Stateless Multicast Forwarding with RPL in 6LowPAN Sensor Networks

George Oikonomou, Iain Phillips
Computer Science, Loughborough University, Loughborough, LE11 3TU, UK
Email: {G.Oikonomou, I.W.Phillips}@lboro.ac.uk

***Abstract*—Recent research efforts have resulted in efficient support for IPv6 in Low power Wireless Personal Area Networks (6LoWPAN), with the "IPv6 Routing Protocol for Low power and Lossy Networks" (RPL) being on the forefront as the state of the art routing approach. However, little attention has been paid to IPv6 multicast for networks of constrained devices. The "Multicast Forwarding Using Trickle" (Trickle Multicast) internet draft is one of the most noteworthy efforts, while RPL's specification also attempts to address the area but leaves many questions unanswered. In this paper we expose our concerns about the Trickle Multicast (TM) algorithm, backed up by thorough performance evaluation. We also introduce SMRF, an alternative multicast forwarding mechanism for RPL networks, which addresses TM's drawbacks. Simulation results demonstrate that SMRF achieves significant delay and energy efficiency improvements at the cost of a small increase in packet loss. We have extended the TCP/IP engine of the Contiki embedded Operating System to support both algorithms. Both implementations have been made available to the community.**

***Keywords*-6LoWPAN; Wireless Sensor Networks; Multicast**

I. Introduction

Over the past decade, research community members have invested considerable efforts towards the seamless integration of wireless sensor networks (WSNs) with the internet. Previous work has demonstrated that pure IPv6-based WSN architectures are not only viable but can also outperform application-centric designs [1]. Significant standardisation efforts have contributed to mature, interoperable implementations of embedded IPv6 stacks, such as uIPv6 which is distributed as part of the Contiki embedded Operating System. Among those standards are RFC 4944 [2] and RFC 6282 [3]. Published by IETF's 6LoWPAN work group, they discuss techniques for IPv6 datagram fragmentation and header compression, achieving their efficient transmission within IEEE 802.15.4 low power radio frames. For those networks (6LoWPANs), the emerging standard for routing is the "IPv6 Routing Protocol for Low power and Lossy Networks" (RPL) [4].

Despite all recent advances in the area of 6LoWPAN networking, IPv6 multicast has been somewhat overlooked by the community. The "Multicast Forwarding Using Trickle" internet draft (or Trickle Multicast - TM) [5] poses among the most suitable candidates. In this context, this paper's contributions are the following:

- We scrutinise TM from a design perspective and we present our concerns, backed up by simulation results. We demonstrate that its performance is very sensitive to configuration changes and we recommend optimal parameters for various scenarios.
- We present SMRF, our lightweight, RPL-specific "Stateless Multicast RPL Forwarding" algorithm. We discuss how it addresses open issues and demonstrate that it out-performs TM in terms of delay and energy consumption.
- We have extended Contiki's TCP/IP stack to support both algorithms. The source code has been released to the community for adoption and further scrutiny as part of our port of the Contiki OS [6] and is soon to be merged with the official Contiki distribution.

II. Related Work

Clausen and Herberg conducted relevant research with focus on RPL networks [7]. Despite the fact that this work only discusses broadcast communication, it offers some important insight on related issues and techniques. Koutsonikolas et al. investigate multicast in WSNs, but without discussing 6LoWPAN-specific problems [8].

In RPL networks, nodes advertise unicast downward paths inside Destination Advertisement Object (DAO) messages. A section in the RPL internet draft discusses its "*Storing with multicast support*" Mode of Operation (MOP). In this MOP, DAO messages are also used to relay multicast group registrations. Those DAOs are identical to the ones carrying unicast information except for the type of prefix being advertised (a routable multicast IPv6 address). This approach leaves many open issues, as we discuss in sec. II-A.

A. Multicast Forwarding with Trickle

Trickle [9], [10] is an algorithm that governs the frequency of periodic information exchange among neighbouring nodes in a low power, lossy network. It provides a method of propagating information efficiently, without constantly flooding the network with control messages. The algorithm only dictates the behaviour of timers, in other words *when* nodes should exchange messages, not *how* nor their format. In simple terms: when two single-hop neighbours share the same knowledge (*agree*), control message exchange rate slows down exponentially, achieving energy and bandwidth efficiency. Conversely, when a change is detected, the trickle timer is reset to a minimum interval (called $Imin$) and changes propagate within milliseconds.

The algorithm's properties make it very attractive for any protocol involving "*periodic*" exchange of state information. Trickle was originally designed for data dissemination and network reprogramming but has since then been adopted by multiple works. For example, it handles the frequency of RPL DIO (upward route advertisement) messages [4]. It also forms the basis of the "Multicast Forwarding with Trickle" algorithm, further discussed in the remainder of this session.

Multicast mechanisms for wired networks maintain network topology information in order to forward packets to their intended destinations. Due to memory restrictions, this is a very challenging task in networks of constrained nodes. TM specifies a method of supporting IPv6 multicast without having to rely on topology maintenance [5].

With TM, each multicast datagram must carry a *Multicast Option* header (in the shape of an IPv6 Hop-by-Hop Option extension header - HBHO). Network nodes maintain a cache of recently seen multicast packets, uniquely identified with the assistance of the HBHO. Upon reception of a multicast datagram a node inspects the multicast option and, if the packet is new, it gets added to the cache.

Neighbouring nodes exchange information about their cache contents through ICMPv6 datagrams, at a frequency controlled by trickle timers. If the receiving node's cache contents don't match the information in the ICMPv6 datagram, the node resets its trickle timer to its minimum interval ($Imin$) in order to facilitate quick propagation of new packets. Inconsistency is also triggered upon reception of a new multicast datagram. At every trickle interval, nodes forward inconsistent datagrams to their single-hop neighbours inside link-layer broadcast frames.

1) Advantages: By design, TM has two very significant advantages. *Generality*: TM will work, without modifications, alongside any IPv6 routing protocol. *Reliability*: By caching and maintaining per-datagram state information, TM increases its reliability (high packet delivery ratio / low loss). The exact reliability levels are heavily influenced by the choice of $Imin$ and the underlying duty cycling algorithm.

2) Concerns: i) *Scalability*: TM replaces topology maintenance with per-packet state. This raises concerns regarding scalability with traffic volume and number of traffic sources.

ii) *Performance*: In order to avoid duplication, nodes never forward multicast datagrams immediately. Instead, they cache them and wait for ICMPv6 control messages. When an inconsistency is detected, the packets causing it are scheduled for transmission during the next trickle interval. This forwarding delay has an impact on end-to-end delay and can be heavily influenced by trickle parametrization. The trickle RFC [10] dictates that "*A protocol specification that uses Trickle MUST specify: Default values for* Imin*, Imax, and k...*". Currently, this is not the case for TM; its internet draft only outlines examples with an indicative value (100ms) [5]. As we demonstrate in section IV, this value can lead to very poor performance. Experimental results guide us to more suitable alternatives.

iii) *Complexity*: Nodes maintain two trickle timers, a sliding window for each source of multicast traffic and a cache of recent multicast datagrams. They also need to be able to create and process a new type of ICMPv6 message and a new type of HBHO extension header. Especially in the case of incoming ICMPv6 messages, a node needs to compare all entries in the message against all cached messages. This raises concerns in terms of code size and memory footprint.

iv) *Multicast vs Broadcast*: Due to lack of topology maintenance and group registrations, TM will forward multicast messages to all parts of the network, regardless whether they are needed or not. Any datagram with a routable multicast IPv6 destination address is in practice treated as a *network-wide broadcast*, causing energy and bandwidth inefficiencies.

v) *Arrival Order*: Due to its store and forward nature and per-packet state maintenance, TM is very susceptible to out-of-order datagram arrivals. Depending on the requirements of a multicast application, this trait can be a serious problem.

III. Stateless Multicast RPL Forwarding - SMRF

The principal rationale behind SMRF is that nodes participating in an RPL network exchange topology information in order to build a Destination-Oriented Directed Acyclic Graph (DODAG) (the basic RPL construct) and populate their routing tables. Since each node will maintain a state of the network topology anyway, we can capitalise on it in order to perform multicast forwarding without having to define and implement further control messages. Additionally, by being a tree structure, a DODAG is a particularly attractive candidate to form the basis of multicast forwarding.

When a network uses the "*Storing with multicast support*" Mode of Operation (MOP), nodes join a multicast group by advertising its address in their outgoing DAO messages, which only travel upwards in the DODAG. Upon reception of such message from one of its children, a router makes an entry in its forwarding table for this multicast address. Conceptually this entry indicates that "*a node under us in the DODAG is a member of this group*". This router will then i) advertise this address in its own DAOs and ii) relay multicast datagrams destined to this address.

This RPL built-in mechanism addresses the problem of propagating group membership information towards the DODAG root. However, it suffers from two severe drawbacks: i) It lacks a method to prevent a node from accepting the same datagram twice or more. ii) The RPL internet draft specifies that each router should copy multicast datagrams to a subset of its link layer neighbours, for instance only its preferred parent or only those children that are registered group members. This destination filtering can only be achieved by using frames with a unicast destination at the link layer. Thus, a node would have to transmit each datagram multiple times, once per intended recipient. This

Table I
EXAMPLES OF SMRF CONFIGURATION PARAMETERS AND RESULTING FORWARDING DELAY

Duty Cycling		*Configuration*		*Outcome*	
Algorithm	CCI (ms)	$Fmin$ (ms)	$Spread$	D (ms)	Final Delay (ms)
ContikiMAC	125	$\leq$ 125	1	125	125
ContikiMAC	125	$\leq$ 125	4	125	[125 , 500]
NullRDC	0	0	ignored	0	Immediate
NullRDC	0	31.25	8	31.25	[31.25 , 250]

would incur additional costs in terms of traffic, delay and processing time and would be largely inefficient in dense networks. It would also increase memory requirements, since each router would need to maintain associations between multicast groups and neighbour subsets.

SMRF is a multicast forwarding algorithm that uses information provided by RPL's group membership information and addresses both those drawbacks. A node will accept an incoming multicast datagram if, and only if the datagram's link layer source address is the link layer address of the node's preferred parent, which can be looked up in the recipient's local neighbour cache. If the message passes this check then:

- It will get delivered up the local stack if, and only if the node is a member of the multicast group.
- It will get forwarded if, and only if there is an entry for the datagram's IPv6 destination address (multicast group) in the node's routing table ("a node under us is also a member").

A. Cross-layer optimizations

Channel Check Interval (CCI) is a duty cycling parameter which corresponds to the time between two consecutive radio-on cycles. SMRF introduces a short delay (D) between accepting a datagram and copying it forward. This delay is defined as $D = max(Fmin, CCI)$, where $Fmin$ is a configuration parameter and CCI is the value reported by the underlying duty cycling algorithm. Configuring SMRF with a non-zero value for $Fmin$ is particularly useful in the case of duty cycling algorithms with a $CCI = 0ms$, such as NullRDC, which keeps RF hardware always on.

In order to mitigate the negative effect of hidden terminals, SMRF can also optionally further delay the transmission by a random factor. This is parametrised on *Spread*, a positive integer. The final forwarding delay is a random number in $[D, Spread * D]$ with granularity equal to D. Table I outlines the resulting forwarding delays for various configuration values and duty cycling algorithms.

B. Benefits and Drawbacks

With SMRF, multicast traffic can only travel downwards in the DODAG. This makes the algorithm useful for applications such as code dissemination or network management. Since each node will only consider packets received from its preferred parent and will forward each packet at most once, it guarantees that each datagram can be received at most once per node, without need for a method of uniquely identifying messages.

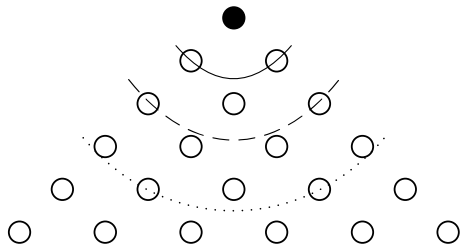


Figure 1. Simulated tree topologies differentiated by network density. *solid line* with $ND \approx 0.14$, *dashed* ($ND \approx 0.36$) and *dotted* ($ND \approx 0.71$). The *solid black* node acts as RPL root and multicast traffic source.

The gain in comparison to TM is multi-fold: SMRF uses multicast groups to differentiate between nodes that are interested in a flow and those that are not. Instead of blindly forwarding all datagrams to all nodes, multicast datagrams will only reach parts of the network with nodes that have expressed an interest in the flow by joining a group.

SMRF does not define any control messages of its own. It operates based on RPL parent information and on multicast group membership information, carried inside RPL DAO messages, as specified in [4]. SMRF is stateless. Nodes do not need to maintain per-packet information. A decision whether to forward a datagram or not is taken based on information available at the moment of its arrival, irrespective of other packets of the same flow. A positive side-effect of this "*on the spot*" approach is that SMRF will never entangle packet ordering. Compared to TM, SMRF achieves lower end-to-end delays and demonstrates better energy efficiency. This is further analysed in sections IV-B and IV-C.

The trade-off in order to achieve the aforementioned improvements, is a decrease in packet delivery ratio (increased packet loss), compared to TM which is by design more reliable. Packet delivery ratio is scrutinised for different traffic rates under multiple network topologies in section IV-A.

IV. EVALUATION

In order to evaluate the algorithms, we performed a series of experiments in Contiki's cooja simulator, with common parameters outlined in Table II. Our discussion in the following paragraphs uses the term *network density* (ND). In this context, ND is defined in the same way as the density of an undirected graph with edge set E and set of vertices V (eq. 1), with $ND \in [0, 1]$. $ND = 0$ for an *edgeless graph* and $ND = 1$ for a *complete graph*.

$$ND = \frac{2\,|E|}{|V|\,(|V| - 1)} \tag{1}$$

Table II
SIMULATION CONFIGURATION

Nodes	21 sky motes (1 traffic source, 20 sinks)
Radio Medium	Unit Disk Graph Medium (UDGM)
Ranges	TX: 50m, Interference: 60m
MAC Layer	IEEE 802.15.4
Duty Cycling	ContikiMAC & NullRDC
Iterations	10 for each parameter permutation
RNG Seeds	New seed each iteration
Duration	5 minutes of actual time each iteration
Traffic Pattern	CBR (exact rate discussed in text)
Message Size	4 app. layer bytes (variable number of bytes on link)
TM Params	$Imin \in \{125, 250, 375, 500, 625, 750\}\, ms$
SMRF Params	$Fmin = 0ms$ ($Spread$ is ignored)
	$Fmin = 31.25ms, Spread \in \{2, 4, 8\}$

This is a link layer metric: an edge between nodes A and B exists if, and only if the two nodes are single-hop neighbours (can directly hear each other) and under the assumption of symmetric links, which holds true for cooja's UDGM environment. If this assumption didn't hold then the network would have to be modelled as a directed graph, creating a need for a more complex density metric.

We ran our experiments in multiple tree topologies (Figure 1). By keeping node transmission range constant and by increasing the length of each edge, we can achieve topologies that allow us to examine the algorithms under different network densities. For each of those topologies, we experimented with two duty cycling algorithms: NullRDC and ContikiMAC (which is actually a duty cycling layer, despite the misleading 'MAC' suffix in its name). For TM, we used six different configurations of $Imin$ and for SMRF four different ($Fmin$, $Spread$) pairs (Table II). We ran ten iterations (each one with a different random seed) per topology, per configuration, per traffic rate. For each permutation we evaluated three metrics: i) packet delivery ratio, ii) end to end delay and iii) energy consumption.

From an application layer perspective, our multicast traffic was Constant Bit Rate (CBR) with a payload of 4 bytes. For each of the configurations above, we experimented with four multicast flows differentiated by the interval between two successive message transmissions (250, 500, 750ms and 1sec). As a result of IPv6 extension headers and 6LoWPAN header compression, layer two frames varied in size between 35 and 61 bytes. Thus, we use the inter-packet interval to refer to the flows, instead of bytes/sec.

A. Packet Delivery Ratio

As discussed in section II-A, TM is designed to be reliable, especially when operating in dense networks with high path redundancy. Investigating its behaviour for different $Imin$ values, we observe that packet delivery ratio can vary between perfect (0% loss) and extremely poor. The bar charts in Figure 2 and 3 illustrate the results for both algorithms over both duty cycling layers. In both cases,

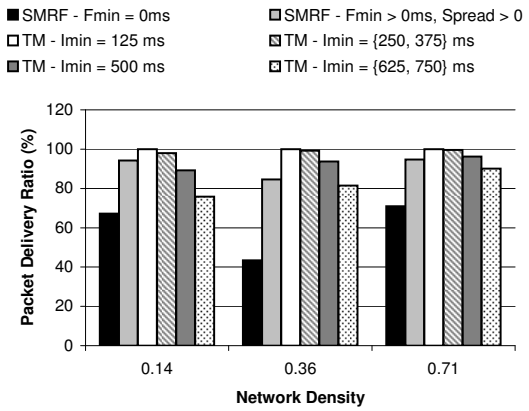


Figure 2. Packet Delivery Ratio over NullRDC.

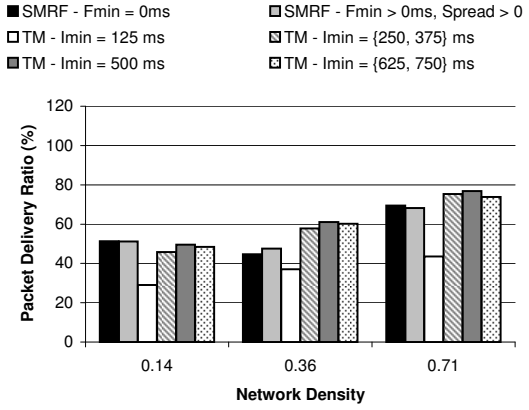


Figure 3. Packet Delivery Ratio over ContikiMAC.

configurations with similar results are averaged out as a single bar. Since inter-packet interval did not influence the results significantly, each bar presents the result average across all four different values.

In the case of a NullRDC network and when configured with $Imin = 125ms$, TM achieves 100% delivery ratio regardless of network density and traffic rate. Delivery ratio drops with higher $Imin$ values. Losses occur if a node cache is full when a new packet arrives. This happens when a new datagram overwrites an older one before the latter gets copied further down the line. SMRF on the other hand only forwards each datagram once and was expected to demonstrate a higher packet loss rate. Results confirm this, but they also indicate that loss rate is lower than what we anticipated. Over NullRDC, the (0 , 0) SMRF configuration severely under-performs its $Spread > 0$ counterparts. The reason is that $D = 0$ is very susceptible to hidden terminals, which was the original motivation behind introducing *Spread*.

We performed the same measurements over ContikiMAC and results are significantly different, as illustrated in Figure 3. The first observation is that with ContikiMAC, packet loss rates are higher across the board. This is caused

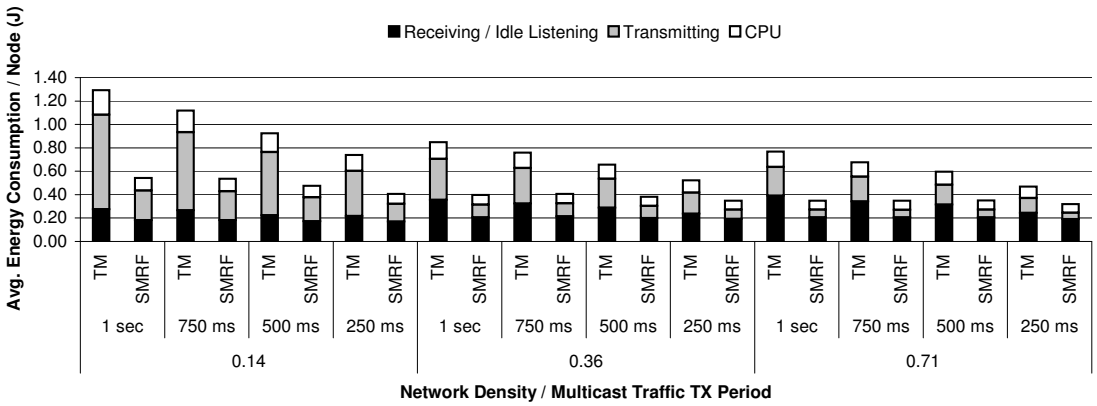
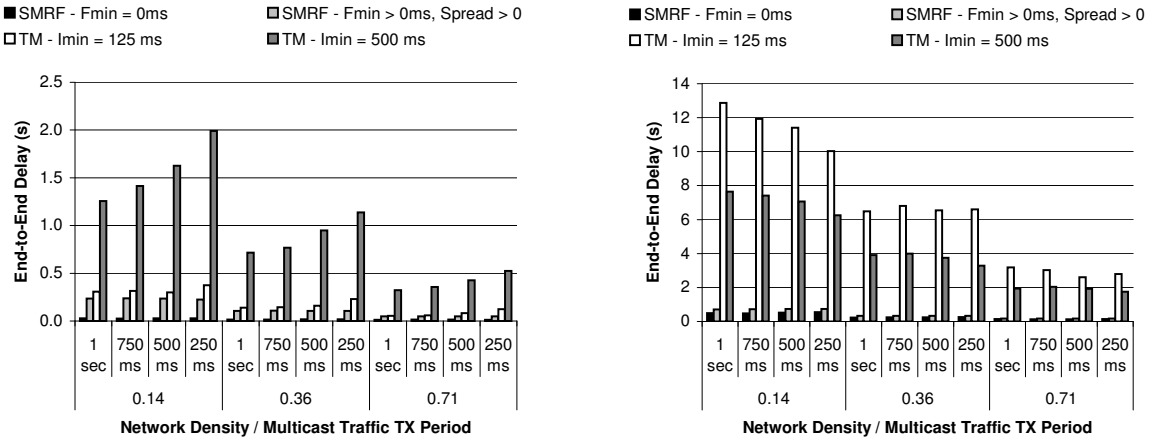


Figure 4. Energy Consumption for various network densities under different levels of traffic.



(a) NullRDC (b) ContikiMAC

Figure 5. End to end delay for different algorithm configurations, over both RDC layers. *Left*: Over NullRDC. *Right*: Over ContikiMAC.

by the fact that TM relies heavily on link-local multicast messages (link layer broadcasts) to exchange cache content information between nodes. Due to the technique used by ContikiMAC to transmit frames (*packet trains*), link layer broadcasts are fundamentally inefficient [11].

Over ContikiMAC, lowering TM's $Imin$ has an adverse result: delivery ratio decreases instead of increasing. $Imin = 125ms$ stands out as having underperformed compared to its counterparts, with $Imin = 500ms$ yielding optimal results for all densities. Additionally, results confirm TM's improvement with increasing density, a result of its ability to exploit path redundancy. Since packet trains indirectly mitigate the hidden terminal problem, the performance of SMRF's (0 , 0) configuration is comparable to the delivery ratio exhibited when $Spread > 0$. Depending on network density, SMRF can actually deliver more packets than TM when the latter is configured with a sub-optimal $Imin$.

B. End to End Delay

As discussed in sections II-A and III, after our initial analysis we anticipated TM to exhibit high delays compared to the very straightforward SMRF. Figure 5 illustrates the results over both duty cycling algorithms. For clarity, in the case of TM we have cherry-picked $Imin = 125ms$ and $Imin = 500ms$, the two values which performed best in terms of losses for NullRDC and ContikiMAC respectively.

For TM in the case of NullRDC we observe similar results as those for our analysis of Packet Delivery Ratio (Figure 5a). Reducing the $Imin$ value yields significant performance improvements: As $Imin$ decreases, trickle timers reset more often, nodes exchange cache content information more frequently and inconsistencies are detected earlier, leading to lower hop-by-hop forwarding delay. A similar observation applies for SMRF's *Spread*: Increasing its value effectively increases forwarding delay. The effect is augmented over multiple hops, leading to longer end to end delays. However, results confirm our expectations: Any SMRF configuration is faster than TM with $Imin = 125ms$, with gains increasing further when TM's $Imin$ is suboptimal.

For ContikiMAC, TM's end to end delay behaves in a similar fashion to packet delivery ratio with respect to $Imin$

values. Reducing $Imin$ has a positive impact until the value of 500ms. Any further reductions cause a radical performance degradation, with $Imin = 125ms$ severely under performing. The difference between the two algorithms is even more extreme than in the case of NullRDC, as shown in Figure 5b, with SMRF being over 5 times faster than TM. Same as in the previous section, this is a result of ContikiMAC's broadcast inefficiency due to *packet trains*.

C. Energy Consumption

Through the facilities provided by Contiki's energy consumption estimation module (energest) [12], we measured the time each node spent in each of the following three states over the duration of each experiment: i) MCU active, ii) RF listening / receiving, iii) RF transmitting. We then converted time values to estimated energy consumption based on typical datasheet levels at an operating voltage of 3.0V.

NullRDC keeps radio transceivers always on (no duty cycling). As a result, the majority of energy is consumed during idle listening or packet reception, with the remaining factors contributing insignificantly. For this reason, we only consider ContikiMAC for the evaluation of the two algorithms in terms of energy consumption. Figure 4 illustrates average (per-node) energy consumption algorithm under different multicast traffic rates. The main difference between the two algorithms is caused by radio transmissions, with TM consuming more energy in this state due to its periodic ICMPv6 control datagram exchange and due to the fact that each node may end up forwarding the same cached datagram multiple times (until all its neighbours have received it or until it gets replaced by a newer one in the node's cache). In the case of SMRF, radio reception and radio transmission contribute to total consumption at a ratio of about 1:1. Consumption attributed to micro-controller activity is also higher in the case of TM, providing an indication of the algorithm's increased complexity compared to SMRF.

In contrast to our anticipation, network-wide energy consumption decreases as multicast traffic rate increases. This may seem surprising but it is a reasonable side-effect of increased packet losses at higher traffic rates. Lastly, higher network densities lead to lower energy consumptions, with the effect being more substantial in the case of TM, since the algorithm is "*density-aware*" by design [5].

V. Conclusions and Future Work

In this work we have demonstrated that TM's performance and energy consumption are very sensitive to changes in the value of configuration parameter $Imin$, with the optimal depending on the choice of underlying duty cycling algorithm. On the other hand, SMRF is less susceptible to variances of this nature and is faster and more energy efficient in exchange for an occasional slight reliability drop.

We are currently in the process of conducting further simulations, investigating out-of-order packet arrivals with TM and the performance of both algorithms on a hop-by-hop basis. Additionally, we are running experiments on a hardware test bed in order to evaluate the accuracy of the simulated findings. Lastly, we are looking into the complexity, code size and memory footprint of both algorithms.

References

[1] J. W. Hui and D. E. Culler, "IP is dead, long live IP for wireless sensor networks," in *Proc. 6th ACM conference on Embedded network sensor systems (SenSys '08)*. New York, NY, USA: ACM, 2008, pp. 15–28.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944, Sep. 2007.

[3] J. Hui (editor) and P. Thubert, "Compression format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, Sep. 2011.

[4] T. Winter (editor), P. Thubert (editor), A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. P. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," IETF Internet Draft, Oct. 2010.

[5] J. Hui and P. Thubert, "Multicast forwarding using trickle," Internet Draft, Apr. 2011, (draft-ietf-roll-trickle-mcast).

[6] G. Oikonomou and I. Phillips, "Experiences from Porting the Contiki Operating System to a Popular Hardware Platform," in *Proc. 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, Barcelona, Spain, Jun. 2011.

[7] T. Clausen and U. Herberg, "Comparative Study of RPL-Enabled Optimized Broadcast in Wireless Sensor Networks," in *Proc. Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2010))*, Brisbane, Australia, Dec. 2010.

[8] D. Koutsonikolas, S. Das, Y. Hu, and I. Stojmenovic, "Hierarchical geographic multicast routing for wireless sensor networks," *Wireless Networks*, vol. 16, pp. 449–466, Oct. 2010.

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

[10] P. Levis, T. H. Clausen, J. Hui, O. Gnawali, and J. Ko, "The trickle algorithm," RFC 6206, Mar. 2011.

[11] A. Dunkels, L. Mottola, N. Tsiftes, Fredrik Österlind, J. Eriksson, and N. Finne, "The Announcement Layer: Beacon Coordination for the Sensornet Stack," in *Proc. European Conference on Wireless Sensor Networks (EWSN)*, Feb. 2011.

[12] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Demo abstract: Software-based sensor node energy estimation," in *Proc. Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, Nov. 2007.