Modelling and Simulating the Trickle Algorithm

Markus Becker, Koojana Kuladinithi, and Carmelita Görg

Communication Networks, TZI
University Bremen, Bremen, Germany
{mab,koo,cg}@comnets.uni-bremen.de

Abstract. The Trickle algorithm has proven to be of great benefit to the Wireless Sensor Networking area. It has shown general applicability in this field, e.g. for code distribution to smart objects and routing information distribution between smart objects. Up to now analysis of the algorithm has focussed on simulation studies and measurement campaigns. This paper introduces an analytical models for the algorithm's behaviour for the time to consistency. The model is compared with simulation results for a set of network topologies and enables to discover efficient settings of the algorithm for various application areas, such as logistics.

Keywords: Trickle algorithm, RPL, WSN, Analytical model, Simulation, TOSSIM.

1 Introduction

The Trickle algorithm has been proposed in [4] in order to effectively and efficiently distribute code in a Wireless Sensor Network (WSN). The algorithm itself however is more generally applicable: it tries to create consistency of information in a distributed network. The definition of consistency is left to the user of the algorithm. Currently, the algorithm is employed in the routing protocol IPv6 Routing Protocol for Low power and Lossy Networks (RPL) [6] for the distribution of Destination Oriented Directed Acyclic Graph Information Objects (DIO). Because of the general applicability of the algorithm, the definition of Trickle has been split out into its own IETF RFC 6206 [3]. The algorithm has been studied by means of simulations before, an analytical model hasn't been published yet. By means of an analytical model, appropriate settings for the Trickle algorithm to be used in the user application scenarios and for the envisioned application demands can be derived and trade-offs between propagation time and the number of sent packets can be found.

1.1 Trickle Algorithm

The Trickle algorithm is a simple, yet elegant and powerful algorithm. It consists of the following 3 variables[1]:

[1] The notation is according to [4], the notation in [3] has slightly changed.

τ Communication interval length
T Timer value in range $[\tau/2, \tau]$
C Communication counter

The algorithm can be parameterized with the following three options:

K Redundancy constant
τ_L Lowest τ
τ_H Highest τ

The algorithm consists of the following 2 transmission rules and 2 receiption rules:

- τ expires
  $\rightarrow$ Double τ, up to τ_H, pick a new T from range $[\tau/2, \tau]$
- T expires
  $\rightarrow$ If C < K, transmit

- Received consistent data
  $\rightarrow$ Increment C
- Received inconsistent data
  $\rightarrow$ Set τ to τ_L. Reset C to 0, pick a new T from $[\tau/2, \tau]$

With those basic rules, the algorithm adapts its communication well to different network densities and consistency churns.

Trickle regulates the nodes' sending rate in such a way that it sends frequently, when the density of nodes is low; it sends rarely, when the density is high. When there is a lot of inconsistency churn, Trickle tries to propagate the information fast with a high rate, but backs off to a lower rate when the information is detected to be consistent. Additionally, Trickle does not exhibit the problem of broadcast storms as simple Flooding does.

2 Scenarios

In order to study the behaviour of the Trickle algorithm, several scenarios have been set up to cover the most common network topologies. The scenarios are described in the following sections. The parameters that can be controlled for the scenarios are the number of nodes in the scenario and the scenario size. Additionally to the scenarios listed below, the authors have also setup a Random and a real testbed scenario (as well as models for the number of packets sent by Trickle), whose details and results cannot be presented due to page limitations.

2.1 Line Scenario

In this type of scenario, the topology consists of all nodes arranged only on one axis in a line. All nodes are connected according to the Closest-Pattern Matching (CPM) propagation model [2]. This scenario will be referenced by the name 'Line-CPM'.

2.2 Grid Scenario

The nodes are aligned regularly in a grid. Each node has the same distance to its closest neighbors. This scenario will be referenced by the name 'Grid-CPM'.

3 Simulation Tool

In order to validate the analytical model, a simulation has been setup. The TinyOS simulation tool TOSSIM has been used in combination with an implementation of the Trickle algorithm in the application layer. The lower layers conform to the Berkeley Low-Power IP (blip-1.0) stack, which has been modified to be simulatable[2]. Note that blip's built-in Trickle timer in its ICMP implementation has not been part of this study, solely the application layer Trickle instance has been evaluated. The simulations were performed with up to 300 Monte-Carlo repetitions for each scenario instance with varying seeds in the TinyOS executable as well as the Python TOSSIM script. The simulation tool can be used with the previously mentioned scenarios. A simulation suite run consists for the 'Line', 'Grid' and 'Random' scenario types of instances of those scenarios with varying number of nodes and inter-node distances.

4 Analytical Model

The main factors governing the efficiency of the Trickle algorithm are the number of messages issued by the algorithm and the delay until the network has reached consistency. Analytical models have been created for both metrics, in the submission only the analytical model for the delay will be presented.

4.1 Consistency Delay

The model for the complete network consistency delay distribution is created from the individual nodes' delay distribution.

The analytical model is based on the fact that the Trickle algorithm draws uniformly distributed pseudo random numbers between $\frac{\tau}{2}$ and τ. If an inconsistency is detected, the algorithm immediately sets $\tau = \tau_L$.

For the simplest scenario 'Line-CPM' as described in section 2.1, the consistency model can be setup in the following way. The seed of the inconsistency is the 0th hop. It does not draw a random number, but immediately knows the consistent information, thus this results in a Dirac impulse at $t = 0$. That particular node chooses its time to send the inconsistent information uniformly distributed between $\frac{\tau_L}{2}$ and τ_L, cf. figure 1. The 1st hop neighbor (assuming a perfect link for the moment and neglecting processing and communication time) thus detects the inconsistency uniformly distributed between $\frac{\tau_L}{2}$ and τ_L. For the

[2] The authors have also modified the upcoming version of blip, so that it can simulated with TOSSIM.

2nd hop 2 uniformly distributed random variables are added, the convolution of the 2 random variables leads to a triangle distribution as shown in figure 2. The 3rd hop adds another uniformly distributed random variable resulting in the bell shape shown in the same figure. The central limit theorem states that the mean of a summation of independent and identically distributed random variables, each with finite mean and variance, will be approximately normally distributed. For larger number of hops, the node consistency distribution will become normally distributed.

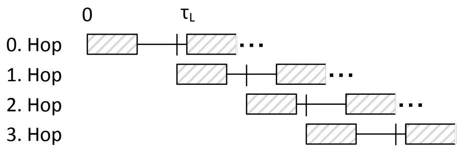

Fig. 1. Consistency Delay Addition

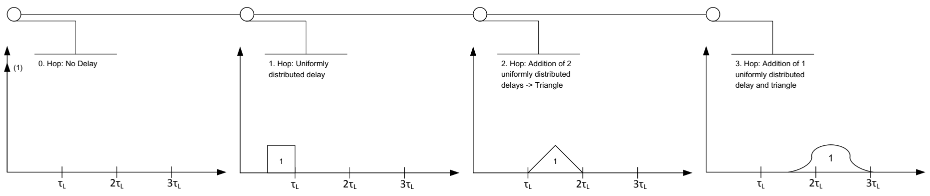

Fig. 2. Node Consistency Delay Distribution

The network-wide distribution of the consistency delay can be deducted from the individual distributions by a normalization of the sum of all the node consistency distributions. For the particular scenario, that process is depicted in figure 3.

The probability density function (pdf) of the time to consistency scenario can be modeled in detail by

$$p(t) = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{h=0}^{N-1} \sum_{w=0}^{W} \sum_{r=0}^{1} h_{n,h,w,r}(t) * p_{n,h,w,r}(t), \tag{1}$$

where n: node; h: hops; w: way; r: 'retransmission'/ next Trickle cycle.

The individual nodes' delay pdf in 1 can be calculated according to:

$$p_{n,h,w,r}(t) = \begin{cases} \delta(t) & , n = 0, \\ \mathcal{L}^{-1}\{\mathcal{L}\{\Theta(t - \frac{\tau_L}{2}) - \Theta(t - \tau_L)\}^h\} & , n \geq 1, r = 0, \\ \mathcal{L}^{-1}\{\mathcal{L}\{\Theta(t - \frac{\tau_L}{2}) - \Theta(t - \tau_L)\} * & \\ \mathcal{L}\{\Theta(t - 2\tau_L) - \Theta(t - 3\tau_L)\}^{h-1}\} & , n \geq 1, r = 1. \end{cases} \tag{2}$$

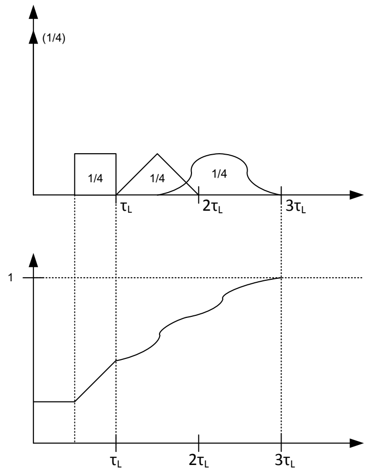

Fig. 3. Probability Density and Cumulative Distribution Function of the Network Consistency Delay Distribution

$\Theta(\cdot)$ denotes the Heaviside step function. $\mathcal{L}$ denotes the Laplace transform and $\mathcal{L}^{-1}$ denotes the inverse Laplace transform. For $n \geq 1$ this equation denotes a repeated folding of a unit step function. For $n = 1$ the unit step is broader than for $n = 0$.

The derivation of the weighting factor $h_{n,h,w,r}(t)$, which describes how often a certain distribution is represented in the network delay distribution is scenario dependent. In the following the steps to calculate $h_{n,h,w,r}(t)$ are shown for a 4 node Line-CPM scenario.

Figure 4 shows the Packet Receive Ratio (PRR) depending on the distance as employed by the Closest Pattern Matching (CPM) propagation model of TOSSIM.

Based on the CPM model the various PRRs of the scenario, can be calculated, shown in figure 5 for the line scenario.

Figure 6 lists all possible node and hop count combinations for a 4 node Line-CPM scenario. The combinations can be created using integer partitioning algorithms, e.g. as in [1]. Creating all possible node and hopcount combinations of a 4 node line scenario, involves splitting up 3 into the list [(3), (2,1), (1,2), (1,1,1)], splitting up 2 into [(2), (1,1)] and 1 into [(1)]. To get to node 3 from node 0 in the line scenario is thus possible, either by going 1 hop of 3 times the base distance, or going 2 hops (with a 2 base distance hop and a 1 base distance hop in two variations), or going 3 hops of the base distance. The multiple base distances of course have an influence on the PRR according to the CPM propagation model.

If the transmission to node 3 did not succeed with a 1 hop transmission, due to the propagation model, then the transmission might succeed via intermediate nodes which have received the broadcasted transmission and transmit the same information based on their Trickle timer. The transmission updates node 3 only

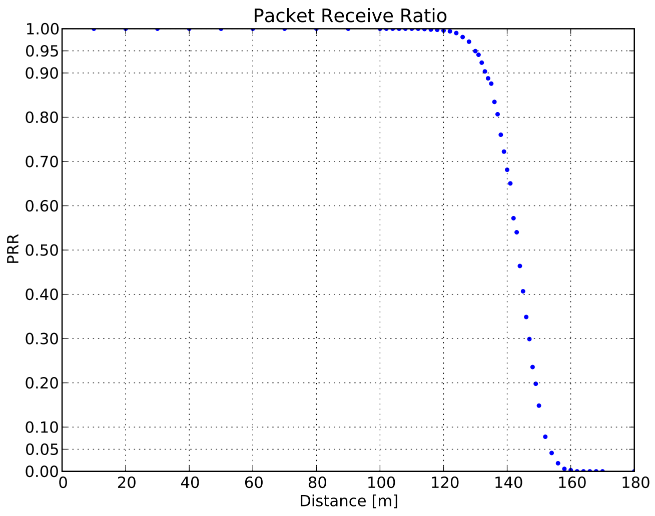

Fig. 4. Packet Receive Ratio Model in TOSSIM

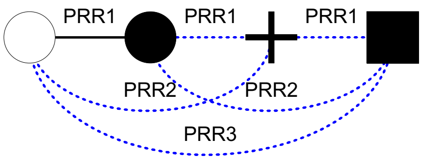

Fig. 5. Packet Receive Ratios for the Line-CPM scenario

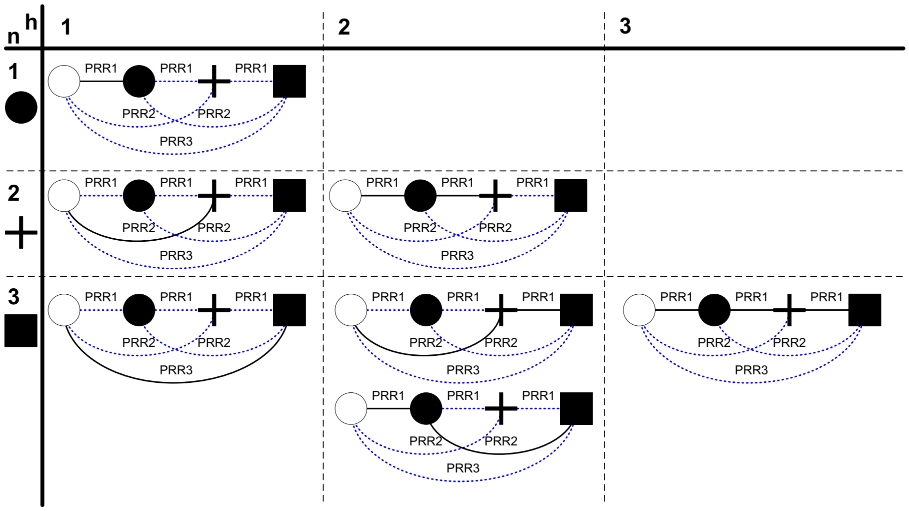

Fig. 6. Possible Node and Hopcount Combinations

with a certain conditional probability that it did not receive the information earlier directly. A conditional probability tree, with which the various conditional probabilities can be calculated is shown in figure 7. The trees are shown for all 3 nodes of the 4 node Line-CPM scenario, that have to receive the information from node 0. The nodes are signified with the respective shapes as used in figure 5 previously.

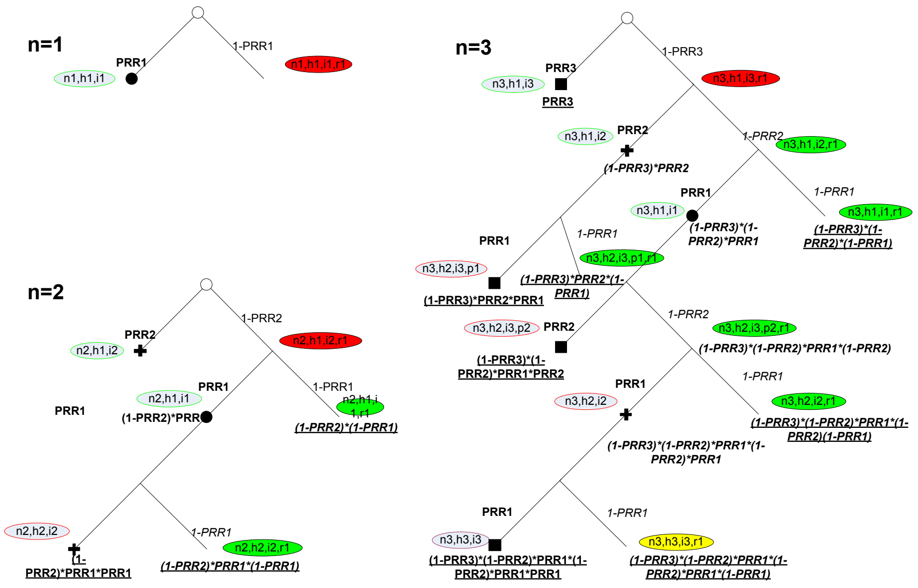

Fig. 7. Conditional Probability Tree derivation from PRR

The probability of 'retransmissions' (actually transmissions from the next Trickle cycle with a different probability density function) can be derived from the PRR tree as well.

Currently, the analytical model is fitted for the Trickle parameter $K = 3$. The authors are working on extending the model to include the parameter, so that different Trickle aggressiveness settings can be judged. Lower K values will reduce the probability of sending out the information, while higher values will incrase the probability of each node sending.

5 Comparison of Analytical Model and Simulation Results

The results of the numerically solved analytical model and the simulation results of the tool described in section 3 are shown in figure 8 for the Line-CPM scenario with 4 nodes. A very good match can be seen between the analytical and simulated results.

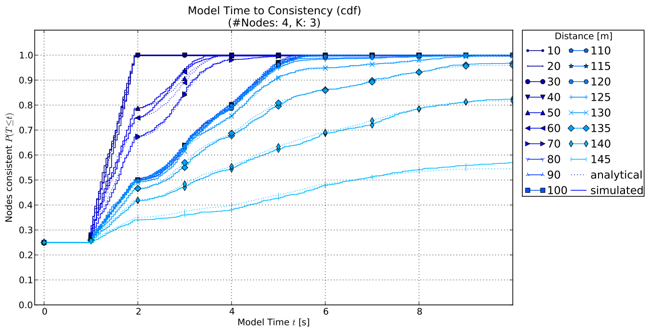

Fig. 8. Comparison of Analytical Model and Simulation Results of the Consistency Delay CDF (Line-CPM scenario, 4 nodes)

The Trickle settings for the results are:

- $\tau_L = 2$ s
- $\tau_H = 32$ s
- $K = 3$

The results are also adhering to the expectation, that for short distances the delays are lower, due to more nodes being in one-hop distance. For inter-node distances between 80 m and 120 m, the nodes are in 1, 2 and 3 hop distance with no difference in PRR due to the propagation model, thus showing the typical expected behaviour of summed convoluted unit step functions. For higher inter-node distances than 120 m the PRR of the CPM propagation model is not 1 anymore and thus reduced success probabilities and retransmissions from the next Trickle interval are governing the distribution.

In figure 9 the consistency delay CDF is shown for a Line-CPM scenario with 9 nodes. Again a good match between the simulated results and the analytical model can be seen. Slight differences are caused by using uniform pdf function,

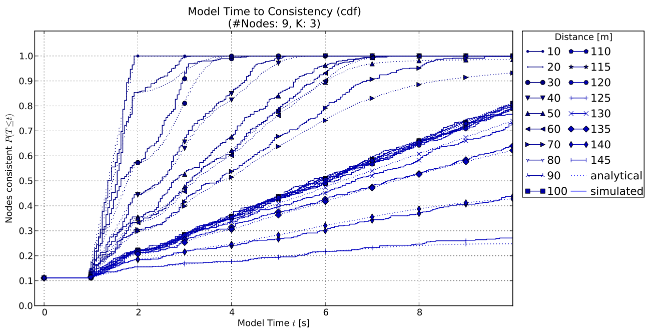

Fig. 9. Comparison of Analytical Model and Simulation Results of the Consistency Delay CDF (Line-CPM scenario, 9 nodes)

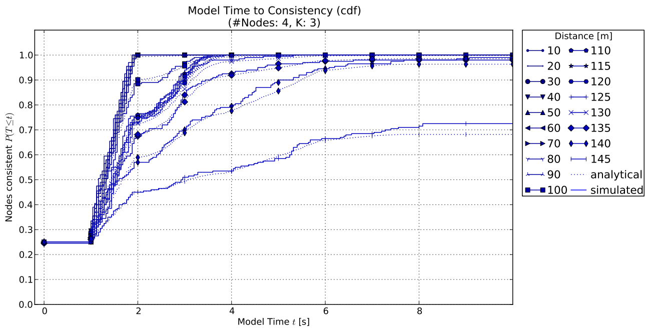

Fig. 10. Comparison of Analytical Model and Simulation Results of the Consistency Delay CDF (Grid-CPM scenario, 4 nodes)

while actually a pdf based on the minimum of several uniformly distributed random variables should be used.

As an example for the results of a Grid-CPM scenario figure 10 shows again the consistency delay CDF for 4 nodes. Also for this scenario, a good match between simulated and analytical results has been achieved.

6 Conclusions

A Wireless Sensor Network which employs the Trickle algorithm at the application layer based on blip-1.0 has been implemented. In order to be able to simulate the application, the 6LoWPAN implementation blip-1.0 has been instrumented so that it can be simulated and external tools can inject traffic into the simulated 6LoWPAN network.

Analytical models for the behaviour of the Trickle algorithms – with regard to the delay of the network consistency as well as the number of packets sent – have been derived and shown to fit the results that were obtained from the simulation for several scenarios. To the knowledge of the authors, there is no published analytical model on the Trickle algorithm, although this particular algorithm is employed for the IETF RPL routing protocol for WSNs.

Using the analytical model and the simulation tool developed, and the derived CDFs design decisions and tradeoffs can be made, e.g. for the settings of the Trickle parameters (e.g. for the routing information propagation of RPL), the number of nodes supported and the network size.

7 Outlook

Based on the analytical model further studies with regard to the parameters of the Trickle algorithm are planned, so that optimal parameters for varying use cases of the algorithm can be derived easily.

Employing the Trickle algorithm, e.g. for Constrained Application Protocol (CoAP [5]) multicast collision avoidance and service distribution for self-organizing networks of smart objects in the application area of logistics are other fields that the authors are studying.

Acknowledgments. This research project ('The Intelligent Container') is supported by the Federal Ministry of Education and Research, Germany, under reference number 01IA10001.

References

1. Eppstein, D.: Generator for integer partitions (Python recipe) (2011), `http://code.activestate.com/recipes/218332-generator-for-integer-partitions/`
2. Lee, H., Cerpa, A., Levis, P.: Improving wireless simulation through noise modeling. In: IPSN 2007: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, pp. 21–30. ACM Press (2007)
3. Levis, P., Clausen, T., Hui, J., Gnawali, O., Ko, J.: The Trickle Algorithm. RFC 6206 (Proposed Standard) (March 2011), `http://www.ietf.org/rfc/rfc6206.txt`
4. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks Reasoning About Naming Systems. In: NSDI 2004 Proceedings (2004)
5. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained Application Protocol (CoAP). Internet-Draft (work in progress) (January 2012), `http://tools.ietf.org/html/draft-ietf-core-coap-07`
6. Winter, T., Thubert, P., Brandt, A., Clausen, T., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.: RPL: IPv6 Routing Protocol for Low power and Lossy Networks. Internet-Draft (work in progress), draft-ietf-roll-rpl-19.txt (March 2011), `http://tools.ietf.org/html/draft-ietf-roll-rpl-19`