s_flag ← flase
Function TimerExpires()
    if c < k then
        Transmit DIO
    else
        s_flag ← true
    end if
    c ← 0
Function IntervalEnds()
    if s_flag == false then
        I ← I × 2
        if I ≥ I_max then
            I ← I_max
        end if
    end if
    s_flag ← false
```

obvious especially when the redundancy constant k is set to a lower value.

According to the previous observations, two simple modifications are made on the Trickle algorithm. First, an *s_flag* variable is introduced to indicate that the transmission at time t is suppressed or not so that FI-Trickle can decide if it should double the interval for the next round according to the value of the flag at the end of the interval. As a consequence of that, and because a suppressed node will have a shorter interval for the next round as compared to its neighbors, the node then will have a higher chance to transmit before its neighbors do it. Second, the consistency counter c is reset to 0 at time t during the interval instead of at the beginning of the interval. This modification aims to eliminate redundant transmissions. As reported in [6], it is possible to have redundant transmissions in an asynchronous network without applying the listen-only period. Even though FI-Trickle applies the listen-only period, the first modification of the algorithm may also cause the network to produce redundant transmissions. As shown in Fig. 2, for example, node 2 emits a redundant transmission on its third interval if the proposed algorithm resets the consistency counter c at the beginning of the interval.

It is worth to note that the first modification alone has the effect to balance the load distribution of DIO messages among nodes. However, the modification also produces a

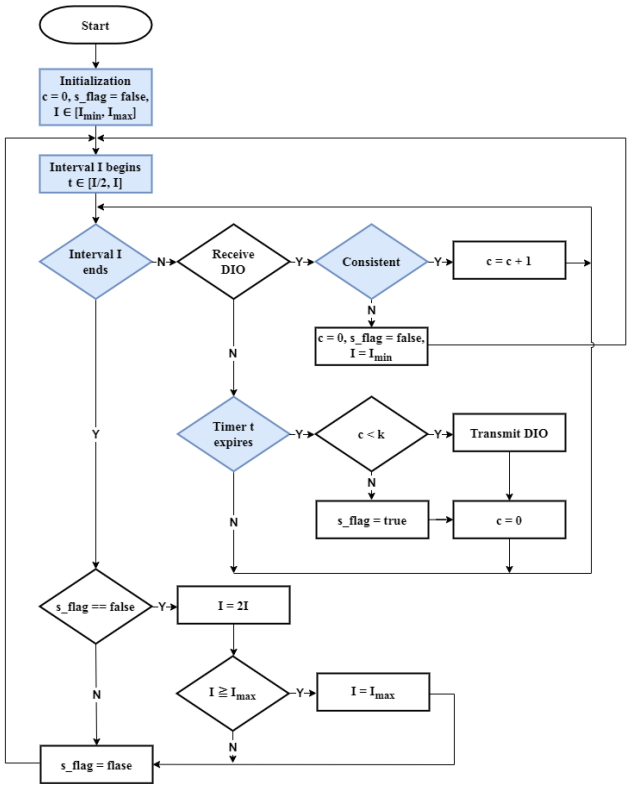


FIGURE 1. Flowchart for FI-Trickle algorithm.

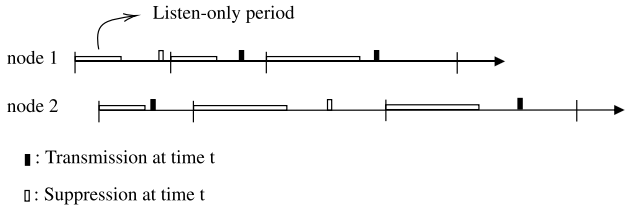


FIGURE 2. *k* is set to 1 and the *c* is reset to 0 at the beginning of every interval, while a redundant transmission happened for node 2.

greater total number of DIO messages in comparison with the Trickle algorithm. Consequently, the two modifications has to be combined together in order to make the load distribution even as well as simultaneously reduce the total number of DIO messages.

V. EXPERIMENT DESIGN AND SIMULATION RESULTS

A. SIMULATION ENVIRONMENT AND METHODOLOGY

In general, Contiki OS has two generations: the historical generation Contiki[1] and the new generation Contiki-NG.[2] Both generations share the same protocol stack and RPL features. Most of the work use Contiki OS to verify the performance for two reasons. First, it has a practical RPL implementation that uses the Trickle algorithm. Second, it comes with a network simulator called Cooja that allows users to conduct experiments on fully emulated hardware devices. In this work, Contiki-NG is selected to conduct the experiment. In the experiment, the MAC and adaptation layers of Contiki-NG are set to use CSMA and 6LoWPAN, respectively. As for the configuration of the RPL protocol, MRHOF is chosen as the OF to be used when constructing the DODAG for the network. Additionally, the rpl-udp application of Contiki OS executes at the same time while the simulation is running to facilitate the measurement of the PDR results. The PDR is measured by the ratio of the number of application packets a sink node receives to the number of application packets sender nodes inject in to the network. The power profiling is done along with the application by using the powertrace library for the sake of power measurement. The emulated device is Tmote Sky that can be powered by battery with 2900mAh capacity and we assume the device operating voltage is 3V.

The experiment is designed to explore how FI-Trickle will perform on different network sizes and under difference interference conditions. To this end, the network sizes are set to be 25, 49, 81, or 100 nodes and for each network size the RX success ratio is adjusted to be 60%, 80%, or 100%. The nodes are deployed in a grid topology with a fixed distance in-between or in a random topology within a 200m × 200m field size. It is important to note that the network density gets higher when adding more nodes to the field for the random topology scenario. In each network, regardless of the grid or the random topology, one of the nodes is set to be the sink node which collects the application packets from all the other nodes. For instance, the sink node is placed in the center of the grid and random topology as demonstrated in Fig. 3 and Fig. 4, respectively. The Unit Disk Graph Radio Medium (UDGM) model of the Cooja simulator is used to define the characteristics of the transmission range, the interference range, and the radio propagation for all nodes.

For each experiment, the minimum interval length I_{min}, the maximum interval length I_{max}, and the redundancy constant k of the Trickle algorithm are set to 2^4, 2^{14}, and 2, respectively. Each experiment with 25, 49, or 81 nodes was carried out 50 times with different random seeds to get the average results. The same approach of getting the average results is applied to each experiment with 100 nodes but each experiment was only carried out 15 times due to significantly longer runtime for each result. The simulation time of each run is 10 virtual minutes, which is enough to cover the elapsed intervals from the minimum interval length I_{min} to the maximum interval length I_{max}. Table 2 summarizes the configurations and parameters used in the experiments.

All the experiments are also conducted on Trickle, Trickle-F, I-Trickle, and Drizzle for the comparison purpose. E-Trickle, Opt-Trickle, and A^2-Trickle are not considered in the comparison because their goal is to shorten the route convergence time which is not the focus of the paper. We also did not compare with Adaptive-k and Trickle-D because they need additional setting of the constants, k_{min} and k_{max}. The

[1]https://github.com/contiki-os/contiki
[2]https://github.com/contiki-ng/contiki-ng

TABLE 2. Experimental configurations and parameters.

The MAC/Adaptation Layer	
MAC	CSMA
Adaptation	6LoWPAN
The Trickle algorithm	
I_{min} / I_{max}	2^4 / 2^{14} (ms)
Redundancy constant k	2
The RPL protocol	
OF	MRHOF
The rpl-udp application	
Data packet generation rate	1 packet per minute
Data payload size	46 bytes
The Cooja simulator	
Number of nodes	25, 49, 81, 100
Topology	grid / random
Radio medium model	UDGM
Transmission range	75m
Interference range	100m
RX success ratio	60%, 80%, 100%
Simulation time	10 virtual minutes
The emulated device	
Device	Tmote Sky
Operating voltage	3V
Battery capacity	2900mAh

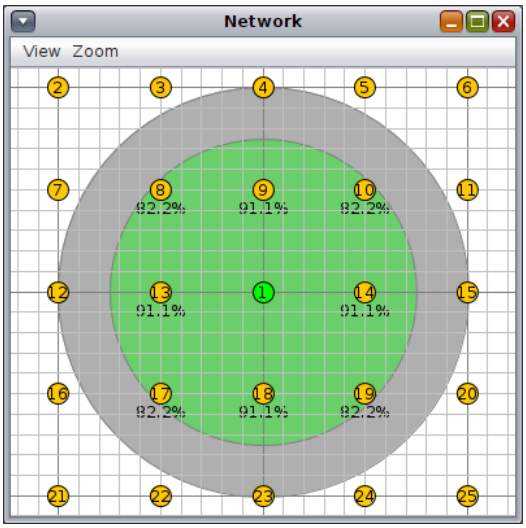


FIGURE 3. Grid topology with 25 nodes in the network. The green node represents the sink node.

following subsections will present the simulation results on the grid and the random topology, respectively.

B. GRID: SIMULATION RESULTS

The first part of simulation results to be examined is the average summation of DIO messages emitted by sender nodes. This part of results is to evaluate the behavior of sender nodes so that the sink node is excluded. From Fig. 5 it can be seen that Drizzle causes the network to produce the fewest messages across the network size and across the interference condition, respectively. The reason is that Drizzle adjusts the I interval to the maximum interval length I_{max} for the lowest sending rate once a node has encountered some inconsistent events. Besides, I-Trickle also does a lot more suppressions

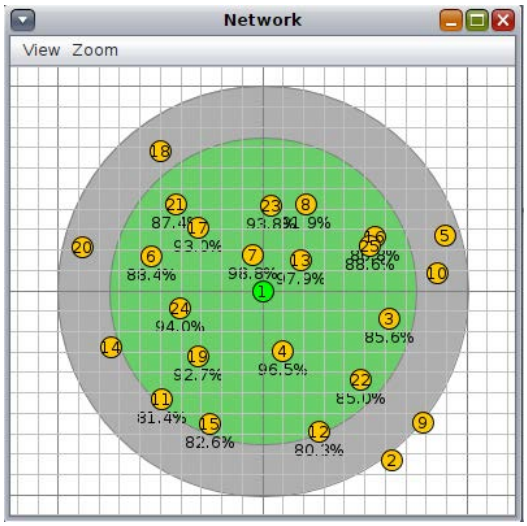


FIGURE 4. Random topology with 25 nodes in the network. The green node represents the sink node.

to nodes in the network as compared to Trickle, Trickle-F, and FI-trickle. Interestingly, the figure shows that the result of FI-Trickle falls between that of Trickle-F and I-Trickle for each network size and for each interference condition. This outcome indicates that FI-Trickle will also do some more suppressions to nodes in the network as compared to Trickle-F, but not as much as I-Trickle does.

In general, more DIO messages in the network will lead to higher power consumption. Fig. 6 explicitly shows that the average power consumption of sender nodes has a strong relation with the average summation of DIO messages emitted by sender nodes and Fig. 7 shows the average network lifetime results based on Fig. 6 for the emulated device with the operating voltage and the battery capacity assumed in Table 2. It turns out that nodes can save the most power overall when nodes use Drizzle to emit DIO messages. However, one important thing to notice here is that this power saving characteristic of Drizzle has a side effect of degrading the PDR. This is because nodes rely on DIO messages to update their routes and nodes with lower numbers of DIO messages can result in a relatively rare update of the route for the network. According to this fact, it appears that there is a trade-off between the power consumption and the PDR. The following discussion will focus more on Trickle, Trickle-F, I-Trickle, and FI-Trickle since they are on a similar level of emitted DIO message numbers according to Fig. 5.

Trickle-F

and FI-Trickle share the same design goal (i.e., fairness among nodes). However, there are three major differences between them that cause different results on the average summation of DIO messages emitted by sender nodes and on the average power consumption of sender nodes as shown in Fig. 5 and in Fig. 6, respectively. The average number of DIO messages emitted by each node is further examined to discover the three major differences. For instance, Fig. 8

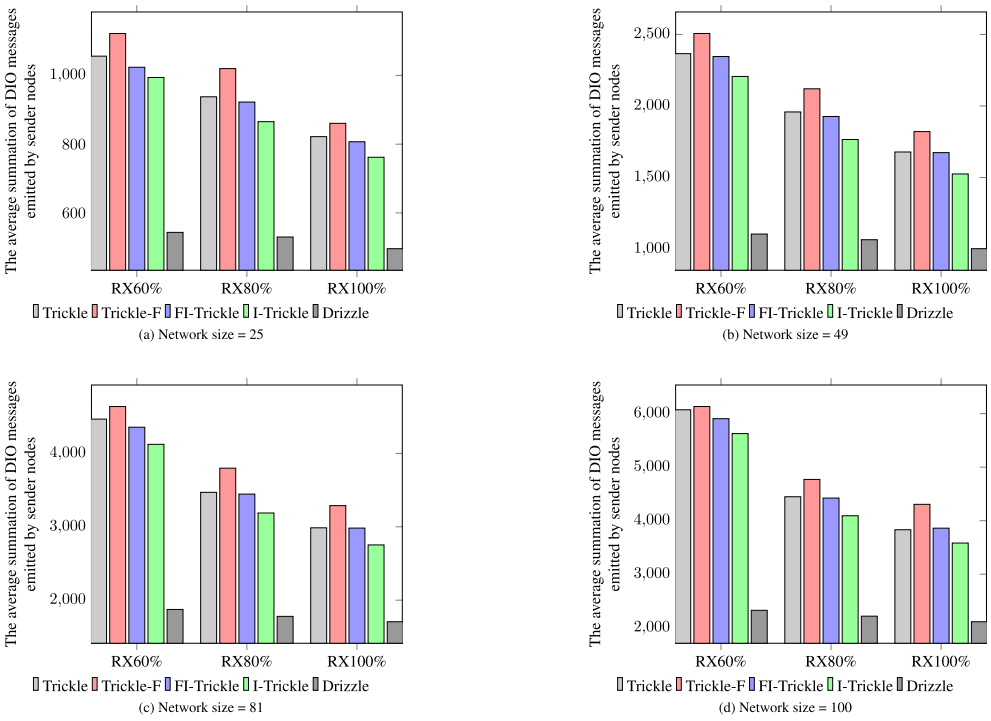


FIGURE 5. Grid: the average summation of DIO messages emitted by sender nodes on various network sizes and under various RX success ratios.

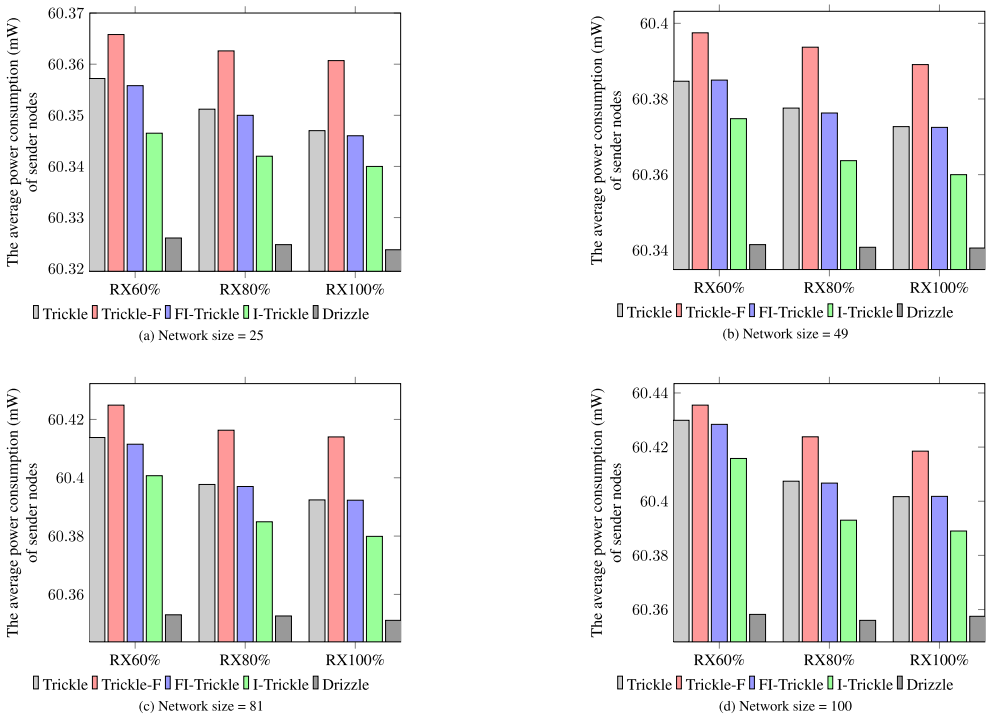


FIGURE 6. Grid: the average power consumption (mW) of sender nodes on various network sizes and under various RX success ratios.

shows the average number of DIO messages emitted by each node when the network size = 25 and the RX success ratio = 100% for Trickle-F and FI-Trickle. Each cell in Fig. 8 can find a corresponding node mapping from Fig. 3. The three differences between Trickle-F and FI-Trickle: First, the sink node in FI-Trickle emitted fewer DIO messages than in Trickle-F.

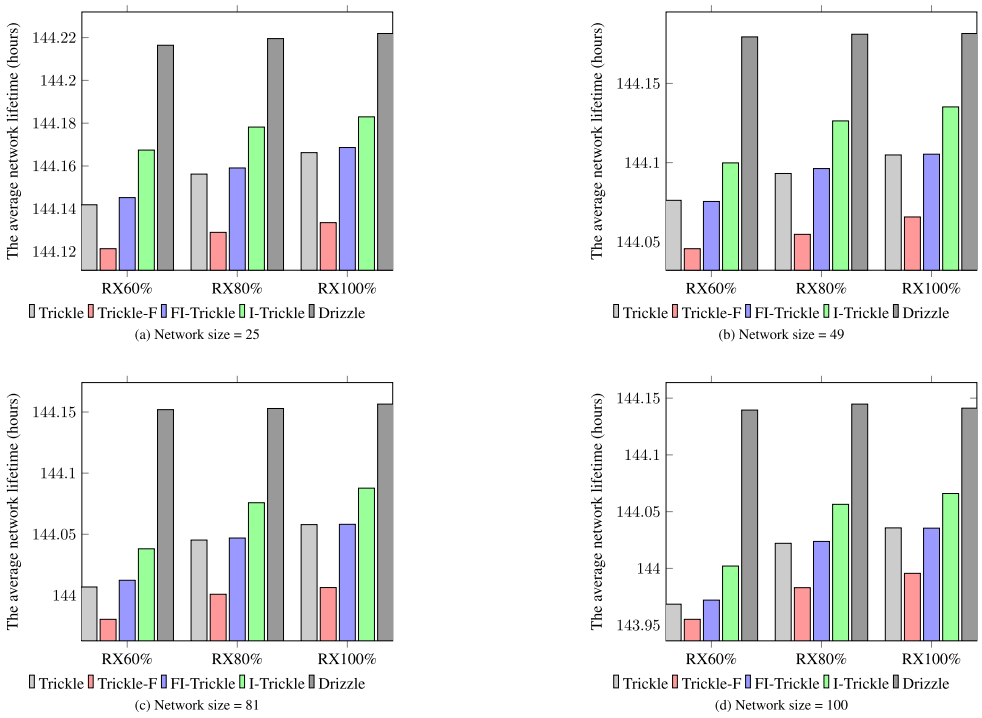
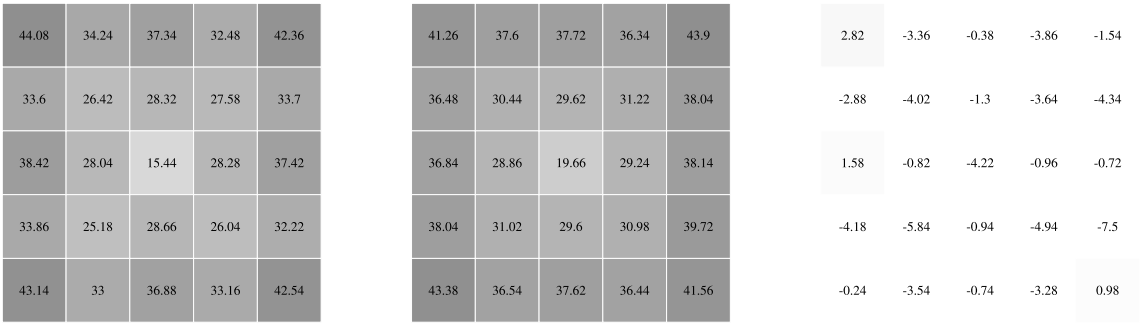


FIGURE 7. Grid: the average network lifetime (hours) on various network sizes and under various RX success ratios.



(a) FI-Trickle (b) Trickle-F (c) The difference

FIGURE 8. Grid: the average number of DIO messages emitted by each node when the network size = 25 and the RX success ratio = 100%.

Secondly, most of the nodes at the edge in FI-Trickle emitted fewer DIO messages than in Trickle-F. Lastly, the intermediate nodes in FI-Trickle can effectively disseminate the DIO messages to their neighbors with fewer collisions due to the first two differences. These three differences can be seen across the network size and across the interference condition for Trickle-F and FI-Trickle. The differences are due to the mechanism within FI-Trickle that keeps the interval length unchanged for the next round when a node is suppressed. The mechanism will cause the asymmetry on the interval lengths between the sink node and the intermediate nodes (i.e., the neighbors of the sink node), and the effect of this asymmetry will further cause the sink node to be highly suppressed since the period of every two consecutive times ts of the sink node will be overlapped with time ts of the intermediate nodes most of the time. Subsequently, more suppressions will be added to most of the nodes at the edge since the intermediate nodes gain more opportunities over the sink node to transmit more DIO messages. Specifically, the highly suppressed sink node will not damage the network. In fact, it is unnecessary for the sink node to tell its neighbors about the parentship all the time since the sink node is always the preferred parent of its neighbors. Fig. 8(c) summarizes the differences, cell by cell, by subtracting the corresponding value of the cell in Fig. 8(b) from the corresponding value of the cell in Fig. 8(a). According to Fig. 8(c), the summation of all cells except the cell related to the sink node exactly explains that the DIO messages of FI-Trickle are fewer than those of Trickle-F

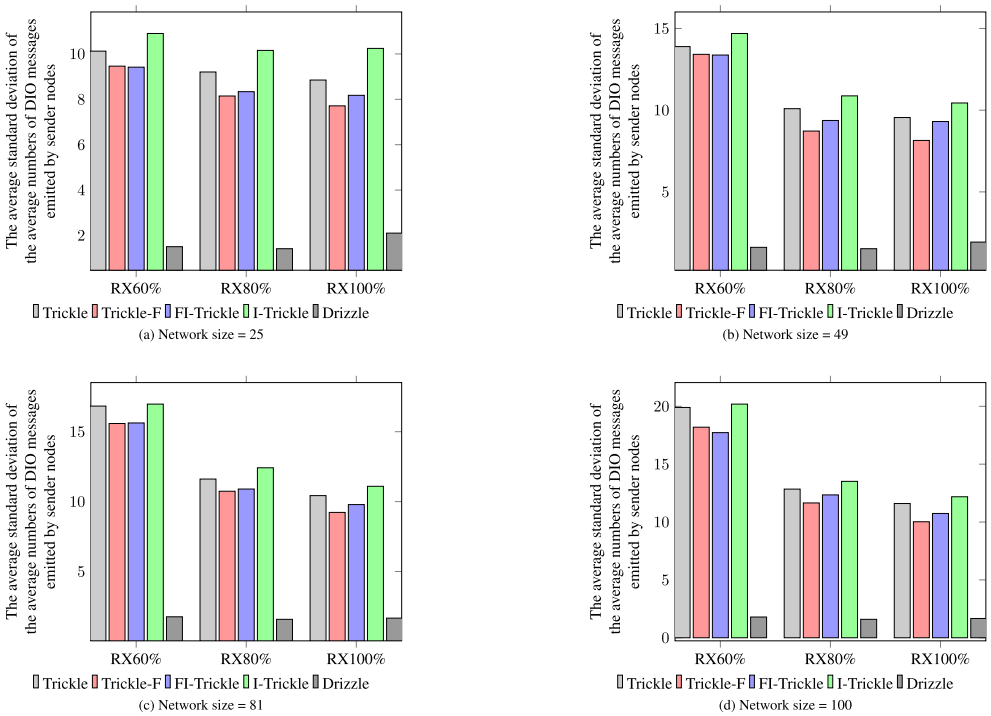


FIGURE 9. Grid: the average standard deviation of the average numbers of DIO messages emitted by sender nodes on various network sizes and under various RX success ratios.

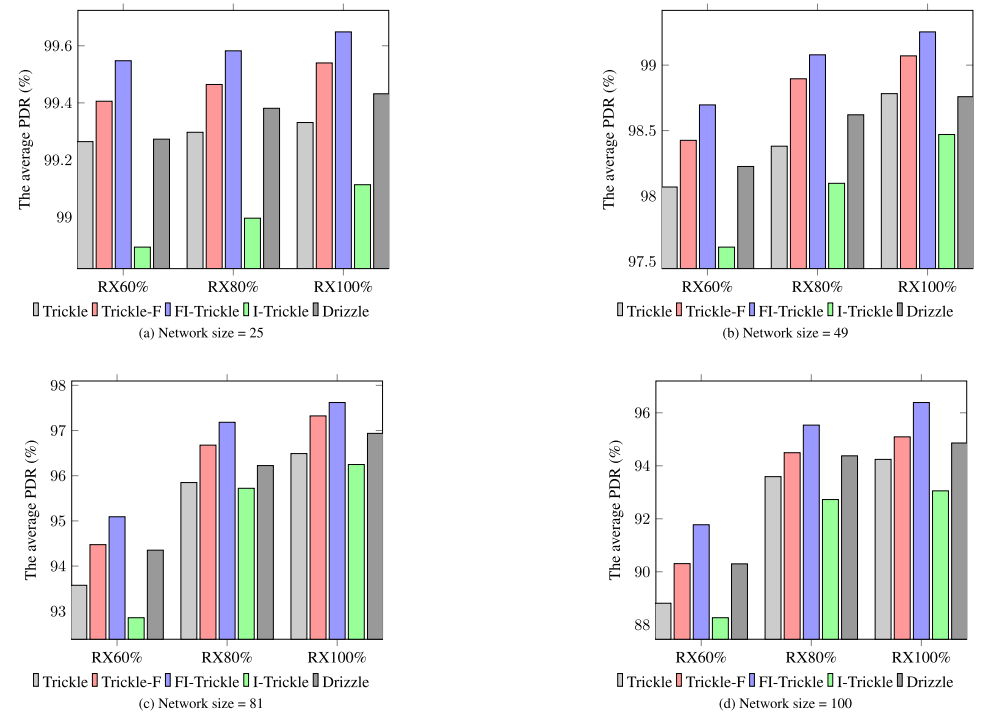


FIGURE 10. Grid: the average PDR (%) on various network sizes and under various RX success ratios.

both in Fig. 5 and Fig. 6. Most importantly, the decreased number of DIO messages emitted by the sink node and the decreased numbers of DIO messages emitted by nodes at the edge can result in a more effective update of the route for the network.

The average standard deviation of the average numbers of DIO messages emitted by sender nodes is used to evaluate fairness among sender nodes. As shown in Fig. 9, Drizzle has the lowest standard deviation across the network size and across the interference condition. This is because its

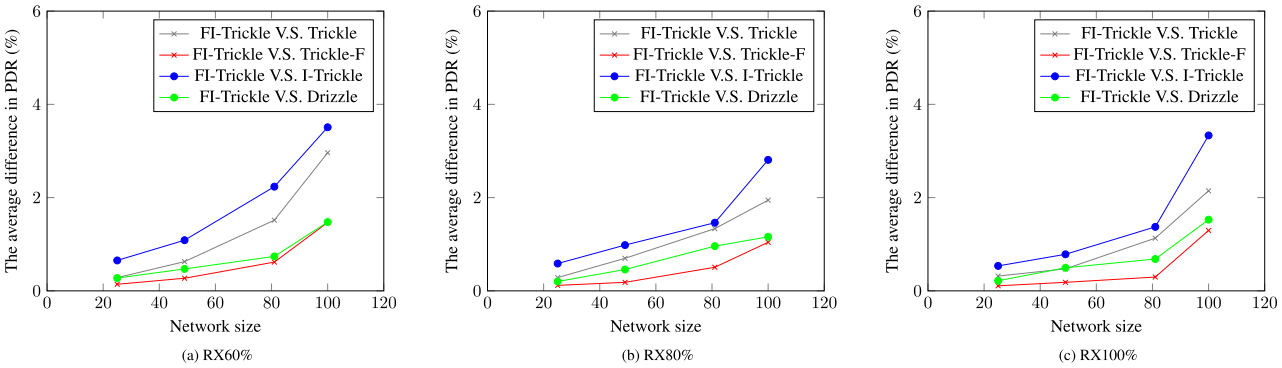


FIGURE 11. Grid: the average difference in PDR (%) as the network size grows and under various RX success ratios when comparing FI-Trickle to Trickle, Trickle-F, I-Trickle, and Drizzle, respectively.

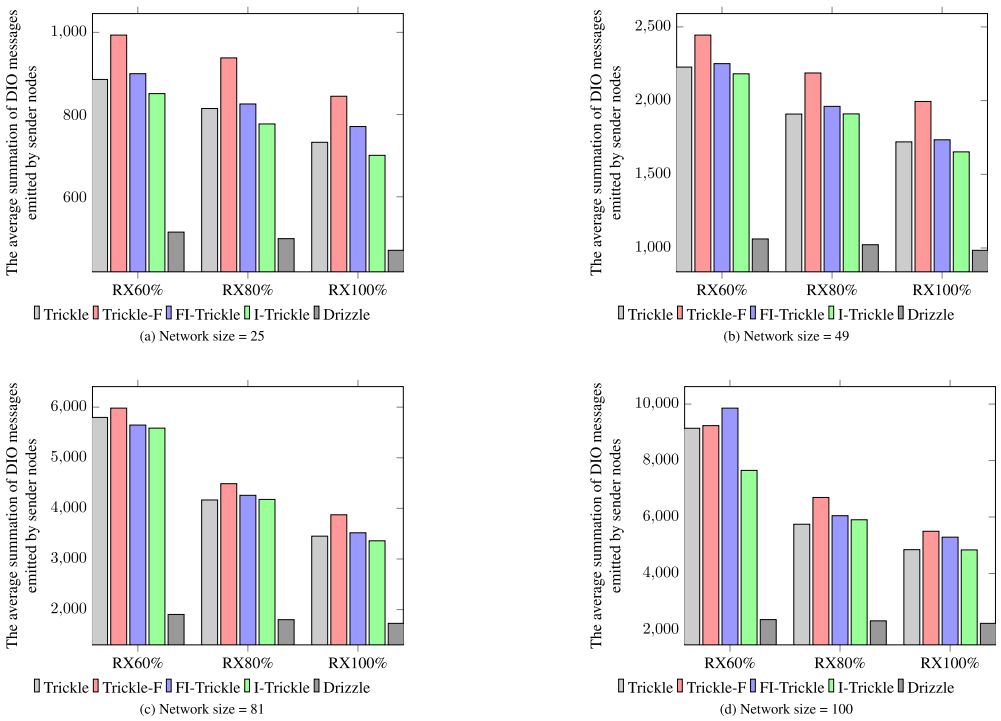


FIGURE 12. Random: the average summation of DIO messages emitted by sender nodes on various network sizes and under various RX success ratios.

mechanism of random selection for the time t can well distribute the time t between nodes on the interval-level. On the other hand, I-Trickle has the highest standard deviation across the network size and across the interference condition. This high value of standard deviation indicates that some sender nodes emit either a lot more or a lot fewer DIO messages than the average value of all sender nodes. In contrast to Trickle and I-Trickle, Trickle-F and FI-Trickle both have lower standard deviations, which justify the improvement in fairness among sender nodes. Moreover, the result also shows that FI-Trickle can achieve fairness among sender nodes as much as Trickle-F does.

Fig. 10 shows the results of average PDR for each algorithm across the network size and across the interference condition. As can be seen, I-Trickle has the lowest PDR. According to Fig. 5, both I-Trickle and Drizzle have lower numbers of DIO messages which result in a relatively rare update of the route. However, the PDR of Drizzle is slightly better than that of I-Trickle because Drizzle can achieve fairness among sender nodes. On the other hand, FI-Trickle has the best PDR result. This can be explained with previous observations obtained from Fig. 8 which show that FI-Trickle can cause intermediate nodes to have a more effective update of the route for the network. Additionally, it is important to note that FI-Trickle breaks the trade-off between the power consumption and the PDR when compared to Trickle-F because of the three major differences between FI-Trickle and Trickle-F. Overall, through out the results from Fig. 5 to Fig. 10, it shows

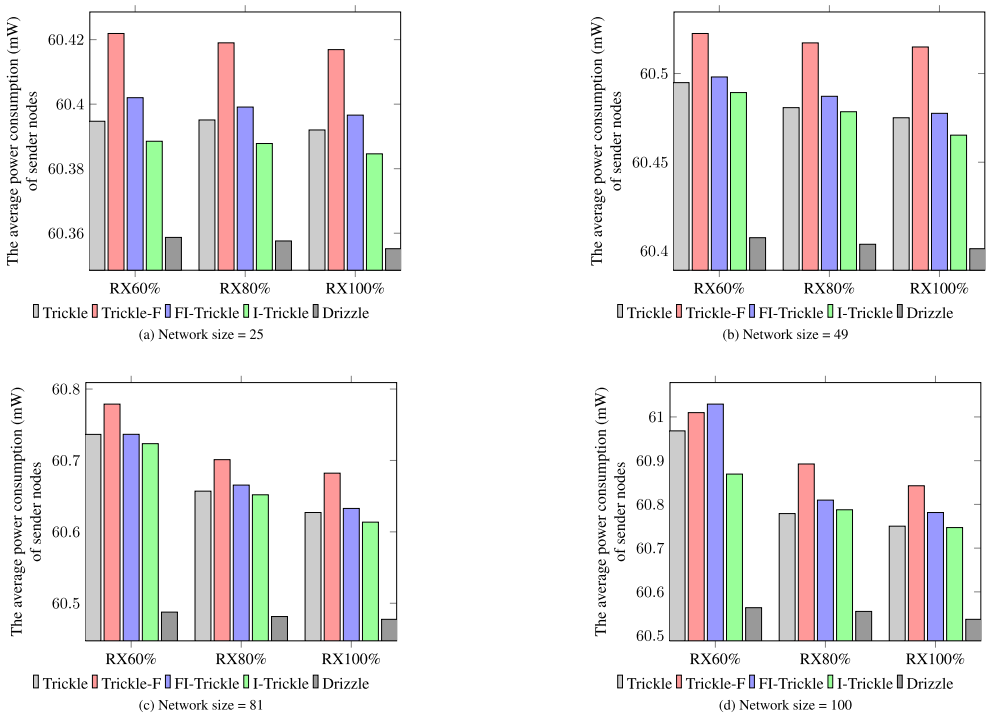


FIGURE 13. Random: the average power consumption (mW) of sender nodes on various network sizes and under various RX success ratios.

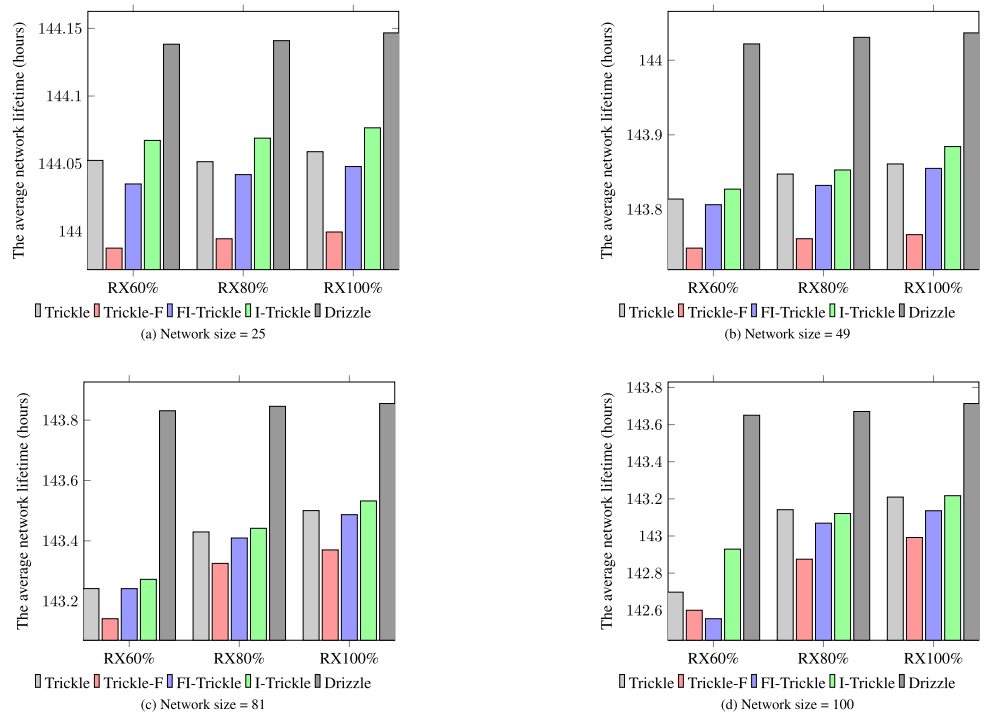


FIGURE 14. Random: the average network lifetime (hours) on various network sizes and under various RX success ratios.

that FI-Trickle produces the best PDR result with a power consumption less than the power consumption of Trickle-F. In addition, Drizzle uses the least power consumption to produce a PDR result better than the PDR of I-Trickle.

Finally, Fig. 11 shows the average difference in PDR as the network size grows and under various RX success ratios when comparing FI-Trickle to Trickle, Trickle-F, I-Trickle, and Drizzle, respectively. It can be seen that as the network

43.56	29.82	31.48	36.86	31.02
26.42	30.44	40.82	31.52	35.94
27.84	28.5	10.06	32.22	34.22
30.72	28.94	34.54	30.74	35.42
29.14	30.8	30	29.32	30.58

(a) FI-Trickle

44.08	34.56	32.84	37.56	31.58
30.44	32.56	44.2	34.4	38.12
35.56	30.32	15.92	37.64	38.38
35.42	32.28	37.98	33.46	39.54
33.32	34.32	32.6	30.92	32.5

(b) Trickle-F

-0.52	-4.74	-1.36	-0.7	-0.56
-4.02	-2.12	-3.38	-2.88	-2.18
-7.72	-1.82	-5.86	-5.42	-4.16
-4.7	-3.34	-3.44	-2.72	-4.12
-4.18	-3.52	-2.6	-1.6	-1.92

(c) The difference

FIGURE 15. Random: the average number of DIO messages emitted by each node when the network size = 25 and the RX success ratio = 100%.

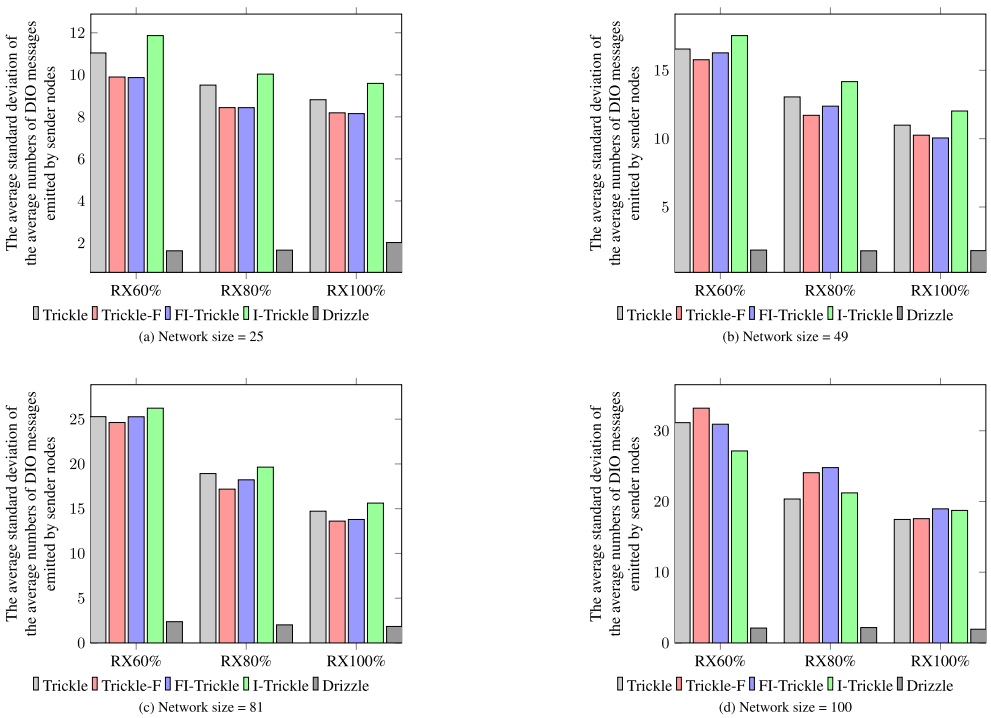


FIGURE 16. Random: the average standard deviation of the average numbers of DIO messages emitted by sender nodes on various network sizes and under various RX success ratios.

size grows the differences show a tendency to increase for all the results, i.e., the gray, red, green, and blue lines in Fig. 11. According to IETF RFCs, LLN application scenarios will involve high numbers of wireless devices [15]–[18]. The increasing trend of the differences in PDR results implies that FI-Trickle can be a better candidate for the dissemination of the RPL message for large-scale networks.

C. RANDOM: SIMULATION RESULTS

In the random topology scenario the nodes are deployed within a 200m × 200m field size therefore the network density gets higher when adding more nodes to the field. It is expected that more collisions can be observed when the network density gets higher and the network performance will eventually be impacted by the collisions if the network size grows over a threshold, such as the results are impacted and becomes chaotic when the network size = 100 in the experiment. Consequently, the following discussion will focus on network size = 25, 49, and 81.

As in the case of the grid topology, the first part of simulation results to be examined is the average summation of DIO messages emitted by sender nodes. From Fig. 12 it can be seen that for the same reason found in the grid simulation results, Drizzle still causes the network to produce the fewest messages over all network sizes and interference conditions, respectively. However, when the network size grows the average summation of DIO messages given by FI-Trickle becomes close to the average summation of DIO messages given by I-Trickle, which is different to the results of the grid case, as can be seen by comparing Fig. 12 with

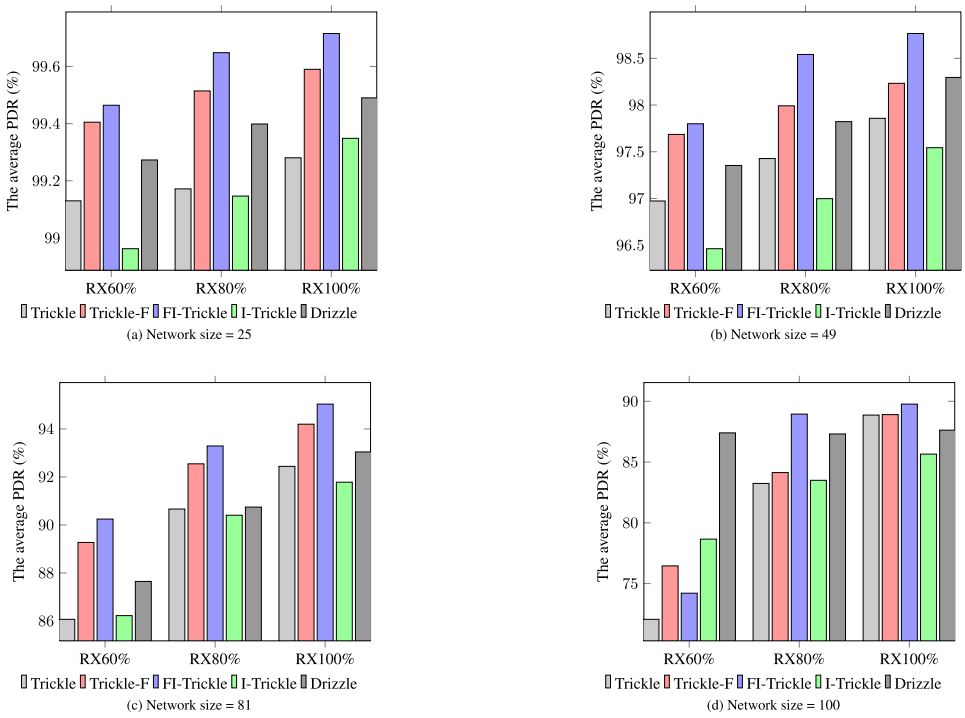


FIGURE 17. Random: the average PDR (%) on various network sizes and under various RX success ratios.

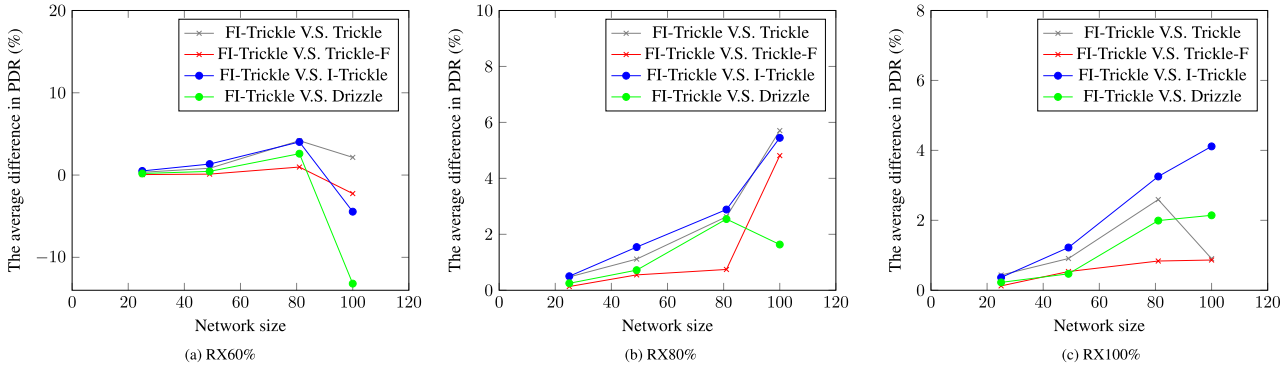


FIGURE 18. Random: the average difference in PDR (%) as the network size grows and under various RX success ratios when comparing FI-Trickle to Trickle, Trickle-F, I-Trickle, and Drizzle, respectively.

Fig. 5. The reason behind this difference will be discussed in the following paragraph. In addition, when putting Fig. 12 and Fig. 13 side by side, it can be seen that a strong relation still exists between the average summation of DIO messages emitted by sender nodes and the average power consumption of sender nodes. Fig. 14 shows the average network lifetime results based on Fig. 13 for the emulated device with the operating voltage and the battery capacity assumed in Table 2.

The result that FI-Trickle produces fewer DIO messages in the random topology can be explained with Fig. 15. Each cell in Fig. 15 can find a corresponding node mapping from Fig. 4. Fig. 15(a) and Fig. 15(b) present the average number of DIO messages emitted by each node for FI-Trickle and Trickle-F, respectively. In Fig. 15(c), it can be seen that each node running with FI-Trickle produces fewer DIO messages as compared to Trickle-F. This phenomenon can be further identified by comparing Fig. 3 and Fig. 4. In the grid topology for instance Fig. 3, nodes will have a neighbor count between 3 and 8. On the other hand, in the random topology for instance Fig. 4, some nodes can have a neighbor count more than 10, showing a situation of the network clustering. It appears that in the random topology FI-Trickle remains the capability of effective route update but has slightly more suppressions as compared to the grid topology due to the network clustering. In the random topology, even though more suppressions happened to FI-Trickle the average standard deviation of DIO messages of FI-Trickle remains similar to the results of Trickle-F as shown in Fig. 16.

Fig. 17 presents the results of average PDR for each algorithm over various network sizes and interference conditions. When excluding the network size = 100, these results are consistent with the results presented in Fig. 10 thanks to the effective route update capability of FI-Trickle. Overall, through out the results from Fig. 12 to Fig. 17, it shows again

that FI-Trickle produces the best PDR result with a power consumption less than the power consumption of Trickle-F. In addition, Drizzle uses the least power consumption to produce a PDR result better than the PDR of I-Trickle. Finally, Fig. 18 shows the increasing average differences are noticed in PDR as the network size grows and under various RX success ratios when comparing FI-Trickle to Trickle, Trickle-F, I-Trickle, and Drizzle, respectively.

VI. CONCLUSION

This paper proposed an improved Trickle algorithm called FI-Trickle to achieve fairness among nodes with a new approach. The new approach allows each node to double or remain their interval length for their subsequent interval based on their suppression information in order to give each suppressed node a higher chance to transmit before the transmissions of its neighbors. Also, the suppression mechanism of FI-Trickle is enhanced in order to eliminate redundant transmissions.

The performance of FI-Trickle is verified with extensive experiments over various network sizes, interference conditions, and network topologies. The obtained simulation results show that FI-Trickle can achieve a similar fairness level as compared to Trickle-F. In addition, FI-Trickle can improve the PDR of the network and the power consumption of nodes at the same time. The reason is that in FI-Trickle most nodes emit fewer DIO messages and intermediate nodes can effectively disseminate the DIO messages to their neighbors with fewer collisions for the route update. Finally, the results further show that FI-Trickle can increasingly outperform Trickle, Trickle-F, I-Trickle, and Drizzle in terms of PDR as the network size grows. Since LLN applications will involve high numbers of wireless devices, the final result of the experiments implies that FI-Trickle can be a better candidate for the dissemination of the RPL message for large-scale networks.

REFERENCES

[1] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proc. 1st USENIX/ACM Symp. Netw. Syst. Design Implement. (NSDI)*, San Francisco, CA, USA, 2004, pp. 15–28.

[2] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, *The Trickle Algorithm*, document RFC 6206, IETF, Mar. 2011.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, document RFC 6550, IETF, Mar. 2012.

[4] P. Thubert, *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*, document RFC 6552, IETF, Mar. 2012.

[5] O. Gnawali and P. Levis, *The Minimum Rank With Hysteresis Objective Function*, document RFC 6719, IETF, Sep. 2012.

[6] B. Ghaleb, A. Al-Dubai, and E. Ekonomou, "E-trickle: Enhanced trickle algorithm for low-power and lossy networks," in *Proc. IEEE Int. Conf. Ubiquitous Comput. Commun. (IUCC)*, Liverpool, U.K., Oct. 2015, pp. 1123–1129.

[7] B. Djamaa and M. Richardson, "Optimizing the trickle algorithm," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 819–822, May 2015.

[8] G. Jeong, M. Park, and J. Paek, "A^2-trickle: Adaptive & aligned trickle for rapid and reliable dissemination in low-power wireless networks," *IEEE Access*, vol. 8, pp. 214374–214382, 2020.

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

[10] T. M. M. Meyfroyt, M. Stolikj, and J. J. Lukkien, "Adaptive broadcast suppression for trickle-based protocols," in *Proc. IEEE 16th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[11] M. Vucinic, M. Król, B. Jonglez, T. Coladon, and B. Tourancheau, "Trickle-D: High fairness and low transmission load with dynamic redundancy," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1477–1488, Oct. 2017.

[12] B. Ghaleb, A. Al-Dubai, I. Romdha, Y. Nasser, and A. Boukerche, "Drizzle: Adaptive and fair route maintenance algorithm for low-power and lossy networks in IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[13] B. Ghaleb, A. Y. Al-Dubai, E. Ekonomou, I. Romdhani, Y. Nasser, and A. Boukerche, "A novel adaptive and efficient routing update scheme for low-power lossy networks in IoT," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5177–5189, Dec. 2018.

[14] S. Goyal and T. Chand, "Improved trickle algorithm for routing protocol for low power and lossy networks," *IEEE Sensors J.*, vol. 18, no. 5, pp. 2178–2183, Mar. 2018.

[15] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen, *Building Automation Routing Requirements in Low-Power and Lossy Networks*, document RFC 5867, IETF, Jun. 2010.

[16] A. Brandt, J. Buron, and G. Porcu, *Home Automation Routing Requirements in Low-Power and Lossy Networks*, document RFC 5826, IETF, Apr. 2010.

[17] K. Pister, P. Thubert, S. Dwars, and T. Phinney, *Industrial Routing Requirements in Low-Power and Lossy Networks*, document RFC 5673, IETF, Oct. 2009.

[18] M. Dohler, T. Watteyne, T. Winter, and D. Barthel, *Routing Requirements for Urban Low-Power and Lossy Networks*, document RFC 5548, IETF, May 2009.

SSU-TING LIU received the B.B.A. degree in information management from Yuan Ze University, Taoyuan, Taiwan, in 2006, and the M.S. degree in computer science and information engineering from the National Central University, Taoyuan, in 2008. He is currently pursuing the Ph.D. degree in electrical engineering with the National Taiwan University, Taipei, Taiwan. He was an Intern at IBM, Greater China, and Asia–Pacific—SWG, Taipei, from 2013 to 2014. He was an Engineer at Insyde Software, Taipei, from 2019 to 2020. His research interests include the Internet of Things, computer networks, embedded systems, and other similar related areas.

SHENG-DE WANG (Member, IEEE) received the B.S. degree from the National Tsing Hua University, Hsinchu, Taiwan, in 1980, and the M.S. and Ph.D. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1982 and 1986, respectively.

From 1995 to 2001, he has served as the Director for the Computer Operating Group, Computer and Information Network Center, National Taiwan University. He was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle, during the academic year of 1998–1999. From 2001 to 2003, he has also served as the Department Chair for the Department of Electrical Engineering, National Chi Nan University, Puli, Taiwan, for the two years of appointment. Since 1986, he has been with the Faculty of the Department of Electrical Engineering, National Taiwan University, where he is currently a Professor. His research interests include parallel and distributed computing, embedded systems, and computer security.

Prof. Wang is a member of the Association for Computing Machinery and IEEE Computer Society. He is also a member of Phi Tau Phi Honor Society.