FL-Trickle: New Enhancement of Trickle Algorithm for Low Power and Lossy Networks

Hanane Lamaazi[1], Nabil Benamar[2], Nassima EL Kahili[3], Tarik Taleb[4]

Lamaazi.hanane@gmail.com, n.benamar@est.umi.ac.ma, tarik.taleb@aalto.fi

1 Faculty of Sciences, 2 School of Technology, University Moulay Ismail, Meknes, Morocco

3 University Ibn Tofail, National School of Applied Sciences Kenitra, Morocco

4 School of Electrical Engineering, Aalto University, and Center for Wireless Communications, Oulu University, Finland

***Abstract*— The Trickle algorithm is one of the main components of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL). Trickle is used to maintain and to control messages in the network. However, this algorithm has some limitations in terms of power consumption, overhead, and convergence time. In this paper, we present a new improvement of the Standard Trickle algorithm, named Flexible Trickle Algorithm (FL-Trickle). Based on the T time parameters and the Minimum Interval values, the new trickle allows reducing the delay needed to transmit control messages as well as the transmission rate. A comparison has been made between the FL-Trickle, the standard Trickle and the Trickle-Plus algorithms. Simulation results show that our proposed approach outperforms both the standard trickle and the Trickle-Plus in terms of convergence time, overhead and energy consumption. FL-Trickle increases the convergence time by up 58%, reduces the overhead by up to 69% and the energy consumption by up to 62%.**

Keywords—Internet of Things; LLNs; RPL; Trickle Algorithm;

I. INTRODUCTION

The Low Power and Lossy Networks (LLN), standardized by the IETF ROLL working group [1] is a subclass of the Internet of things networks. It is formed mostly from heavily resource constrained and low-cost tiny devices and sensors.

The interconnection between routers is based on Lossy Links, which are unstable by nature and characterized by relatively low packet delivery rates. This kind of devices needs an efficient routing protocol to reach the Internet. RPL is an IPV6 proactive distance-vector routing protocol, which has several components used to construct the Destination-Oriented Directed Acyclic Graphs (DODAGs).

The first one is the objective function (OF) [2]. It defines how nodes select and optimize the routes toward the destination, while the second component is the trickle algorithm. In the RPL network, the trickle allows controlling the transmission of the signalling traffic used to build the DODAG topology [3] [4]. It allows exchanging information between constrained nodes in a simple way with an achievement of the scalability and energy-efficient goal. To this end, Trickle uses two mechanisms. The first one is when it detects an inconsistency in the network; it increases the signalling rate to be adaptable to this event. However, when the network reaches its steady phase it gradually and exponentially reduces the signalling rate for an efficient energy and bandwidth consumptions. The second mechanism is the suppression mechanism that allows removing the control packets if the node detects that enough number of its neighbours have transmitted the same information. This mechanism allows decreasing the energy consumption [5]. Many contributions have been proposed by researchers to improve the Trickle Algorithm. However, to the best of our knowledge, there is no proposed approach that combines the three metrics (Low overhead, low Convergence time and decreased energy consumption), which are more important in LLN networks. In this study, we propose a new extended trickle called FL-Trickle. Our proposed algorithm leads to a faster convergence time and a lower overhead and energy consumption.

The remaining of this paper is organized in the following fashion. Section II, presents an overview of RPL and Trickle algorithm. Section III, describes some studies related to the Trickle algorithm. Motivation and proposed enhancement are explained in Section VI. An assessment of the different results obtained from simulation is described in section V. Finally, we conclude the paper by discussing the results and proposing a future work.

II. RPL OVERVIEW

A. RPL

RPL is a routing protocol for Low Power and Lossy Networks (LLN), made specifically for Wireless Sensor Networks (WSN) in the Internet of Things [6]. It has a routing table to decide which neighbor node is the next hop for a given message. RPL operates by assigning each node a rank, which is increased as far as the node is from the border router. It allows creating a Destination Oriented Directed Acyclic Graph (DODAG), which is a graph of paths of communication through the network [7].

RPL works in two directions; upward routing from leaf nodes to the border router and downward routing from the border to any node.

There are three types of messages that RPL used for creating an optimized routing table [8] [9]; DODAG Information Objects (DIO) forms the DODAG and allows other nodes to discover an RPL instance and join it. DODAG Information Solicitations (DIS) solicits DIOs from other neighbor nodes. Finally, DODAG Destination Advertisement Objects (DAO) supports downward paths to parent or ancestor nodes.

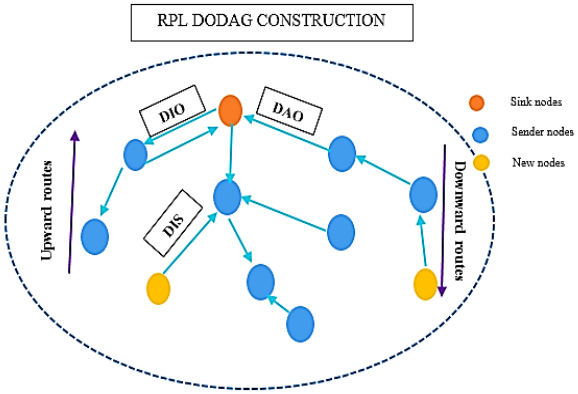

Fig 1: RPL DODAG Construction

RPL is based on the Trickle Timer Algorithm to control the DODAG graph and to construct and update it. The timer controls specifically the transmission of the DIO's messages that enter the network and also the time that a node listens for a new information and how it will be sending its own and current information to its neighbors.

B. Trickle Timer Algorithm

The trickle algorithm is developed to disseminate code updates and construct the DODAG through LLN Network. It runs for a defined interval and made up of three configuration parameters:

- ✓ ***Imin:*** The minimum interval size, which is defined in units of time (e.g., Milliseconds, seconds ...).
- ✓ ***Imax:*** The maximum interval size that considered a number of doublings of Imin, also defined in units of time (e.g., Milliseconds, seconds ..).
- ✓ ***K:*** presents the redundancy constant [10].

In addition to these parameters, three variables are maintained by Trickle:

- **I**: The current interval size defined in units of time (e.g., Milliseconds, seconds ...).
- **t**: a time within the current interval defined in units of time (e.g., Milliseconds, seconds ..).
- **c**: A counter.

To operate, the trickle algorithm uses six steps [10]: first, it starts its first interval from a range of [Imin, Imax]. Then, it resets the counter to zero and chooses a transmission time randomly from a range of [I/2, I]. In the third step, if the trickle detects a consistent it increments the counter. At a specific time "T", nodes can transmit only if the counter is less than the redundancy K and which forms the step 4. At an expired interval, trickle doubles its length. If the new length is high than the Imax, then the trickle sets this interval to Imax and returns to step 2. At least, step 6 is related to the inconsistent messages. If a node detects an inconsistent, it resets the interval length to Imin and re-executs step 2.

III. Related Works

The IETF has adopted Trickle Algorithm as a core component of RPL that is responsible for regulating the transmission of the control traffic data. In addition, Trickle has been used by various applications to disseminate data in service discovery protocols in an efficient way [13]. Despite its importance, only a few studies have been conducted to validate its efficiency and optimizing its operation. In [4], authors proposed an improved version of Trickle called “Enhanced-Trickle”. This algorithm solves the short listen problem. It changes three configuration parameters in the standard Trickle; First, it chooses the time "t" from within the range [0, I], instead of [I/2, I]. Then, the algorithm resets the counter "c" to zero only at the beginning of the first interval "Imin" and at the randomly chosen time "t" to enable the suppression mechanism. However, this produces unequal intervals among nodes. Hence, some nodes have more chances to transmit than others do. As a solution, authors proposed to stretch the redundancy of the constant "k" by using a new formula: $k = (k * (2 * Inz – I)) / I$. Therefore, if a node has two successive transmissions, "k" will increase until exceeding the redundancy counter "c". As a result, it allows decreasing the convergence time. In contrast, it still suffers from the unfairness problem, because it is rare when this formula is applied, so the algorithm will operate as the standard. Hence, some nodes will send more than others in most times. For instance, the authors in [11] and [12] analysed the underlying causes of unfairness in Trickle networks, addressing the unfairness of Trickle into two contributors: desynchronization and non-uniform topology. To balance Trickle's communications in heterogeneous topologies, the authors in [11] proposed an algorithm called "Multiple Redundancy Constants K" that computes the redundancy constants "k" locally at each node depending on the number of neighbours. In [12], the authors analysed the unfairness, linking the problem's cause to the desynchronization among nodes. Therefore, they proposed an algorithm called "Trickle-D" that adapts the redundancy constant "k" to improve global fairness based on the Jain's fairness index ensuring that all nodes have the same number of transmissions. Consequently, Trickle-D achieves high

fairness while keeping a low overhead. Another version of Trickle, called "New Elastic Trickle", is introduced in [13], in which the convergence time and the latency are improved. The authors examined the relationship between the number of neighbours and the listen-only period, discovering that when the number of neighbours is small, the listen-only period is short, and vice versa. In addition, they eliminated the listen-only period for the first interval; hence, nodes can resolve the inconsistency earlier. As a result, the New Elastic Trickle gets better results in term of convergence time. However, it increases the control traffic overhead in the first Interval due to the elimination of the listen-only period. Hence, it consumes more energy.

IV. Motivation and Contribution

1. *The proposed Algorithm (FL-Trickle)*

The Trickle algorithm is the main component of the RPL routing protocol. It allows controlling and regulating the load of the traffic in the network. However, Trickle has some limits that reduce its efficiency in terms of convergence time, overhead and energy consumption. To overcome these limitations, we propose a new enhancement of the Trickle called "FL-Trickle". We noticed that choosing a low value for Imin in the first intervals is a major issue [1]. Indeed, it lets the nodes sending many control messages to converge quickly. As a result, it introduces high control traffic overhead. For this reason, we aimed to find a way to converge as quickly as possible with low control traffic overhead and at the same time decreasing the power consumption. The proposed FL-Trickle is illustrated in Figure 2

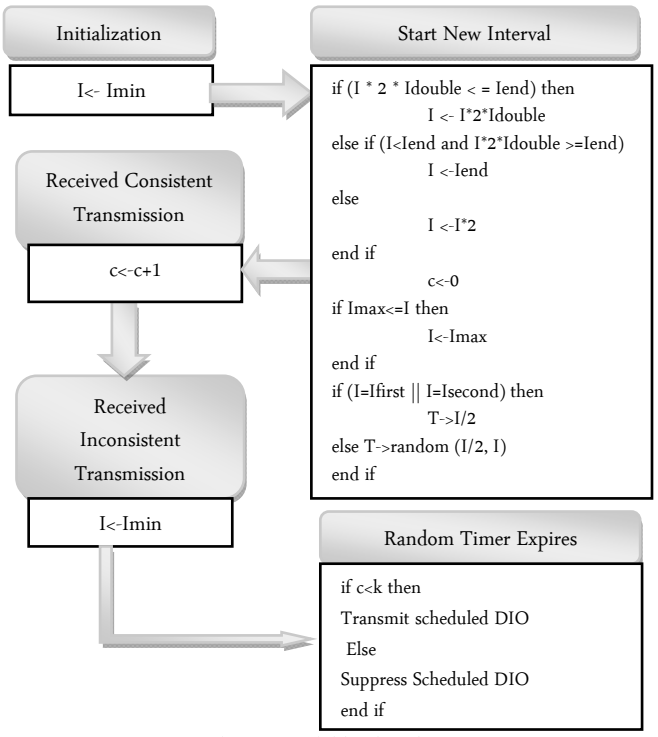

Fig 2: FL-Trickle Flowchart

The main contribution of this paper is that the new FL-Trickle decreases the delay to transmit the control messages by fixing the transmission time T at I/2, instead of choosing it randomly in [I, I/2]. Hence, our algorithm has a faster converge network. In additions FL-Trickle decreases the transmission rate by working with a high value of the Minimum Interval (Imin) in order to have a low overhead with no extra in terms of energy consumption. To decrease the energy consumption, FL-Trickle uses three parameters, the skipped intervals, the start and the end intervals of the skipped process.

V. Simulation Results

a) *Simulation setup:*

To assess our proposed FL-Trickle, we configured a set of parameters using the Cooja simulator running on the Contiki Operating System (version 2.7). The choice of this simulator is based on two main considerations: first, it is an open source emulator designed for IoT applications [14]. Second, it makes the modification and improvement of the trickle very easy due to its implementation on the core RPL in a separate way to the other components of RPL as the objective function. Table (1) describes the different parameters settings.

Tableau 1: Parameters setting using Cooja2.7

Simulator	**Cooja 2.7**
Simulation Duration	600 s
Simulation Area	100 m²
Imin	214
Imax	220
Data Packet Rate	60s
Reception Success Ratio RX value	100%, 80%, 60%, 40%, 20%
Transmission Success Ratio TX Value	100%
Transmission Range	50 m
Objective Function	MRHOF
Network Topology	Random
Interference Range	100 m

We have considered three metrics to evaluate our experiments: the convergence time wich is the time for nodes to join the network, the overhead that is the number of control messages and the energy consumption. Each one has an impact on the behaviour of the Trickle algorithm [15].

b) *Evaluation results:*

To evaluate the performance of FL-Trickle, we considered

two scenarios: different values of the probability of reception ratio and different values of redundancy K. A comparison has been made between the standardized Trickle, the Trickle-Plus, and the new FL-Trickle. The results are analysed for both cases as discussed here under

1) Comparison of different RX values

The first scenario is based on RX values. We compare FL-Trickle, Trickle-Plus, and Standard Trickle in terms of convergence time, energy consumption, and overhead for different RX values from 20% to 100%. As we can observe from Figure 3, FL-Trickle has the optimal convergence time with 100% and 80% of RX value than trickle plus and standard trickle. We notice that the convergence time of FL-Trickle scales logarithmically with the increase of RX values. However, Trickle-Plus provides a slow decrease of convergence time than Standard Trickle from 100% and 80% RX values but strongly decrease from 80% to 20%. Moreover, we notice that the convergence time changes are not proportional to the RX values. The decrease of the probability of reception (RX) provides an increase in terms of convergence time. We conclude that with a low probability of reception, the number of lost packets increases, making the network to converge very slowly.

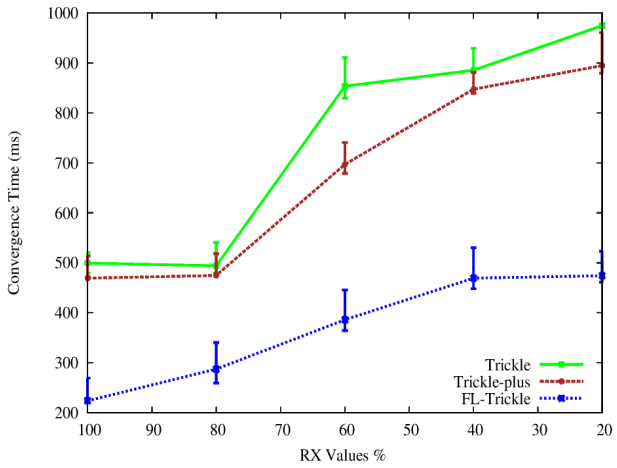

Fig 3: Convergence time Vs the probability of reception (RX)

In addition, the decrease of the RX value lets nodes to transmit more control messages as illustrated in figure 4. We notice that the standard Trickle provides a very high value of overhead due to the listen-only period as explained earlier. However, both FL-Trickle and Trickle-Plus have a low overhead, because they skip the undesired intervals, sending only the necessary control messages needed for the convergence. As we can observe, the decrease in RX values affect the network stability, making nodes to send more control messages. More control messages mean more energy consumption as illustrated in Figure 5. We also observe that FL-Trickle consumes low power than Trickle-Plus and Standard Trickle. This result refers to the skipped interval that reduces the unnecessary messages and the high value of the Imin intervals that balance the transmission load between nodes. By adopting these two parameters, FL-Trickle keeps providing better results in terms of convergence time, overhead and energy consumption. In contrast with the trickle standard, the use of low values of Imin allows for more energy consumption, which affects the network lifetime, making nodes consuming their energy in a short time.

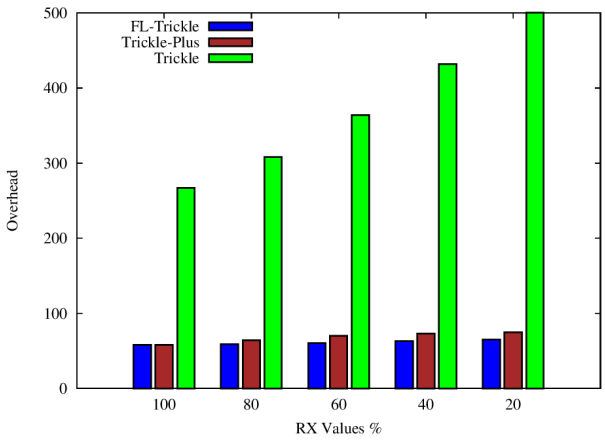

Fig 4: Overhead Vs the probability of reception (RX)

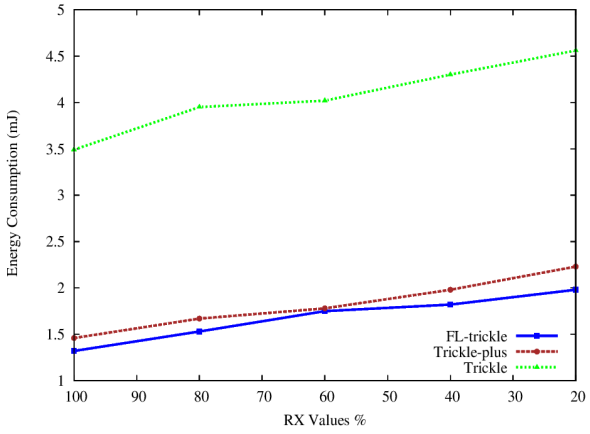

Fig 5: Energy consumption Vs the probability of reception (RX)

2) Comparison of different values of the Redundancy constant (K).

The second scenario focuses on the redundancy constant "K". The experiments demonstrate the impact of this parameter on convergence time, overhead and power consumption. As shown in Figure 6, FL-Trickle makes the network to converge faster than Trickle-Plus and Standard Trickle even if we change the redundancy constant (k). However, as we can observe from Figure 6, the redundancy k can affect the convergence time. The network that uses the Standard Trickle converges very slowly especially with high values of redundancy K. This result is similar to results in

Figure 7, we observe that when the redundancy k increases, nodes transmit more messages which causes congestion. To resolve this problem and then decrease the control messages, nodes should use the suppression mechanism. In contrast, this mechanism cannot be activated until the counter C is greater or equal to redundancy K. For this reason; the Standard Trickle has the highest overhead than Trickle-Plus and FL-Trickle. More overhead means more energy consumption as confirmed in Figure 8. We notice that the FL-Trickle consumes low energy than Trickle-Plus and Standard Trickle. In addition, the increase in redundancy k allows increasing the energy consumption. This can be explained by the fact that with a high value of redundancy k, nodes keep transmitting their data until the counter c attaints a high value or equal to these parameters. This can make the network converge slowly, increase the overhead and makes nodes consume more energy. As a conclusion, we notice that regardless the experiment condition, FL-Trickle provides better results in terms of convergence time, overhead and energy consumption.

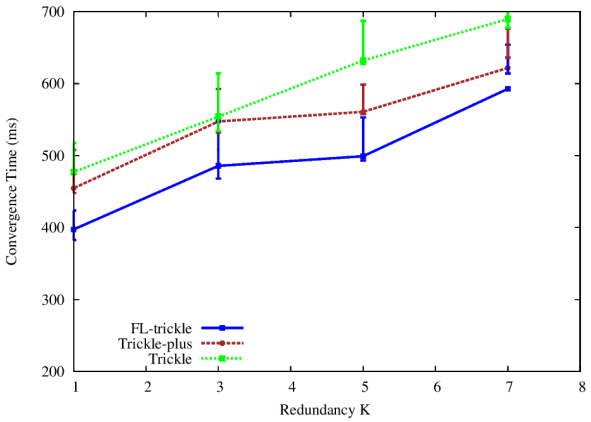

Fig 6: Convergence time Vs the Redundancy Constant (K).

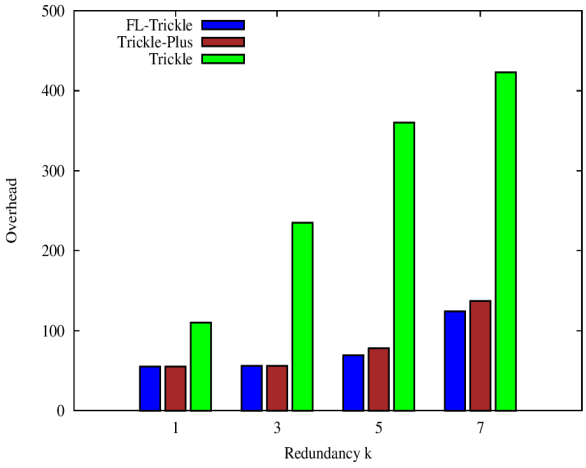

Fig 7: Overhead Vs the Redundancy Constant (K).

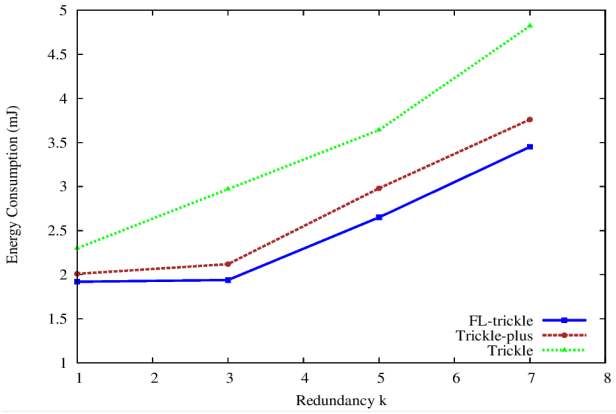

Fig 7: Energy Consumption Vs the Redundancy Constant (K).

ACKNOWLEDGMENT

This work has been supported by the University of Moulay Ismail, Grant Project ITIC-TRANSPORT.

REFERENCES

[1] B. Ghaleb, A. Al-dubai, E. Ekonomou, B. Paechter, and M. Qasem, "Trickle-Plus : Elastic Trickle Algorithm for Low- Power Networks and Internet of Things , Trickle-Plus : Elastic Trickle Algorithm for Low- Power Networks and Internet of Things," *Wirel. Commun. Netw. Conf. (WCNC), 2016 IEEE*, no. April, 2016.

[2] H. Lamaazi and N. Benamar, "OF-EC: A novel energy consumption aware objective function for RPL based on fuzzy logic," J. Netw. Comput. Appl. https//doi.org/10.1016/j.jnca.2018.05.015, 2018.

[3] C. Vallati and E. Mingozzi, "Trickle-F : fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol," *Sustain. Internet ICT Sustain. (SustainIT),* 2013.

[4] B. Ghaleb, A. Al-dubai, and E. Ekonomou, "E-Trickle : Enhanced Trickle Algorithm for Low-Power and Lossy Networks," *IEEE Int. Conf. Comput. Inf. Technol. Ubiquitous Comput. Commun. Dependable, Auton. Secur. Comput. Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM),* pp. 1–7, 2015.

[5] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle : A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," *Tech. Rep.*, 1947.

[6] H. Lamaazi, N. Benamar, A. J. Jara, L. Ladid, and D. El Ouadghiri, "Challenges of the Internet of Things : IPv6 and Network Management," in *International Workshop on Extending Seamlessly to the Internet*

of Things (esIoT-2014), Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on, 2014, pp. 328–333.

[7] H. Lamaazi, N. Benamar, M. I. Imaduddin, and A. J. Jara, "Performance Assessement of the Routing Protocol for Low Power and Lossy Networks," in *the International Workshop WINCOM (Wireless Networks and mobile COMmunications) in Marrakech, Morocco*, 2015.

[8] H. Lamaazi and N. Benamar, "RPL Enhancement using a new Objective Function based on combined metrics," *13th Int. Wirel. Commun. Mob. Comput. (IWCMC); Val. spain*, pp. 1459–1464, 2017.

[9] H. Lamaazi, N. Benamar, M. I. Imaduddin, and A. J. Jara, "Mobility Support for the Routing Protocol in Low Power and Lossy Networks," in *The 30th IEEE International Conference on Advanced Information Networking and Applications (AINA-2016) Le Régent Congress Centre, Crans-Montana, Switzerland,* 2016.

[10] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," *RFC6206*, pp. 1–13, 2011.

[11] T. Coladon, M. Vucinic, and B. Tourancheau, "Multiple Redundancy Constants with Trickle," *IEEE 26th Annu. Int. Symp. Pers. Indoor, Mob. Radio Commun. (PIMRC),* pp. 1951–1956, 2015.

[12] M. Vucinic, M. Krol, 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, 2017.

[13] M. B. Yassein, S. Aljawarneh, and E. Masadeh, "A new elastic trickle timer algorithm for Internet of Things," *J. Netw. Comput. Appl.*, no. January, pp. 1–10, 2017.

[14] H. Lamaazi, N. Benamar, and A. J. Jara, "Study of the Impact of Designed Objective Function on the RPL-Based Routing Protocol," *Third Int. Symp. Ubiquitous Networking, Unet, Adv. Ubiquitous Netw. 2 pp 67-80*, pp. 67–80, 2016.

[15] H. Lamaazi, N. Benamar, and A. J. Jara, "RPL-Based Networks in static and mobile environment: a performance assessment analysis," *J. King Saud Univ. Inf. Sci.*, 2017.