Performance analysis of Trickle as a flooding mechanism

Thomas Clausen, Axel Colin de Verdiere, Jiazi Yi
Laboratoire d'Informatique (LIX) – Ecole Polytechnique, France
Thomas@ThomasClausen.org, Axel@Axelcdv.com, Jiazi@JiaziYi.com

***Abstract*—"The Trickle Algorithm" is conceived as an adaptive mechanism for allowing efficient and reliable information sharing among nodes, communicating across a lossy and shared medium. Its basic principle is, for each node, to monitor transmissions from its neighbours, compare what it receives with its current state, and schedule future transmissions accordingly: if an inconsistency of information is detected, or if few or no neighbours have transmitted consistent information "recently", the next transmission is scheduled "soon" – and, in case consistent information from a sufficient number of neighbours is received, the next transmission is scheduled to be "later".**

Developed originally as a means of distributing firmware updates among sensor devices, this algorithm has found use also for distribution of routing information in the routing protocol RPL, standardised within the IETF for maintaining a routing topology for low-power and lossy networks (LLNs). Its use is also proposed in a protocol for multicast in LLNs, denoted "Multicast Forwarding Using Trickle". This paper studies the performance of the Trickle algorithm, as it is used in that multicast protocol.

I. Introduction

Low power and Lossy Networks (LLNs) are typically constituted by a very large number of very small and very cheap devices – each generally equipped with at most a few kilobytes of RAM and around 100kB of ROM – communicating through a very lossy, low-capacity wireless or wired medium (*e.g.,* Power Line Communication - PLC, or low-power wireless channels, like 802.15.4 PHY). Applications of these networks include utility metering, earthquake monitoring, vehicular networks, factory and home automation, etc.

The main constraints on such networks are channel utilisation, energy consumption, and storage and processing capabilities of each device. Furthermore, the lossy media and minimal routers typically used by LLNs often lead to dynamic network topologies. Protocols for these networks thus have to be specifically designed to use as little energy as possible, minimise channel utilisation and storage – while being as simple (both in terms of algorithmic and implementation complexity) as possible and be able to adapt to a changing network topology. As an example, the unicast routing protocol, LOADng [1], standardised as part of [2], uses a greatly simplified reactive route discovery mechanism to discover and maintain routes only on-demand, emphasising state reduction and implementation simplicity and eliminating rarely beneficial options [3].

LOADng necessitates dissemination of control messages throughout the entire network, often found to be an an expensive and unreliable task in LLNs when implemented by way of flooding. More generally, a multitude of protocols and applications require the ability to send a message to all devices in a network, *e.g.,* to bring all devices into a common state ("turn off all the light in this room"). In a constrained and dynamic environment, maintaining a multicast tree or multicast mesh structure in such an environment is challenging – and scoped flooding remains a potentially cheaper alternative in some scenarios [4].

This underlines the importance of efficient message dissemination techniques, adapted for LLNs. Classic approaches for efficient message dissemination include mechanisms such as Multipoint Relay [5], Essential Connecting Dominating Set (E-CDS) [6], or MPR+SP [7] – all of which do require that devices in the network maintain some additional topology knowledge, the acquisition of which incurs extra control message generation and, thus, network load. Alternative, potentially more effective, dissemination algorithms have been proposed for LLNs, including use of "The Trickle Algorithm" [8] [9], an efficient dissemination mechanism first designed for code updates in wireless sensor networks. The use of this algorithm has been proposed within the IETF [10] – denoted "Trickle Multicast" – as a standard mechanism for efficient message dissemination through an LLN.

This paper evaluates the performance Trickle Multicast, and offers a comparison with classic and MPR flooding.

A. Paper Outline

The remainder of this paper is organised as follows: section II details the operations of Trickle Multicast, and section III outlines classic flooding and MPR Flooding, used as baselines for evaluating the performance of Trickle Multicast. The different mechanisms are evaluated by way of network simulations, which are described in section IV, with the results presented in section VI. This paper is concluded in section VII.

II. Trickle Multicast

Trickle [8] was originally conceived a self-adapting algorithm for reliable code propagation in wireless sensor networks. It applies a "polite gossip" policy, where each device periodically transmits a summary of its state (say, the firmware version) to its neighbours, but suppresses this transmission if it has, recently, heard enough neighbours advertising the same state. Furthermore, as the network converges to a consistent

situation, where all devices have the same state, the algorithm generates fewer transmissions.

The Trickle algorithm is defined in terms of a received transmission being "consistent" (*i.e.,* the neighbour sending the advertisement has "the same" information as the receiver) or "inconsistent" (*i.e.,* the neighbour sending the advertisement has "different' information than the receiver), with the exact definitions of "consistent" and "inconsistent" being dependent on the protocol using the Trickle algorithm. For firmware updates, for example, a version number equality may imply "consistent" whereas a version number inequality may imply "inconsistent" – or, consistency may be defined as "using a version 3.xx firmware, for any value of xx".

Trickle defines three constants, I_{min}, the minimum interval size, I_{max}, the maximum interval size, defined in terms of doublings of I_{min}, and k, the redundancy constant. Each device maintains two parameters: I, the size of the current transmission interval, t, a time within the current interval, and a counter c, the number of consistent transmission received during this interval. A device implementing Trickle proceeds as follows:

1) At initialization, the device chooses I randomly in $[I_{min}, I_{max}]$.
2) An interval start, t is chosen randomly in $[I/2, I)$ and c is reset to 0. The interval finishes a I. The device then listens to incoming transmissions until time t.
3) When the device receives a transmission from one of its neighbours, it determines if either itself or its neighbour has more recent data than the other:
   a) If the transmission represents a consistent event, *i.e.,* no device has new data for the other, c is incremented.
   b) Else, if the neighbour device has new data, this device updates its data, resets I to I_{min} and starts a new interval.
   c) Else, if this device has data unknown to the neighbour device, it directly transmits it to all of its neighbors, without resetting the interval.
4) At time t, the device transmits a summary of its data to all of its neighbours only if $c < k$. Otherwise, the device doubles the interval size. If this new interval size would be greater than the time specified by I_{max}, Trickle sets the interval size I to be the time specified by I_{max}.

Trickle can be used as an optimised flooding algorithm in LLNs, proposed in [10] and henceforth denoted "Trickle Multicast", for support of IPv6 multicast forwarding in LLNs. This use of Trickle introduces a "seed" and a sequence number for multicast messages dissemination.

"Trickle Multicast" uses a Trickle IPv6 option to carry SeedID[1] and Sequence number[2] with each multicast message, allowing them to be uniquely identified. Each device also maintains sliding windows (one per known SeedID), which ensures that each message received is processed at most once by the device: an incoming multicast message is accepted if and only if its sequence number is not stored in the corresponding (*i.e.,* identified by the received message's SeedID) sliding window, and the Sequence Number is greater than the lower bound of that sliding window. Devices then advertise their state, expressed through a Trickle ICMPv6 message summarizing the recently received multicast messages, to their neighbours through transmissions regulated by the Trickle algorithm

"Trickle Multicast" defines a received summary message to be "consistent" if the recently received multicast messages identified therein are identical to those also received by the recipient of the summary message.

[1]The SeedID is an identifier of the Trickle Multicast router by which the multicast message enters the Trickle Multicast domain; it may be different from the source IP address if, for example, the multicast packet originates from a device outside that domain.

[2]Monotonically increasing, maintained by the Trickle Multicast router identified by the SeedID.

III. Baseline Flooding Mechanisms

As baselines for comparing Trickle Multicast, this section briefly discusses Classic Flooding and MPR Flooding.

A. Classic Flooding

Classic flooding is the simplest form multicast message diffusion: when receiving a multicast message, each device will verify if it has already received a copy hereof. If yes, the message is silently dropped, if no, the message is sent to each of its neighbours. This implies that the only state a router has to maintain is a buffer, retaining identifying information for the recently flooded messages received, and that each flooded data message must contain a message identifier (typically, by way of a sequence number and the address of the source of the flooded message). No explicit control messages are generated, and the only control signalling incurred is the message identifier. On the other hand, since each flooded message is transmitted exactly once by each device in the network, it is a potentially expensive operation [11], and subject to the infamous "broadcast storm problem" [12]. Classic Flooding is considered the absolute baseline, against which any more complex flooding mechanisms must provide a measurable improvement.

B. MPR Flooding

With Multipoint Relay Flooding [13], each device selects a set of relays (its MPRs) from among its direct neighbours. A device, X, selects its MPR Set such that a message transmitted by X and relayed only by the members of its MPR set will be received by all devices 2 hops away, and stipulates that a device relays a multicast message only if received from a device having selected it as MPR. Thus, MPR Flooding reduces the number of redundant copies of messages sent through the network. MPR selection requires that each device maintains, at least, topology up to two hops away from itself, and to this end, participating devices periodically exchange HELLO messages. Furthermore, a greedy algorithm is applied to determine the relay set, which yields an approximation of an optimal connected dominating set – this set is also signalled by each device through HELLO messages.

IV. Simulation Environment

In order to evaluate the performance of Trickle Multicast, network simulations by way of NS2 are employed. While network simulations are, at best, an approximation of real-world performance (particularly due to the fidelity of lower layers vs. reality), they do provide a baseline for comparison and, generally, best-case results, *i.e.,* real-world performance is expected to be no better than that which is obtained through simulations. The reason for using network simulations is, that such allow running experiments with different protocols (in this case, Trickle Multicast, Classic Flooding and MPR Flooding) under identical conditions and parameters (MAC layer, distribution, number of nodes, etc.)

Simulations were conducted using the TwoRayGround propagation model and the IEEE 802.11 MAC. Although there are various low-layer technologies more commonly (and, perhaps, more viably) used for LLNs (power line communication, 802.15.4, low-power wifi, bluetooth low energy, etc.), general behaviour of a protocol can be inferred from simulations using 802.11.

A. Network Topology and Multicast Traffic Characteristics

The general network topology of a scenario is as follows: n devices are placed randomly (while ensuring that the network is still connected) in a square field of size $m \times m$ meters. From among the n devices, x devices are randomly chosen as concurrent multicast data sources, each generating a multicast data packet of 15 octets every 30 seconds, and with each multicast data source generating $n-1$ multicast data packets.

These data traffic characteristics are not chosen arbitrarily, but rather to reflect the traffic characteristics that one might see if using Trickle Multicast for carrying the multicast portion of the route discovery mechanism of LOADng – as alluded in the introduction as one interesting use of an optimised flooding mechanism in an LLN – where each of x wishes to send a route discovery to all other devices in the network. This, in turn, may represent a set of controllers in a network, wishing to inquire for sensor readings from, or manipulate actuators on, all of these devices.

B. Protocol Parameters

The protocol parameters used for all the simulations are shown in table I. Jitter is used on multicast messages for MPR Flooding and Classic Flooding, according to [14], so as to reduce the risk of collisions. Trickle has an implicit jitter mechanism, hence [14] is not used with Trickle Multicast.

C. Scenario Descriptions

Four different kinds of scenarios are studied, comparing the three different mechanisms. For the first scenario, different sets of Trickle parameters are considered, including those recommended by [15] for AMI (Automatic Metering Infrastructure) networks[3], and reproduced in the following:

[3][15] recommends parameters for Trickle when used for "flooding" RPL control traffic – DIOs – in an "Automatic Metering Infrastructure" network.

Parameter	Default value	Note
Trickle Multicast parameters		
I_{max}	2^{16} x I_{min}	
Window size	3	Size of the sliding windows
MPR Flooding parameters		
HELLO interval	5s	
Neighbor expiration time	25s	
MPR and Classic Flooding parameters		
Jitter	500ms	Maximum jitter value for broadcast transmissions

Table I
Default Protocol Parameters for Simulation

- I_{min} should be set to at least 50 times as long as it takes to transmit a link-local multicast packet. During the simulations, a transmission time around 1ms was observed, thus I_{min} = 100ms satisfies this constraint without being overly conservative;
- I_{max} should be greater than 2 hours.
- the redundancy constant should be set to at least 10.

These parameters, given in table II, are recommended for the use of Trickle in RPL [16], and used as a point of comparison. The scenarios studied are:

Variable Density
: The performance of Trickle flooding in a given network density depends on the parameters chosen, in particular for k, and I_{min}. These simulations are conducted using an 1000m $\times$ 1000m square, with the number of devices varying between 32 and 250. For the purpose of these simulations, a single multicast data source is used ($x = 1$), and the scenarios are tested with trickle parameters as given by table II.

Fixed Density
: This set of simulations compares the three flooding mechanisms in a network of fixed density of 50 devices/km^2. The number of nodes varies between 15 and 500, and the field dimensions varies correspondingly from 595m$\times$595m to 3162m$\times$3162m
For the purpose of these simulations, a single multicast data source is used ($x = 1$), and for Trickle Multicast, I_{min}= 1s and k=2 is used.

Variable Number of Multicast Sources
: This set of simulation compares the three flooding mechanisms when subject to a variable number of multicast sources, ranging from $x = 1$ to $x = 30$, in a network with 125 devices distributed across a field of 1581m$\times$1581m. For Trickle Multicast, I_{min} = 1s and k=2 is used.

Loss Resiliance
: The inconsistency detection and retransmission mechanism of the Trickle algorithm is supposed to ensure that message losses are eliminated – *i.e.,* that given enough time, all devices will have consistent state information. Thus, this set of simulations compares the performance of the flooding mechanisms in a network subject to (explicit and excessive) message

losses: a packet is lost with a certain (independent) probability, ranging from 0.0 to 0.7.

A single multicast data source is used ($x = 1$), in a network with 125 devices distributed across a field of 1581m×1581m. For Trickle Multicast, I_{min} = 1s and $k = 2$ is used.

Parameter	Values
I_{min}	200ms, 500ms, 1s, 1.5s
k	2, 6
AMI parameters (from [15])	
I_{min}	100ms
k	10

Table II
TRICKLE PARAMETERS FOR VARIABLE DENSITY SIMULATIONS

V. PERFORMANCE METRICS

The metrics used for evaluating the performance Trickle Multicast, and compare with that of MPR Flooding and Classic Flooding are as follows:

Data delivery ratio
: This metric measures the success rate of the different flooding mechanism. For a given simulation, it averages the per-source Data delivery ratio, which itself is the average over all the multicast data packets sent by this source of the number of devices which have received that packet, divided by the number of devices other than the source.

Number of multicast data packet transmissions
: This metric measures the number of times a given data packet has to be retransmitted in order for it to reach all the devices in the network. For a given simulation, it is computed as the total number of multicast data packets sent.

Total number of transmissions
: This metric counts the total number of packets transmitted during the simulation, counted as the source. In other words, each control or data packet sent adds one to the result.

Network load
: The Network load measures the overall load the flooding mechanism has put on the network during the simulation, i.e. the actual number of KB sent, both control and data packets.

Data Delivery Delay
: The Data delivery delay represents the time taken, on average, for a multicast data packet to reach all the devices in the network (or, in case all the devices are not reached, all the devices that will ever receive this packet). As such, it is computed as the average over all the multicast data packet generation of the time between the first transmission of the packet and the last time a device receives that packet for the first time (a device receiving a given data packet twice does not increase the delivery delay).

Flooding Path Length
: This metrics gives a measure of the quality (measured by the hop count) of the paths taken by the multicast data packets from their source to each device in the network. Hence it is the average, over all the data packets originated during the simulation, of the average hop count observed at every receiving device (each packet is counted only once at each device). Although hop count is generally considered a poor metric for routing, it still gives an indication regarding the efficiency of the flooding mechanism.

VI. SIMULATION RESULTS

The evaluation of Trickle Multicast, and the comparison with MPR Flooding and Classic Flooding in each of the four scenarios described in section IV, and with respect to the performance metrics presented in section V is presented and discussed below.

A. Variable density

Figures 1, 2, 3 and 4 depict the results for the Variable Density scenarios – it is important to recall that only a single multicast data source is present in these simulations.

For all three mechanisms, identical data delivery ratios (approximately of 100%) were attained. Figure 1 shows that Classic Flooding, as expected, presents the largest number of messages sent and figure 2 that – depending on the parameters chosen – Trickle Multicast can offer the fewest number of data message transmissions.

Noting that in terms of media occupation and energy consumed for transmitting and receiving it, a more appropriate metric is the "load" of the network, which is the number of octets (control traffic and data traffic) necessary to complete the flooding operations. This is depicted on figure 3, where it can be seen that the "gain" as compared with classic flooding, in terms of fewer messages sent by Trickle Multicast, is greatly diminished by the size of the Trickle Multicast control messages. Note that this graph does not include the overhead for MPR Flooding which, due to the periodic signals for acquiring local topology, is way higher than that of both Trickle Multicast and Classic Flooding.

It is worth noting the sensitivity of the choice of Trickle parameters. First, as expected, the value of I_{min} directly affects the data delivery delay, as depicted in figure 4 – whereas the value of k (the redundancy constant) has very little effect on the incurred delay. However, increasing k and decreasing I_{min} negatively affects the overhead incurred. The effect of k is very intuitive, since it represents the number of transmissions a given "area" will do during each transmission interval. Figure 4 also shows that Trickle Multicast consistently incurs higher delivery delay, even when choosing the best possible parameters for the scenario, from among those tested. In particular, the value of I_{min} directly affects the delay - thus forcing a tradeoff between overhead and delay when choosing this parameter. In summary, the performance of Trickle Multicast depends on the choice of parameters, and

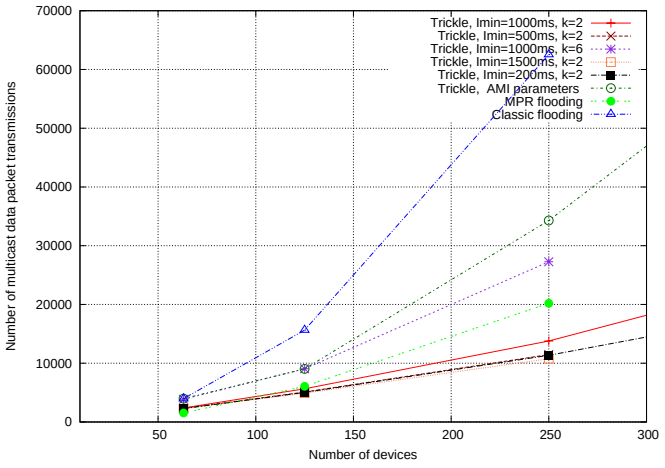

Figure 1. Variable Density: Number of multicast data packet transmissions

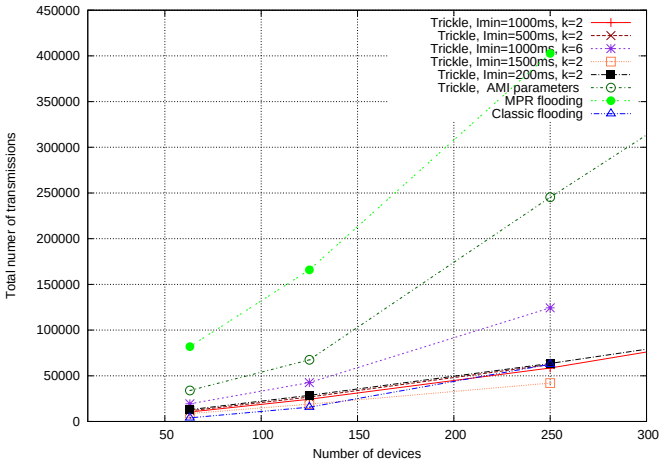

Figure 2. Variable Density: Total number of transmissions

the set of parameters recommended by [15] exhibits the worst performance from among those tested.

B. Fixed density

Figures 5, 6, 7, and 8 depict the results for Fixed Density scenarios. For all three mechanisms, identical data delivery ratios (approximately of 100%) were attained.

These simulations suggest that the performance of Trickle flooding is tightly related to the network size, noting that even with the "best" (according to the previous set of simulations) parameters chosen for Trickle, the overall network load remains higher than with classic flooding. This, due to the overhead incurred by the Trickle control messages, which is proportional to the number of devices in the network.

Figure 8 depicts the average path lengths that data packets follow from the source to all destinations. As all mechanisms attain identical data delivery rates close to 100%, path lengths can be compared directly, concluding that "shorter is better". MPR Flooding yields shorter path lengths than both Trickle Multicast Classic Flooding, confirming the observations of [17], with Trickle Multicast attaining systematically

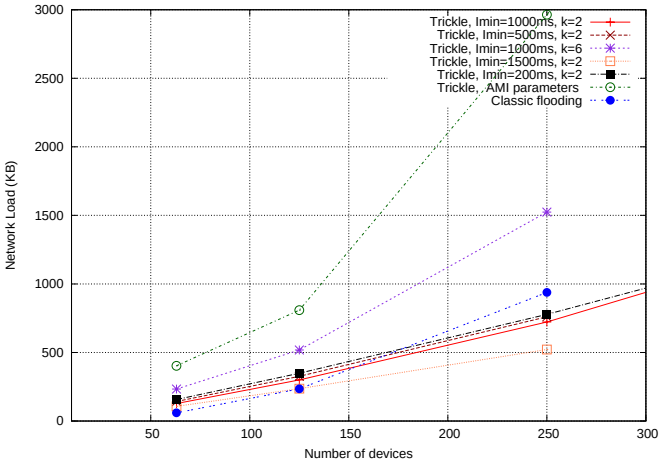

Figure 3. Variable Density: Network Load (control + data, in KB).

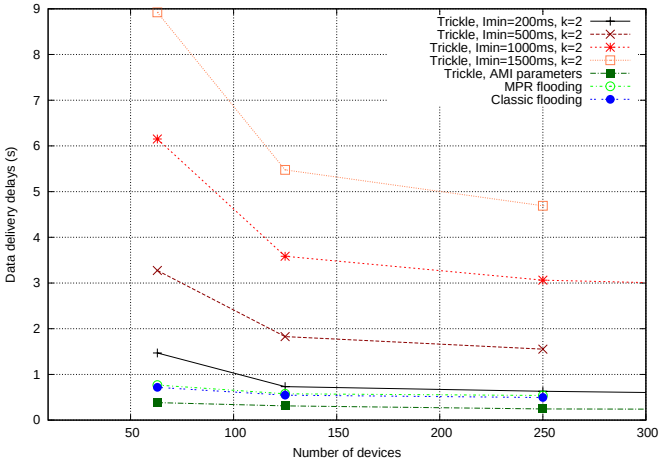

Figure 4. Variable Density: Data Delivery Delays.

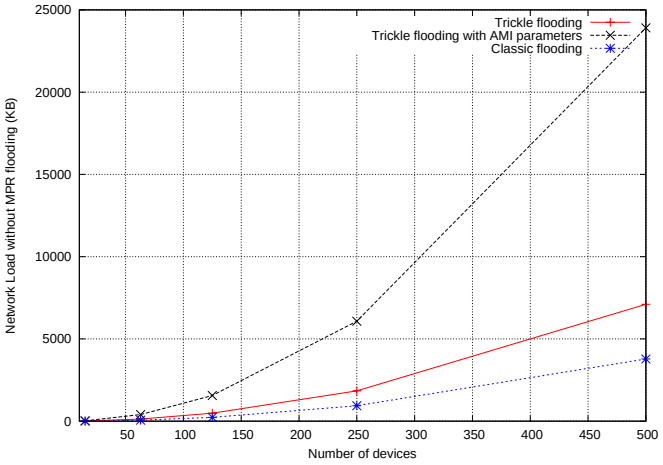

Figure 5. Fixed Density: Network Load (control + data, in KB).

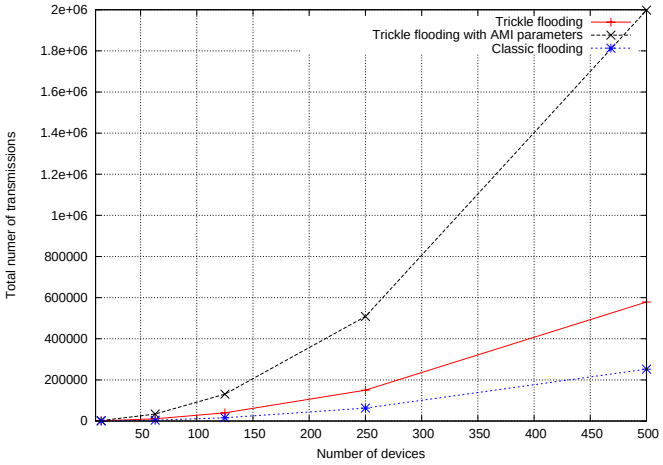

Figure 6. Fixed Density: Number of transmissions (control+data).

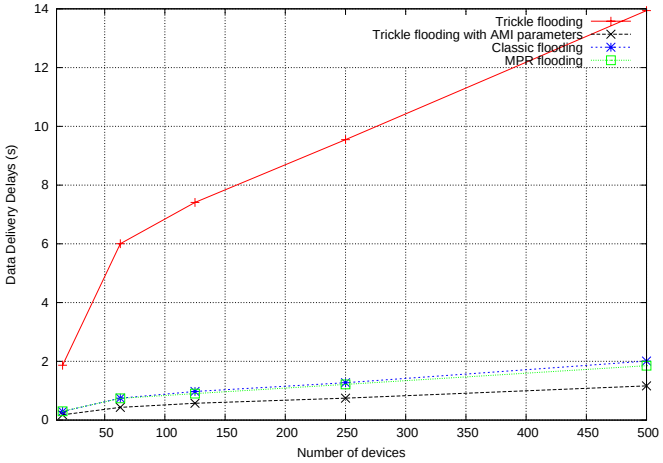

Figure 7. Fixed Density: Data Delivery Delays.

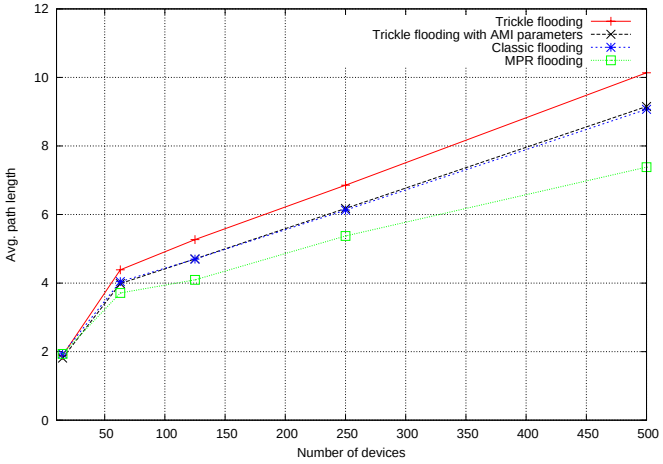

Figure 8. Fixed Density: Flooding Path Lengths.

longer paths than both Classic Flooding and MPR Flooding. The difference between the average path lengths from MPR Flooding to Trickle Multicast remains moderate, however increases as the network grows larger (up to 37%). This can be an issue for some applications: in LOADng for example, this would increase the length of the discovered paths, thus adversely affecting routing protocol performance.

C. Variable Number of Multicast Sources

Figures 9 and 10, depict the results for Variable Number of Multicast Sources. For all three mechanisms, identical data delivery ratios (approximately of 100%) were attained.

Figure 9 shows that the overhead incurred by Trickle increases dramatically as the number of concurrent Multicast Sources increases. Whereas for the previous scenarios featuring only a single Mutlicast Source, the network load (control and data) incurred by MPR Flooding was dramatically larger than that of Classic Flooding and Trickle Multicast, as soon as there are multiple concurrent Multicast Sources in the network, Classic Flooding incurs a lower network load than Trickle Multicast – and from 6 concurrent Multicast Sources, Trickle Flooding incurs a higher network load than MPR Flooding and Classic Flooding, both.

This sharp increase in network load, incurred by Trickle when faced with multiple Multicast Sources, can be explained by the requirement that all devices must generate the "summary packets" (section II), each of which grows as a function of the number of Multicast Sources as each summary packet carries information about all the current active sources, with their associated sliding windows. Thus, Trickle Multicast is negatively affected by the number of multicast sources.

Figure 10 depicts the data delivery delay. While the delays remain constant and independent of number of Multicast Sources for MPR Flooding and Classic Flooding, Trickle Multicast experiences a decrease in delay as the number of Multicast Sources grow – an 21% decrease between a network with 1 Multicast Source and 30 Multicast Sources. The explanation for this behaviour is, that the Trickle timers are reset more often (due to inconsistencies detected more frequently), causing more frequent (re)transmissions. Even so, the data delivery delays incurred by Trickle Multicast remain, consistently, substantially larger than those incurred by Classic Flooding and MPR Flooding.

D. Loss Resilience

Figures 11 and 12 depict the Loss Resilience of the three mechanisms. While the model for packet loss used was very simple, it does provide some insight into the behaviour of Trickle Multicast in lossy environments. While the data delivery delays for Trickle Multicast grow dramatically (figure 11) as the loss rate increases, its data delivery rate (figure 12) stays consistently high – whereas it drops off for Classic Flooding and MPR Flooding. From among the three mechanisms, MPR Flooding is the most vulnerable to lossy links – this, simply, as MPR Flooding optimises flooding by eliminating redundant copies of multicast data packets, present in the network.

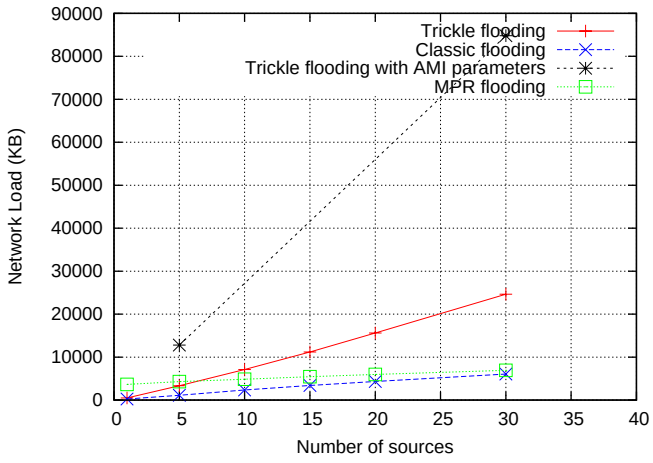

Figure 9. Variable Number of Multicast Sources: Network Load (control + data).

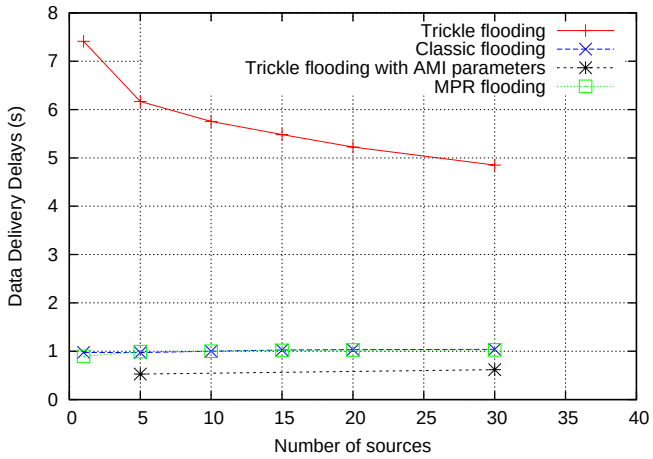

Figure 10. Variable Number of Multicast Sources: Data Delivery Delays.

The gain in loss resilience of Trickle Multicast comes at a cost of increased delay (of up to 1200% compared to classic flooding in the simulations performed).

VII. Conclusion

This paper has evaluated the performance of Trickle Multicast, comparing with that of Classic Flooding and MPR Flooding. Trickle Multicast was found to be resilient to very lossy links, and to maintain an overall high data delivery ratio when faced with such – whereas Classic Flooding and MPR Flooding, both, fail to deliver acceptable delivery ratios in these scenarios. On the other hand, in networks with more reliable links, Trickle Multicast exhibit less excellence.

The simulations have shown that the performance of Trickle Multicast is highly sensitive to the choice of parameters: the simulations showed that the same set of parameters can render Trickle Multicast the best or worst performer in a given scenario – and, in some cases, an inadequate choice of parameters makes Trickle Multicast a far worse solution that Classic Flooding, both in terms of overhead and delay,

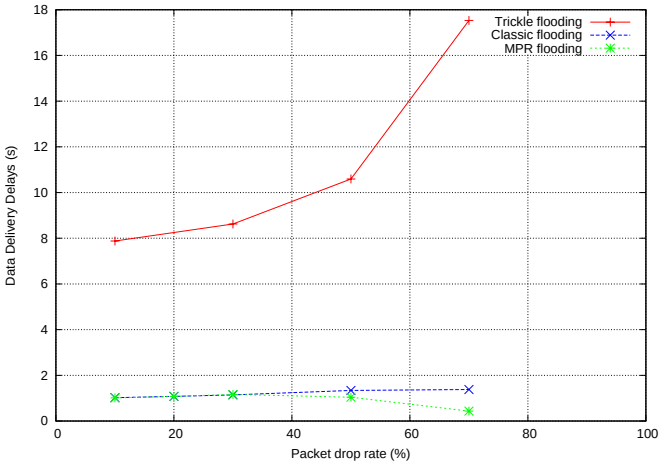

Figure 11. Loss Resiliance: Data Delivery Delays.

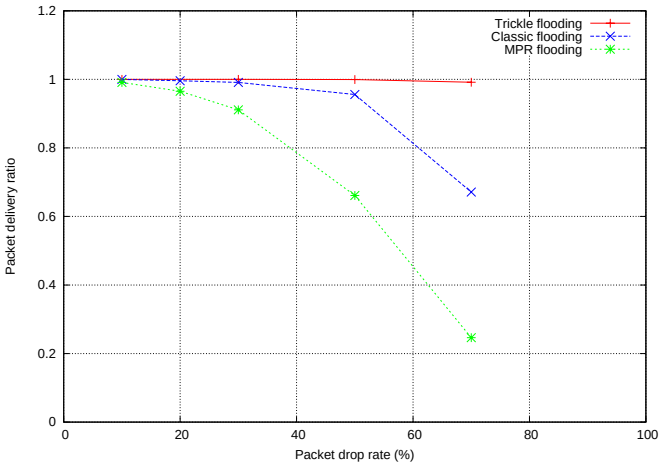

Figure 12. Loss Resilience: Data Delivery Ratio.

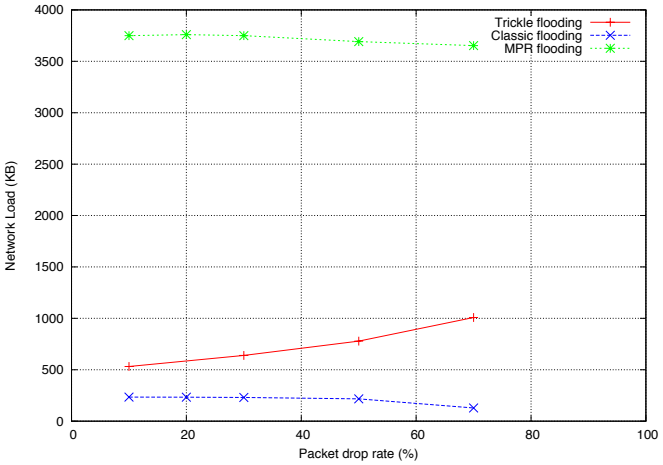

Figure 13. Loss Resilience: Network Load.

while not achieving any better multicast data delivery ratio. This poses two challenges in the use of this protocol for real life network scenarios: first, more in-depths experimental studies are be needed for each particular use-cases, in order to understand how to choose the appropriate parameters. Second, the sensitivity exhibited by Trickle Multicast makes it vulnerable to changes in network conditions over the duration of the network lifetime – which, for sensor networks, is an expected operational condition.

An additional observation for Trickle Multicast is, that its performance (network load and delays incurred) depends on the number of concurrent Multicast Sources active in the network: while with a single Multicast Source, Trickle Multicast may exhibit a reasonably low overhead, the size of the control signals that Trickle Multicast employs is proportional to the number of Multicast Sources. Thus, the network load incurred by Trickle Multicast rapidly exceeds that incurred when using MPR Flooding and Classic Flooding, when there are more than one Multicast Source.

Thus, to make Trickle Multicast generally applicable for autonomous sensor networks, it is necessary to further investigate how to (i) detect changes in network conditions and (ii) automate the adaptation and installation of such configuration parameters. Furthermore, a deployment should carefully consider the number of Multicast Sources expected in the network (at installation and in the future), in order to understand if the network load incurred by Trickle Multicast is commensurate with the network capacity, and the delays incurred on data delivery are acceptable to application. Finally, any use of Trickle Multicast should consider the implications of the Trickle Multicast trade-offs: as indicated, *e.g.,* if using Trickle Multicast for flooding routing protocol control signals, the longer paths that these control signals will take may cause a routing protocol to discover longer routes and thus ultimately degrade unicast performance.

REFERENCES

[1] T. Clausen, A. C. de Verdiere, J. Yi, A. Niktash, Y. Igarashi, H. Satoh, and U. Herberg, "The lln on-demand ad hoc distance-vector routing protocol - next generation," The Internet Engineering Task Force, January 2013, internet Draft, work in progress, draft-clausen-lln-loadng-08.

[2] "ITU-T G.9903: Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks: Amendment 1," May 2013.

[3] T. Clausen, J. Yi, and A. C. de Verdiere, "Loadng: Towards aodv version 2," in *Proceedings of the 2012 IEEE 76th Vehicular Technology Conference: VTC2012-Fall*, September 2012.

[4] J. Macker, "Simplified Multicast Forwarding," RFC 6621 (Experimental), Internet Engineering Task Force, May 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6621.txt

[5] C. Adjih and L. Viennot, "Computing connected dominated sets with multipoint relays," Journal of Ad Hoc and Sensor Wireless Networks, Tech. Rep., 2002.

[6] R. Ogier and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding," RFC 5614 (Experimental), Internet Engineering Task Force, Aug. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5614.txt

[7] J. A. Cordero, T. Clausen, and E. Baccelli, "Mpr+sp: Towards a unified mpr-based manet extension for ospf." Hawaii International Conference on System Sciences, January 2011.

[8] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI*, 2004, pp. 15–28.

[9] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206 (Proposed Standard), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6206.txt

[10] J. Hui and R. Kelsey, "Multicast forwarding using trickle," Internet Draft, work in progress, draft-hui-6man-trickle-mcast, January 2011.

[11] T. H. Clausen, L. Viennot, T. Olesen, and N. Larsen, "Investigating data broadcast performance in mobile ad-hoc networks," in *In Proceeding of Wireless Personal Multimedia Communications. MindPass Center for Distributed Systems, Aalborg University and project Hipercom, INRIA Rocquencourt, Fifth International Symposium on Wireless Personal Multimedia Communications*, 2002.

[12] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ser. MobiCom '99. New York, NY, USA: ACM, 1999, pp. 151–162. [Online]. Available: http://doi.acm.org/10.1145/313451.313525

[13] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," INRIA Research Report No. RR-3898, 2000.

[14] T. Clausen, C. Dearlove, and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)," RFC 5148 (Informational), Internet Engineering Task Force, Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5148.txt

[15] D. Popa, J. Jetcheva, N. Dejean, R. Salazar, J. Hui, and K. Monden, "Applicability statement for the routing protocol for low power and lossy networks (rpl) in ami networks," Internet Draft, work in progress, draft-hui-6man-trickle-mcast, May 2012.

[16] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[17] T. H. Clausen, P. Jacquet, and L. Viennot, "Optimizing route length in reactive protocols for ad hoc networks," in *In Proceeding of The First Annual Mediterranean Ad Hoc Networking Workshop*, 2002.