International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 3
ISSN: 2454-4248
563 – 567

A Survey on Trickle Algorithm: Comparative Analysis

Jinee Goyal
PG Scholar, Department of Computer Science
Punjab Engineering College [Deemed to be University]
Chandigarh, India
jinee.goyal@gmail.com

Dr. Trilok Chand Aseri
Professor, Department of Computer Science
Punjab Engineering College [Deemed to be University]
Chandigarh, India
trilokchand@pec.ac.in

***Abstract*—** Internet of Things (IoT) is an emerging area in the field of wireless communication. Due to its resource constraint environment, IETF gave a standard for IVP6 routing protocol for low power and lossy networks (RPL). The major component of RPL is Trickle algorithm. It is used to control the number of messages exchanged between devices and helps in early network stabilization. Due to its importance, it is crucial for researchers to understand this protocol. The absence of surveys in Trickle Algorithm motivates us to write this paper. In this paper, we compared different Trickle Algorithms based on performance parameters like convergence time, energy consumption, packet delivery ratio and others. Concluding, we can say that it is open research area in the designing parameters of Trickle's Algorithms and we believe that this survey will be beneficial for researchers in their relevant work.

Keywords*-IoT; RPL; Trickle Algortihm; Wireless Communication*

*****

I. INTRODUCTION

In a previous couple of years, there has been an extraordinary advancement in the field of wireless technology. There is a wide scale increase in the inventive applications and it is assumed that the number of devices and embedded gadgets will increment immensely in near future [13]. All these low powered devices which are connected through wireless communication are connected to the internet as Internet of Things (IoT). In IoT [8], a large number of resource-constrained devices are connected with wired or wireless links. These links are commonly known as low power and lossy networks. A robust routing protocol is needed to manage these resource-constrained devices. In these types of devices, energy consumption is a major issue. As these devices are battery operated, there should be minimum energy consumption with maximum functionality. Keeping in mind all these issues, IETF standardized a routing protocol for low power and lossy networks (RPL). This algorithm was specially designed to work in energy-constrained networks.

RPL uses trickle Algorithm to discover routes across the network. Trickle Algorithm finds routes with minimum energy consumption and convergence time. In this paper, we considered different versions of Trickle Algorithm and studied them extensively based on different parameters like route quality, energy consumption, convergence time, packet delivery ratio and others. Our goal is to comprehensively study different versions of Trickle Algorithm and summarizing advantages and improvements of each.

This paper is sorted out into sections as mentioned. Section II gives an overview of RPL and how it works. Section III discusses what original Trickle Algorithm is and how it works. Section IV compares different versions of Trickle Algorithm depending upon certain parameters and finally, Section V summarizes the paper in the form of conclusion and future scope.

II. RPL

RPL is a routing protocol specially intended for resource constrained, battery operated devices [10]. It maximizes its functionality with minimal energy overhead. RPL works on IP layer and can be used in multiple types of link layers [7]. RPL follows distance vector routing algorithm and maintains Destination Oriented Directed Acyclic Graphs (DODAG's). Destination oriented DAG (DODAG) is basically a DAG but with root as sink only and is constructed by using various constraints such as different routing metrics, objective functions etc. In this protocol, nodes have the capability to organize themselves by forming a tree topology with the root at the sink.

A. *Types of Control Messages*

DODAG's are constructed by using signaling or the control traffic composed of three different messages as follows:

1) DODAG Information Solicitation (DIS): This message is multicast and is sent when a node wants to join the network so that a node can solicit a DIO from an RPL node.

2) DODAG Information Object (DIO): This message is first multicast by the root node (or sink) and then is periodically multicast by each and every node of the DODAG to construct the DODAG. Each node sends DIO messages to advertise themselves in the network and update its topology accordingly.

3) Destination Advertisement Object (DAO): This message is a unicast message. It is used to propagate destination related information upwards along with the DODAG. When a node receives a DAO, it updates its routing table.

B. *Modes of RPL*

Intermediate nodes work in two modes [5]:

1) Storing Mode: In this, intermediate nodes maintain a routing table for information storage.

2) Non-Storing mode: In this, intermediate nodes have less number of resources so they do not store routing information, instead pass as it is received.

III. TRICKLE ALGORITHM

Trickle Algorithm [14] is a scheduling method which is used in route discovery by helping in the creation of DODAG in RPL. Trickle Algorithm provides simple, energy efficient and scalable method for information exchange [3]. It is standardized by IETF to control Data Information Objects (DIO's). DIO's are control traffic messages which are used to

 ISSN: 2454-4248
563 – 567

create upward routes in RPL routing protocol. It exchanges information through two different mechanisms. First, it adaptively increases the signaling rate when inconsistency is detected so as to remove this inconsistency as early as possible. When it reaches the consistent state, it starts reducing the signaling rate gradually and exponentially to save energy. Second, to reduce energy consumption, trickle uses suppression mechanism to keep a check on the number of control packets being transferred. In this, a node suppresses the transmission of control packets if they are already enough in the network with the same piece of information.

Trickle Algorithm uses three parameters [12]:

- Minimum size of interval (I_{min})
- Maximum size of interval (I_{max})
- Redundancy counter (k)

It also maintains three variables:

- Size of the current interval (I)
- Random time in the current interval (t)
- Counter (c)

A. *Algorithm*

- **Step 1:** The trickle starts its execution by setting the value of I in the range of [I_{min}; I_{max}].
- **Step 2:** It resets counter c as 0 and set t randomly in the range of [$I/2$; I].
- **Step 3:** If the received message is consistent, it will increment the value of c.
- **Step 4:** It transmits DIO only if $c < k$ otherwise it will follow suppression mechanism.
- **Step 5:** Trickle Algorithm doubles the length of the interval after the expiration of a timer. If the generated new interval size is greater than I_{max}, it resets the value of I to I_{min}.
- **Step 6:** If the received message is in inconsistent form, it resets the trickle timer i.e. it will set I as I_{min} and will start the new interval as in Step 2.

Trickle is easy to implement and requires little energy and still works efficiently but it has a major problem of listen only period. Initially, there was a problem of short listen period, to resolve this issue listen only period was introduced. If there was no provision of listen only period, nodes would start transmitting directly which would cause redundant transmissions. But, listen only period has its own limitation as it induces a delay in the stabilization of nodes i.e. it is increasing convergence time of the network. As a result, more time is needed to detect and resolve inconsistencies. Another issue is load balancing; some nodes may not get a chance to resolve their inconsistencies. Because of these issues, different improvements have been suggested by different authors based on different parameters which are discussed below showing how they are affecting the overall behavior of trickle algorithm.

IV. LITERATURE SURVEY

A. *F-Trickle Algorithm*

In [7], authors solve the problem of Load Balancing in original Trickle Algorithm. Original Trickle Algorithm can lead to suboptimal route formation. This effect is more prominent when the number of suppressed messages is high. In this, authors gave the concept of fair broadcast suppression. Each DIO message contains different information, and the original Trickle Algorithm does not provide any mechanism to decide which message should be suppressed, so it may arises a case that some nodes are not able to transmit DIO messages which may lead to some routes left undiscovered. It becomes important that each node should get a chance to broadcast fairly. In this, authors prioritize each node by checking the number of consecutive suppressions of each node by introducing a new variable 's' to record the number of suppressions. Each node gets a priority which is proportional to 's'. Higher the value of 's' (i.e. more suppressed messages), higher are the chances that node will get a chance to transmit. Changes have also been made to listening and transmitting times to ensure that prioritizations are fair and nodes have a chance to be free from short listen problem. Complete Algorithm is explained in Table 1.

TABLE I. F-TRICKLE ALGORITHM

Algorithm 1: F-Trickle Algorithm

- $I = I_{\min}$
- $s = 0$
- $c = 0$

$$t = random\left(\frac{I}{2^{s+1}}, \frac{I}{2^{s}}\right)$$

- If received message is consistent

$$c = c + 1$$

- If $k \geq c$

  Transmit DIO control message

$$s = 0$$

  Else

$$s = s + 1$$

- $c = 0$
- If received message is inconsistent

$$I = I_{\min}$$

$$s = 0$$

  Else

$$I = I \times 2$$

  If $I_{\max} \leq I$

$$I = I_{\max}$$

where:

- s record number of suppressions
- c counter
- t random time in the current interval
- k redundancy counter
- I current interval size

B. *E-Trickle Algorithm*

In [15], authors give an idea of reducing convergence time and solve the short listen problem without using the concept of listen only period. The authors discuss different reasons that are responsible for high convergence time. The authors state that turning down suppression mechanism is caused by nodes ignoring all the received messages at the end of random interval i.e. where c is set to 0 after each interval, so author introduced the concept of maintaining track of all received messages till the end of the interval. Different modifications have been made by the author in the algorithm. As a result, authors were successful in solving the problem of short listen and listen only period problem. The problem of turning down the suppression mechanism was reduced to a great extent. Authors, in this paper, by comparing results of E-Trickle

 ISSN: 2454-4248
563 – 567

algorithm with original trickle algorithm, clearly shows that newly designed algorithm performs better when compared with respect to convergence time. This reduction in convergence time is due to the elimination of listen only period. The E-Trickle Algorithm also reduces the probability of collisions. In terms of other parameters like energy consumption, packet delivery ratio, their performance is comparable. The new proposed algorithm was able to discover better optimal routes as compared to original trickle algorithm in less time and almost same cost. E-Trickle Algorithm outperforms the original Trickle Algorithm up to 43%. One of the most important characteristics of this new algorithm is the amount of energy saved and authors also proved that suppression mechanism is not the only key factor to measure energy consumption. Complete Algorithm is explained in Table 2.

TABLE II. E-TRICKLE ALGORITHM

Algorithm 2: E-Trickle Algorithm

- $I = I_{\min}$
- $c = 0$
- If $I_{\max} \le I$
  $I = I_{\max}$
- $t = random(0, I)$
- If received message is consistent
  $c = c + 1$
- If received message is inconsistent
  $I = I_{\min}$
  $c = 0$
- If $I_{nz} > I$
  $k_n = \frac{(k \times (2 \times I_{nz} - I))}{I}$
  Else
  $k_n = k$
- If $c < k_n$
  Transmit DIO message
  Else
  Supress DIO message
- $c = 0$

where:

- I_{nz} new interval size
- k_n *new redundancy counter*

C. *Adaptive Trickle Algorithm*

In [4], authors gave the idea that each node should individually adjust their suppression mechanism depending upon the density of the node. In Original Trickle Algorithm, the value of k is fixed. Due to this, some nodes may get the high chance of transmitting as compared to other which may lead to more number of delays. This also leads to suboptimal route formation and nodes try to connect to first available node even if better options are available. That is why, authors gave the idea of changing the value of 'k' with changing node densities i.e. if a node has large number of neighbors then it will set high value of k so that it can compete with medium, otherwise set low value of k to give other nodes a better chance of transmission.

Instead of executing Step 4 of Original Trickle Algorithm, execute it with minor changes as shown in Table 3:

TABLE III. ADAPTIVE TRICKLE ALGORITHM

Algorithm 3: Adaptive Trickle Algorithm

- $k = fun(c)$
  Where
  $$fun(c) = \begin{cases} k_{\min}, \alpha c < k_{\min} \\ floor(\alpha c), k_{\min} \le \alpha c \le k_{\max} \\ k_{\max}, \alpha c > k_{\max} \end{cases}$$
- $I = \min(2I, I_{\max})$

D. *Optimized Trickle Algorithm*

In [11], authors tried to decrease the latency in the network without introducing any additional overhead to scalability and robustness. In order to achieve this, authors gave the idea of setting transmitting time t in range [0, *Imin*) instead of [*Imin*/2,*Imin*). These changes allow Trickle Algorithm in early detection and resolution of inconsistencies. There will be no problem of short listen period. The results of simulation show exceptionally good results. The authors showed that this improvement led to success ratio of about four times as compared to Original Trickle Algorithm in even 90% loss rate. This improvement led to some extra cost but authors also validated the reason for this i.e. issue of synchronization. In Original Trickle Algorithm, transmissions were synchronized with other intervals but opt-trickle does not provide this feature. Instead of executing Step 2 of Original Trickle Algorithm follow one of two Steps as shown in Table 4:

TABLE IV. OPTIMIZED TRICKLE ALGORITHM

Algorithm 4: Optimized Trickle Algorithm

- If Step 6 of Original Trickle Algorithm is executed
  $c = 0$
  $t = [0, I_{\min})$
- If Step 5 or Step 1 of Original Trickle Algorithm is executed
  $c = 0$
  $t = [I/2, I)$

E. *Trickle-Plus Algorithm*

In [1], authors gave the idea of increasing the elasticity of the Original Trickle Algorithm. In this newly proposed Algorithm, authors introduced three parameters. These parameters will help to improve the elasticity as it will allow algorithm to directly jump to required interval without having to go through intermediate intervals. First, Shift Factor (*SF*), it tells us about the number of intervals an algorithm should skip. Second is IShift Start (I_{SE}), it gives the value of interval in which algorithm would start shifting. Third is IShift End (I_{SF}), it gives the value of interval in which algorithm should shift shifting. Algorithm is explained in Table 5.

TABLE V. TRICKLE-PLUS ALGORITHM

Algorithm 5: Trickle-Plus Algorithm

- $I = I_{\min}$
- $If\ (I \times 2 \times SF \le I_{SE})$

 ISSN: 2454-4248
563 – 567

$I = I \times 2 \times SF$
$elseif(i < I_{SE} and I \times 2 \times SF \geq I_{SE})$
$I = I_{SE}$
$else$
$I = I \times 2$

- $I = I \times 2$
- $c = 0$
- If $I_{\max} \leq I$
  $I = I_{\max}$
- $t = random(I/2, I)$
- If received message is consistent
  $c = c + 1$
- If received message is inconsistent
  $I = I_{\min}$
- If $c < k_n$
  Transmit DIO message
  Else
  Suppress DIO message

F. *ME-Trickle Algorithm*

In [9], the authors extended the idea from E-trickle Algorithm. Instead of doubling the interval each time a consistent message is received, it directly jumps to *Imax* value. This improvement resulted in a very less number of packets transmitted as compared to all other algorithms (Original Trickle Algorithm, Opt-Trickle Algorithm, E-Trickle Algorithm). This is because whenever a consistent message is detected, it stops increasing time exponentially. Its convergence time is very less for low range values. For high range values, its convergence time is comparable. Algorithm is as explained in Table 6.

TABLE VI. ME-TRICKLE ALGORITHM

Algorithm 6: ME-Trickle Algorithm

- $I = I_{\min}$
- $c = 0$
- $I = I_{\max}$
- If $I_{\max} \leq I$
  $I = I_{\max}$
- $t = random(0, I)$
- If received message is consistent
  $c = c + 1$
- If received message is inconsistent
  $I = I_{\min}$
  $c = 0$
- If $I_{nz} > I$
  $k_n = \frac{(k \times (2 \times I_{nz} - I))}{I}$
  Else
  $k_n = k$
- If $c < k_n$
  Transmit DIO message
  Else
  Supress DIO message
- $c = 0$

G. *Improved Trickle Algorithm*

In [6], the authors present an idea to further improve the F-Trickle Algorithm. This is done by setting the redundancy counter to zero only when a node is transmitting or suppressing the messages instead of setting at beginning of each interval. Due to this change, previously DIO's (transmitted or suppressed) will also be taken into account which would lead to low power and energy consumption as compared to Trickle-F Algorithm. As authors did not make any changes in the number of DIO's, packet delivery ratio (PDR) remains the same. Table 7 shows the detailed explanation of algorithm.

TABLE VII. IMPROVED TRICKLE ALGORITHM

Algorithm 7: Improved Trickle Algorithm

- $I = I_{\min}$
- $s = 0$
- $c = 0$
  $t = random\left(0, \frac{I}{2^s}\right)$
- If received message is consistent
  $c = c + 1$
- If $k \geq c$
  Transmit DIO control message
  $s = 0$
  $c = 0$
  Else
  $s = s + 1$
- $c = 0$
- If received message is inconsistent
  $I = I_{\min}$
  $s = 0$
  $c = 0$
  Else
  $I = I \times 2$
  If $I_{\max} \leq I$
  $I = I_{\max}$

H. *Summary*

Different Trickle Algorithms have been explained above. Each Algorithm provides a different technique to improve the performance of Original Trickle Algorithm, may it be in terms of Latency, PDR, Convergence time, Power Consumption or Energy Consumption. Table 8 shows the comparative analysis of all the Algorithms explained above.

TABLE VIII. COMPARISION OF DIFFERENT TYPES OF TRICKLE ALGORITHM.

TYPES	DESCRIPTION		
	Total Number of Packets Transferred	***Network Convergence Time***	***Power/Energy Consumption***
Original Trickle [2] [6] [9] [15]	Less as short listen problem is resolved but	More due to introduction of Listen Only	More than E-Trickle and Adaptive Trickle

 ISSN: 2454-4248
563 – 567

	more than other types mentioned below	Period	Algorithm
F-Trickle [1] [6] [7] [15]	Less due to prioritization of nodes	Less than Original Trickle Algorithm but more than E-Trickle Algorithm	Same as that of Original Trickle Algorithm
E-Trickle [1] [6] [9] [15]	Same as that of Original Trickle Algorithm	Less due to elimination of Listen Only Period	Less because of reduction in the probability of collisions
Adaptive Trickle [1] [4] [6] [15]	Less number of transmissions and better route discovery	Almost same in dense network but less in sparse network	Less energy consumption than Original Trickle Algorithm
Optimized Trickle [1] [6] [9] [11] [15]	Same as that of Original Trickle Algorithm	Faster Network Convergence than Original Trickle Algorithm	No additional overhead
Trickle-Plus [1] [6]	Less because interval time directly jumps to required interval without having to go through intermediate intervals	Optimal	High due to traffic overhead
ME-Trickle [6] [9]	Very less as interval directly jumps to I_{max} values	Lowest among all but only for low values of I_{min}	Minimum due to fewer packet transmissions
Improved Trickle [6]	Same as that of F-Trickle Algorithm	Same as that of F-Trickle Algorithm	Low as compared to F-Trickle Algorithm

V. Conclusion and Future Scope

In this paper, we discussed different improvements that have been made to original Trickle Algorithm based on certain parameters like energy consumption, power consumption, number of packets transmitted and convergence time. Simulation results shown in different papers suggest that improvements can be made in future for better results. So, it is an open research area which is a subject of the future work**.**

Acknowledgment

I would like to acknowledge Dr. Trilok Chand Aseri, Professor of Department of Computer Science at Punjab Engineering College as the co-author of this survey research and I am very grateful for his valuable comments.

I would also like to thank my parents (Raman Goyal, Meenu Goyal) for their support and fellow researchers (Ashima Khosla, Vishal Lamba) for their great assistance while researching and writing this paper. This research would have never been possible without them. Thank You.

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," no. MEIoT, pp. 1–6, 2016.

[2] E. Polytechnique and A. R. Corporation, "The Trickle Algorithm," pp. 1–13, 2011.

[3] T. M. M. Meyfroyt, "An analytic evaluation of the Trickle algorithm: Towards efficient, fair, fast and reliable data dissemination," *Proc. WoWMoM 2015 A World Wirel. Mob. Multimed. Networks*, no. July, 2015.

[4] T. M. M. Meyfroyt, M. Stolikj, and J. J. Lukkien, "Adaptive broadcast suppression for Trickle-based protocols," *Proc. WoWMoM 2015 A World Wirel. Mob. Multimed. Networks*, pp. 0–8, 2015.

[5] O. Iova, G. Pietro Picco, T. Istomin, and C. Kiraly, "RPL : The Routing Standard for the Internet of Things … O r I s I t?," *IEEE Commun. Mag.*, no. December, pp. 16–22, 2016.

[6] S. Goyal and T. Chand, "Improved Trickle Algorithm for Routing Protocol for Low Power and Lossy Networks," *IEEE Sens. J.*, vol. 1748, no. c, pp. 1–1, 2018.

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

[8] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things—A survey of topics and trends," *Inf. Syst. Front.*, vol. 17, no. 2, pp. 261–274, 2015.

[9] A. M. Q. Project, "A Performance Evaluation of RPL with Variations of the Trickle Algorithm," 2016.

[10] O. Gaddour and A. Koubâa, "RPL in a nutshell: A survey," *Comput. Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

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

[12] H. Guo, K. Zheng, F. Ouyang, X. Gan, Z. Zhang, and P. Dong, "A Terminable Trickle Algorithm for Lossy Networks," no. c, pp. 221–226, 2016.

[13] M. Zhao, A. Kumar, P. H. Joo Chong, and R. Lu, "A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities," *Peer-to-Peer Netw. Appl.*, vol. 10, no. 5, pp. 1232–1256, 2017.

[14] T. Clausen, A. C. De Verdiere, and J. Yi, "Performance analysis of Trickle as a flooding mechanism," *Int. Conf. Commun. Technol. Proceedings, ICCT*, no. Lix, pp. 565–572, 2013.

[15] B. Ghaleb, A. Al-Dubai, and E. Ekonomou, "E-Trickle: Enhanced Trickle algorithm for low-power and lossy networks," *Proc. - 15th IEEE Int. Conf. Comput. Inf. Technol. CIT 2015, 14th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2015, 13th IEEE Int. Conf. Dependable, Auton. Se*, pp. 1123–1129, 2015.