

# IPv6 Multicast Forwarding in RPL-Based Wireless Sensor Networks

George Oikonomou · Iain Phillips · Theo Tryfonas

Published online: 12 June 2013  
© Springer Science+Business Media New York 2013

**Abstract** In wireless sensor deployments, network layer multicast can be used to improve the bandwidth and energy efficiency for a variety of applications, such as service discovery or network management. However, despite efforts to adopt IPv6 in networks of constrained devices, multicast has been somewhat overlooked. The *Multicast Forwarding Using Trickle* (Trickle Multicast) internet draft is one of the most noteworthy efforts. The specification of the *IPv6 routing protocol for low power and lossy networks* (RPL) also attempts to address the area but leaves many questions unanswered. In this paper we highlight our concerns about both these approaches. Subsequently, we present our alternative mechanism, called stateless multicast RPL forwarding algorithm (SMRF), which addresses the aforementioned drawbacks. Having extended the TCP/IP engine of the Contiki embedded operating system to support both trickle multicast (TM) and SMRF, we present an in-depth comparison, backed by simulated evaluation as well as by experiments conducted on a multi-hop hardware testbed. Results demonstrate that SMRF achieves significant delay and energy efficiency improvements at the cost of a small increase in packet loss. The outcome of our hardware experiments show that simulation results were realistic. Lastly, we evaluate both algorithms in terms of code size and memory requirements, highlighting SMRF's low implementation complexity. Both implementations have been made available to the community for adoption.

---

G. Oikonomou (✉)  
Faculty of Engineering, University of Bristol, Merchant Venturers Building,  
Woodland Road, Clifton BS8 1UB, UK  
e-mail: g.oikonomou@bristol.ac.uk

G. Oikonomou · I. Phillips  
Computer Science, Loughborough University, Loughborough LE11 3TU, UK

I. Phillips  
e-mail: i.w.phillips@lboro.ac.uk

T. Tryfonas  
Faculty of Engineering, University of Bristol, Queen's Building,  
University Walk, Clifton BS8 1TR, UK  
e-mail: theo.tryfonas@bristol.ac.uk

**Keywords** 6LoWPAN · Wireless sensor networks · IPv6 multicast · Trickle

## 1 Introduction and the Need for Multicast in Wireless Sensor Networks

Over the past years, the research community has 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 standardization 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, in order to achieve their efficient transmission within IEEE 802.15.4 low power radio frames. For those networks (6LoWPANs), the most widely adopted standard for routing is the "IPv6 routing protocol for low-power and lossy networks" (RPL), which is specified in RFC 6550 [4].

The importance of network layer multicast forwarding in 6LoWPANs stems from its ability to improve the efficiency of applications adopting a one-to-many communication paradigm. Examples of services which can benefit by multicast include service discovery [5,6], network management and publish/subscribe schemes. For instance, extended multicast DNS (xmdns) builds on the mDNS specification [7] and expands it "*to site-local scope in order to support multi-hop LANs that forward multicast packets but do not provide a unicast DNS service*" [8]. uBonjour [6] is a service discovery scheme for resource-constrained devices in the wireless embedded internet. It is essentially Bonjour's lightweight variant and is based on multicast DNS (mDNS) and DNS service discovery (DNS-SD) [9], which have recently had their message sizes optimised for 6LoWPANs [10]. Additionally, the constrained application protocol (COAP) is an emerging standard aiming at the integration of WSN devices with the web. It operates over UDP (mandatory) or TCP (optionally) and has been designed so that its messages can be easily translated to HTTP. It targets embedded devices with severe memory and power supply restrictions. The current version of its specification provides an extensive discussion on its operation over IPv6 multicast [11].

Previous research efforts in the area of multicast for WSNs focus on bespoke network stack designs and do not investigate IPv6-specific challenges. The majority of multicast forwarding algorithms encountered in current literature are based on geographic routing [12–17]. However, most of those approaches have certain characteristics and make assumptions which render them unsuitable for IPv6-based WSN deployments. For instance, many of them assume that, for every multicast message, the sender is aware of the addresses or IDs of all intended destinations. Additionally, some efforts suffer from poor scalability while others rely on unrealistically large network packets. Lastly, they are only applicable in situations where the source as well as all destinations are within the WSN boundaries.

Despite the number of existing efforts, IPv6-based multicast has been somewhat overlooked by the 6LoWPAN research community, as we discuss further in Sect. 2. The "Multicast Forwarding Using Trickle" internet draft (Trickle Multicast—TM) [18] discusses an algorithm which poses among the most suitable candidates. The RPL RFC also briefly mentions multicast, but the discussion focuses on group management without providing a sufficient level of detail in terms of forwarding.

The open issues outlined above have motivated us to design and implement SMRF, a lightweight Stateless Multicast RPL Forwarding algorithm. In this paper, we disclose its design in depth and we highlight how it addresses current open issues while maintaining high speed,

energy efficiency and low complexity. Compared to geographic multicast algorithms, SMRF does not require geolocation information (neither explicit nor via a location service) and does not suffer from the aforementioned scalability and datagram size issues. Moreover, by recommending a forwarding algorithm, SMRF fills the gaps left open by the RPL RFC. In this context, this paper's contributions are the following:

- We compare the performance of SMRF against TM. For the evaluation, we consider four metrics: (1) Packet delivery ratio, (2) End-to-end delay, (3) Out-of-order datagram delivery ratio and (4) Energy consumption. Evaluation is performed through simulations as well as on a multi-hop hardware testbed. We also investigate the complexity of the two algorithms and we compare their code size and memory requirements. Simulation and testbed results demonstrate that SMRF is less complex and that it outperforms TM on three of the four metrics.
- Based on the outcome of the comparative evaluation, we present a host of criteria, which can be used by network engineers and designers in order to select the more suitable between SMRF and TM, depending on their deployment's specific needs.
- We have extended Contiki's TCP/IP stack to support both algorithms. Both implementations have been released<sup>1</sup> to the community for adoption and further scrutiny as a part of our port of the Contiki OS<sup>2,3</sup> [19].

This paper extends our previous work [20], providing the following additional contributions: (1) Extended design details for SMRF, (2) Evaluation of an additional metric: the ratio of datagrams delivered out of order by TM (Sect. 5.4), (3) Additional simulation experiments in a different topology for the evaluation of both algorithms on a hop-by-hop basis (Sects. 5.2.1, 5.3.1 and 5.5.1), (4) Results from the evaluation of both algorithms on a hardware testbed (Sect. 6) and lastly (5) Discussion on the code size and memory requirements for both algorithms (Sect. 7).

## 2 Related Work

### 2.1 Multicast in Traditional WSNs

Previous research efforts in the area of multicast for WSNs have primarily been focusing on traditional, application-centric network designs and as such do not address IPv6-specific challenges. Multicast forwarding algorithms based on geographic routing are dominant in existing bibliography and can be broadly classified as either *purely geographic* [12–15] or *hybrid* [16,17], whereby the geographic component is complemented by features of other approaches, such as hierarchical routing.

The geographic multicast routing (GMR) algorithm builds on existing unicast geographic routing approaches. By adapting them, it aims to achieve multicast message delivery to all intended destinations while maintaining minimum bandwidth consumption. Nodes exchange position information with their neighbours through periodic beacons. GMR is characterised by low computational complexity and a small memory footprint [12]. According to subsequent works, GMR scales better than some of its predecessors [13] but still suffers from scalability issues when dealing with large deployments [16].

<sup>1</sup> <https://github.com/g-oikonomou/contiki-sensinode/tree/mcast-forward>.

<sup>2</sup> <https://github.com/g-oikonomou/contiki-sensinode/wiki>.

<sup>3</sup> <http://nets-www.lboro.ac.uk/george/contiki-sensinode/>.

Using periodic beacons can have negative side-effects such as collisions and increased energy consumption [13]. Based on this observation, BRUMA attempts geographic multicasting without beacons, whereby neighbour positions are discovered reactively. Next hop selection happens opportunistically through a mechanism which only requires a low number of control messages. The authors demonstrate that BRUMA is more efficient than GMR [13].

Carzaniga et al. [14] propose a compact and completely decentralised multicast scheme with asymptotically optimal network congestion properties. It operates by building a multicast forwarding tree over an underlying geographic unicast routing service, which allows nodes to send messages to a destination defined by a coordinate pair  $(x, y)$ .

Receiver-based multicast (RBMulticast) [15] is another purely geographic approach. Its principal novelty lies in the next-hop determination phase: potential next hops contend for the channel based on their contribution towards delivering a packet to its destination. Nodes offering the highest forward progress have higher probability of getting selected as next hop. By adopting this approach, RBMulticast can operate without routing tables and without maintaining a forwarding tree. To achieve this, RBMulticast embeds the geographic location of all destinations in the packet header. This raises questions regarding its scalability in large deployments.

The hierarchical geographic multicast routing (HGMR) [16] is a hybrid algorithm combining the key design concepts of GMR [12] and the hierarchical rendezvous point multicast (HRPM) protocol [21]. The resulting HGMR algorithm is further optimised to be more energy efficient and scalable. HGMR divides multicast groups into subgroups by using HRPM's geolocation hashing. It takes advantage of layer 2 reliability mechanisms by using HRPM's unicast forwarding approach for long, sparse paths and reverts to layer 2 broadcasting in areas of high density in order to reduce the number of transmissions. When unicast forwarding is in use, HRPM (and therefore HGMR) uses source routing along the branches of an overlay tree generated by the traffic source. In this work the authors conducted a performance evaluation of the three algorithms in a simulated IEEE 802.11 network. It is therefore difficult to understand how the algorithms would behave under the frame size and bandwidth limitations or the loss characteristics related to IEEE 802.15.4 networks.

The multicast routing with branch information nodes (MR.BIN) [17] protocol is a hybrid approach, combining geographic unicast routing with state-based multicast. It maintains multicast states only on branch nodes of the forwarding tree. Communication between non-branching nodes takes place with geographic unicast. In order to perform datagram forwarding, each node maintains a potentially long list of next hops. Messages are tagged with a 2-byte multicast group identifier but the management scheme for those IDs is not discussed; it is unclear how a node can choose which group ID to join and how two different groups are prevented from having the same ID.

Adaptive geo-source multicast routing (AGSMR) [22] is a geographic unicast and source multicast hybrid. It relies on generating a forwarding tree at the traffic source. Path information is embedded in packet headers in a compressed format which uses 2 bytes per hop, an additional 2 bytes per branch and reduced by  $2(n - 2)$  bytes for each  $n$ -node long, non-branching path. The authors demonstrate that, with this compression scheme in place, AGSMR's packets are smaller than GMR's. However, reported packet sizes have an order of magnitude of KBytes (e.g. about 9 KB for a tree with about 50 subscribers in a network of 1024 nodes). Packet size increases with the number of subscribers as well as the total number of network nodes.

Branch aggregation multicast (BAM) [23] differs from the aforementioned efforts in that it operates without knowledge of node geolocation. BAM's design has two components: S-BAM achieves single-hop aggregation at branching nodes and M-BAM aims to reduce the number

of branches. The former tries to combine multiple layer 2 unicasts within a single radio frame, by extending headers to list the addresses of all intended destinations. Recipients reply with an acknowledgement frame, using a random back off to avoid ACK collisions. M-BAM relies on existing forwarding tables, which can be populated by any routing protocol. The authors make the assumption that the routing table contains multiple candidate next hops for the same destination and they propose an algorithm for path aggregation. However, routing table size has an impact on scalability with increasing network size [19] even when the table only lists a single next hop per destination. Storing multiple candidate next hops per destination would impose further memory overheads<sup>4</sup> and should not be considered common practise. Furthermore, BAM does not manage multicast groups internally. Instead, it assumes that the network adopts a data centric routing model, whereby nodes broadcast requests for data and their neighbours remember querying node addresses in order to forward relevant data accordingly [23].

All aforementioned algorithms assume that the message source is aware of and maintains a list of all destinations, uniquely identified by an attribute such as a node ID, address or name. This is untrue in the case of IP and IPv6 multicast, whereby datagram destination is expressed as the multicast group's IP or IPv6 address and the traffic source is oblivious with respect to the unicast address of each individual group member.

The GMR protocols discussed above also require knowledge of the geolocation of all destinations. When this information is not available, a geographic routing service can be used in conjunction with a location service, such as MLS [24] or GLS [25]. Querying the location service incurs time overheads which have an impact on delivery delay, compared to non-geographic approaches which do not rely on this information. However, this is not considered at all in relevant work.

Many of the above multicast forwarding algorithms embed a list of all destinations in the packet header [12, 13, 15, 17, 22, 23]. This increases the byte overhead associated with each data transmission and has an adverse impact on the protocol's scalability [16].

Lastly, with all those approaches traffic is confined within the deployment's boundaries. In order to be able to communicate beyond the WSN's borders, one would need a dedicated gateway. This does not apply to 6LoWPANs and IPv6 multicast forwarding protocols, such as those discussed later on in this paper.

## 2.2 Multicast in 6LoWPANs

Previous research with focus on 6LoWPANs is very limited. Sá Silva et al. [26] investigated the applicability and usefulness of traditional multicast paradigms in WSNs. In this work, the authors evaluate multicast ad hoc on-demand distance vector (MAODV) [27] and source-specific multicast (SSM) [28] in a simulated sensor network. They demonstrate that, in their simulated scenarios, multicast can offer significant advantages over traditional network-wide broadcast flooding and unicast. Results also suggest that MAODV and SSM are comparable in terms of bandwidth consumption and energy efficiency.

Clausen and Herberg [29] conducted relevant research with focus on RPL networks. Despite the fact that this work only discusses network-wide broadcast, it offers some important insight on related issues and techniques. In their contribution, the authors conduct experiments by simulating an IEEE 802.11b network, which cannot capture the duty cycling aspects of modern WSNs, nor the low-power, lossy nature of IEEE 802.15.4-compliant radio hardware.

<sup>4</sup> In the version of the Contiki OS used for this research, each entry in the IPv6 routing table occupies approximately 48 bytes of RAM, the exact number depending on the hardware platform and toolchain.

The RPL RFC [4] briefly discusses built-in multicast support. In RPL networks, nodes advertise unicast downward paths inside destination advertisement object (DAO) messages. An RPL instance is administratively configured with one out of a possible four modes of operation (MOP). Nodes may only participate in the network as routers if they support the advertised MOP, otherwise they may only join as leaf nodes. In the *Storing with multicast support* MOP, DAO messages are also used for group management by advertising multicast prefixes. Unfortunately, the approach leaves many open issues, as we discuss in greater detail in Sect. 4.

### 3 Multicast Forwarding with Trickle

In wired networks, multicast mechanisms rely on topology maintenance in order to forward packets to their intended destinations. Due to memory restrictions, this is a very challenging task in networks of constrained nodes. The *multicast Forwarding with Trickle* algorithm (Trickle Multicast—TM) addresses this challenge by providing a means of supporting IPv6 multicast without having to rely on topology information. To control the frequency of datagram exchange, TM adopts the pre-existing Trickle algorithm.

Trickle [30,31] is a mechanism that governs the frequency of periodic information exchange among neighbouring nodes in a low power, lossy network. Strictly speaking, trickle only specifies the dynamic behaviour of periodic timers. Its aim is to provide a method of propagating state information efficiently, without constantly flooding the network with control messages. 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 an inconsistency is detected, the timer's interval ( $I$ ) is reset to a minimum value (called  $I_{min}$ ) and changes propagate within milliseconds. After a long period without changes, the trickle timer reaches its maximum interval called  $I_{max}$ , which is expressed as the maximum number of  $I_{min}$  doublings. Thus, for a trickle timer configured with  $(I_{min}, I_{max})$ , the maximum trickle interval in time units will be  $I_{min} * 2^{I_{max}}$  and can result in very infrequent message exchange, with nodes sending only a few packets per hour [31]. Trickle only dictates the behaviour of timers, in other words *when* nodes should exchange messages, not *how* nor their format. This trait makes it very attractive for any protocol involving periodic exchange of state information. Trickle was originally designed for data dissemination and network reprogramming [32] and has been adopted by multiple works, such as Deluge [33] and DIP [34]. It also handles the frequency of RPL DIO (upward route advertisement) messages [4]. Lastly, trickle underpins the TM forwarding algorithm which is discussed in the remainder of this section.

With TM, each multicast datagram must carry a *Multicast Option* header in the shape of an IPv6 hop-by-hop option (HBHO) extension header. The multicast option tags the packet with a sequence number, a single-bit  $M$  parameter and the unique identifier (Seed ID) of the sender, which may be different than the datagram's IPv6 source address. The contents of the multicast option remain unchanged en route. Each network node maintains a cache of recently seen multicast packets, uniquely identified by the information in 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 use ICMPv6 datagrams to exchange information about their cache contents, at a frequency controlled by trickle timers. If a node's cache contents don't match the information in a received ICMPv6 datagram, the node resets its trickle timer to its minimum interval ( $I_{min}$ ) 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.

### 3.1 Advantages

By design, TM has some very significant advantages. More specifically:

*Generality* TM will work, without modifications, alongside any routing protocol.

*Reliability* By caching datagrams and maintaining per-packet state information, TM increases its reliability (high packet delivery ratio / low loss). The exact reliability levels are heavily influenced by the choice of *I<sub>min</sub>* and the underlying duty cycling algorithm.

*Guaranteed no duplicates* With the help of the HBHO, TM guarantees that each node in the network will receive each individual datagram at most once.

### 3.2 Concerns

The design of the TM algorithm raises a number of concerns which are outlined below and further analysed in the evaluation sections (Sects. 5, 6, 7).

*Scalability* One of the arguments for specifying the TM algorithm in the first place was that maintaining topology information is hindered by memory constraints. The algorithm bypasses this requirement and as a result scales well with the number of nodes in a network. However, topology maintenance is replaced with bespoke, per-packet state maintenance. This raises concerns regarding scalability with traffic volume, cache size and number of multicast traffic sources.

*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 parameters. The trickle RFC [31] dictates that “A protocol specification that uses Trickle MUST specify: Default values for *I<sub>min</sub>*, *I<sub>max</sub>*, and *k...*”. Currently, this is not the case for TM; its internet draft only outlines examples with indicative values [18]. Furthermore, as we demonstrate in Sects. 5 and 6, the values used in these examples are sub-optimal. We outline alternative recommendations, supported by experimental results.

*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 complexity, code size and memory requirements. This is further investigated in Sect. 7.

*Multicast versus Broadcast* Due to lack of topology maintenance and group registrations, TM forwards all multicast messages to all parts of the network, irrespective of whether they are needed or not. Any datagram with a routable multicast IPv6 destination address is in practice treated as a *network-wide broadcast*. In sparse multicast topologies (where only a small percentage of nodes is interested in a multicast flow), adopting TM leads to energy and bandwidth inefficiencies.

*Arrival Order* Due to its store and forward nature and per-packet state maintenance, TM is susceptible to out-of-order datagram arrivals. Depending on the application relying on multicast, this trait may or may not be a problem. For instance, code dissemination applications

generally distribute an image in a number of chunks. Changes only get committed after all chunks have been received successfully, thus arrival order is highly irrelevant. However, in other types of applications (e.g. network management) a packet arriving out of order could cause undesirable behaviour. In this case, the application would have to employ a technique to detect out of order datagram delivery and mitigate its impact.

*Trickle Multicast in RPL Networks* Even though there is no technical reason preventing TM and RPL from happily coexisting in the same network, contradiction in the respective specifications makes the situation less clear. More specifically, the TM internet draft specifies that “*The TM option is carried in an IPv6 Hop-by-Hop Options header, immediately following the IPv6 header*” [18]. The RPL RFC specifies that all data plane datagrams also carry an HBHO, used for loop detection and avoidance. The format and functionality of this header is further specified in a separate RFC, which states that “*The RPL Option is carried in an IPv6 Hop-by-Hop Options header, immediately following the IPv6 header*” [35]. With both HBHOS immediately following the IPv6 header, co-existence of the two protocols becomes less straightforward.

#### 4 Stateless Multicast RPL Forwarding—SMRF

In this paper, we contribute a multicast forwarding algorithm called SMRF, as an alternative to TM for RPL networks. The principal rationale behind SMRF is that nodes participating in an RPL network exchange topology information in order to build the basic RPL construct (called a destination-oriented directed acyclic graph—DODAG) and to populate their routing tables. The DODAG is a tree structure and is thus particularly attractive to form the basis of multicast forwarding. Since network nodes perceive the network as a tree, we can capitalise on it in order to perform multicast forwarding without defining new message formats.

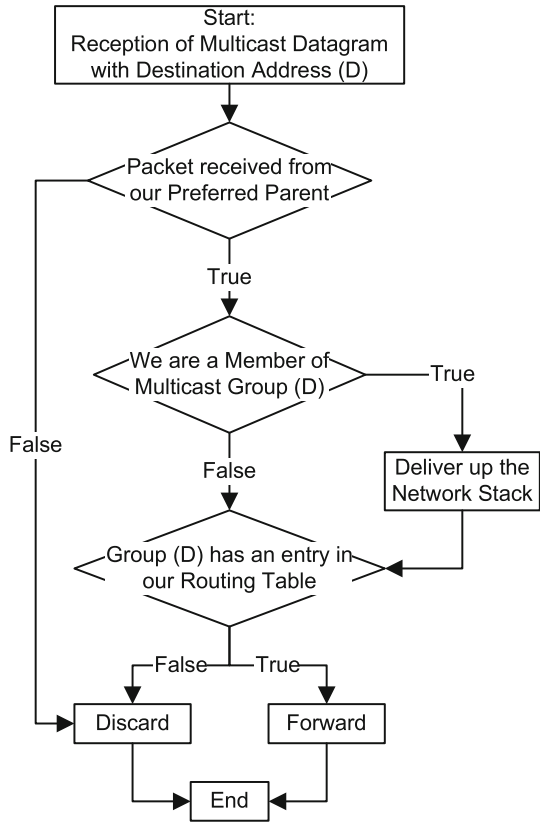
IPv6 routing protocol for low power and lossy networks (RPL) nodes advertise downward paths inside DAO messages (data messages in RPL networks can only flow up or down the DODAG). According to its specification, one of RPL’s MOP is *Storing with multicast support*. In this MOP, unicast DAO messages are also used to relay multicast group registrations up the DODAG. Those DAOs are identical to the ones conveying unicast information except for the type of prefix being advertised, which is a routable multicast IPv6 address.

Nodes can join a multicast group by advertising the group’s multicast 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 routing table for the advertised multicast address. Conceptually, this entry indicates that *a node under us in the DODAG is a member of this group*. This router will then (1) advertise this prefix in its own DAOs and (2) relay multicast datagrams addressed to this destination.

This RPL built-in mechanism solves the problem of propagating group membership information towards the DODAG root. However, it suffers from two limitations: (1) It lacks a method which would prevent a node from accepting the same datagram twice or more. (2) The RPL RFC 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 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.



**Fig. 1** The SMRF algorithm

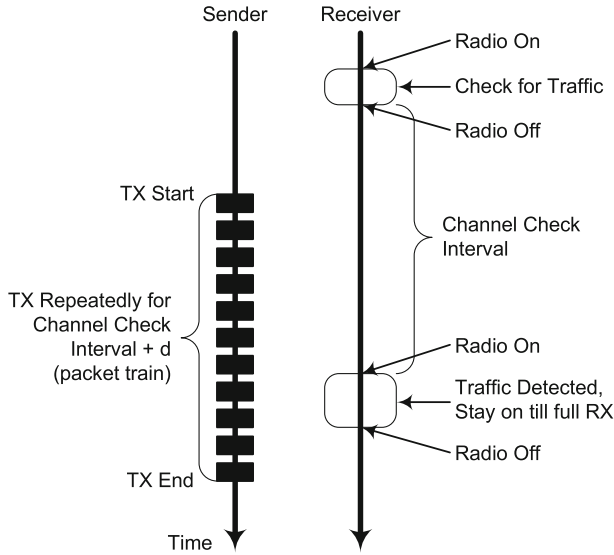


Stateless multicast RPL forwarding algorithm (SMRF) is a multicast forwarding algorithm that uses information provided by RPL’s group membership scheme and addresses both those drawbacks. Its operation, illustrated in the flowchart in Fig. 1, is the following:

- 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 RPL parent (which can be looked up in the node’s neighbour cache).
- If the message gets accepted, it will get delivered up the network stack locally if and only if the node is a member of the multicast group.
- If the message gets accepted, 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 below us in the DODAG is a group member).

#### 4.1 Cross-Layer Optimizations

With SMRF, multicast datagrams are always transmitted as layer 2 broadcast frames, with some cross-layer optimisations in place so as to improve performance. To better understand how these work, it is necessary to visit the concept of duty cycling and 802.15.4 frame transmission with ContikiMAC [36,37], which is one of the main duty cycling algorithms used by the Contiki OS. Despite the misleading ‘MAC’ suffix, ContikiMAC is actually a duty cycling mechanism which can operate in conjunction with a variety of MAC layers (e.g. CSMA). In



**Fig. 2** Broadcast packet transmission with ContikiMAC

very simple terms, each node wakes up every few milliseconds and checks the channel for traffic. This interval is called channel check interval (CCI) or Channel Sampling Period. If no traffic is present, the radio transceiver is turned back off. If traffic is detected, the node stays on until complete reception. To send a frame, a node will transmit it repeatedly (strokes) for slightly longer than CCI, waiting for a brief time interval between two strokes for a potential acknowledgement frame (ACK). This repeated transmission, often called a *packet train*, lasts long enough for intended recipients to wake up, detect the packet and receive it, irrespective of exactly when they last went to sleep. This removes the complexity of maintaining synchronisation between neighbours. In the case of unicast packets, the receiver will send an ACK frame, causing the sender to terminate its chain of strokes and thus conserve energy. However, broadcast frames must be received by all neighbours and there are no ACKs; the sender always has to go through the entire packet train, as illustrated in Fig. 2. Thus, broadcast transmissions are fundamentally more costly. The actual implementation of the ContikiMAC algorithm is more sophisticated and optimised (e.g. phase locks, burst support), but the concept remains the same.

A side effect of the algorithm discussed above is that when a node receives a broadcast frame, it should not attempt to transmit before the sender has gone through its entire packet train. Immediate transmission would signal a collision and the outgoing packet would be dropped. Thus, SMRF introduces a short delay ( $D$ ), defined as  $D = \max(Fmin, CCI)$  where  $CCI$  is the Channel Sampling Interval as reported by the underlying duty cycling algorithm and  $Fmin$  is a configuration parameter. Configuring SMRF with a non-zero value for  $Fmin$  is particularly useful in the case of duty cycling algorithms which keep RF hardware always on ( $CCI = 0$ ms). One such duty cycling algorithm is Contiki's NullRDC.

In order to mitigate the negative effect of hidden terminals, SMRF can also optionally further delay datagram forwarding 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 1 outlines the resulting forwarding delays for various configuration values and duty cycling algorithms.

**Table 1** Examples of SMRF configuration parameters and resulting forwarding delay

Duty cycling		Configuration		Outcome	
Algorithm	<i>cci</i> (ms)	<i>Fmin</i> (ms)	<i>Spread</i> (ms)	<i>D</i> (ms)	Final delay (ms)
ContikiMAC	125	≤125	1	125	125
ContikiMAC	125	≤125	4	125	[125, 500]
NullRDC	0	0	Ignored	0	Immediate
NullRDC	0	31.25	8	31.25	[31.25, 250]

### 4.2 Benefits and Drawbacks

With SMRF, multicast traffic can only travel downwards in the DODAG. This makes the algorithm useful for applications such as service discovery or network management. Since each node will only consider datagrams 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.

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 that have expressed an interest in the flow by joining a multicast group.

Stateless multicast RPL forwarding algorithm (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 defined in [4].

Nodes do not need to maintain per-packet state. A *drop or forward* decision is taken for each datagram individually and is based on information available at the moment of its arrival. A positive side-effect of this *on the spot* approach is that SMRF can't entangle datagram ordering. Thus, unless message order gets shuffled around by an underlying layer, SMRF will deliver them in the correct order.

Stateless multicast RPL forwarding algorithm (SMRF) is very lightweight in terms of complexity, code footprint and memory requirements, as demonstrated in Sect. 7. This makes it a very attractive option for severely constrained hardware.

Compared to TM, SMRF achieves lower end-to-end delays and demonstrates better energy efficiency. This is further analysed in Sects. 5.3 and 5.5.

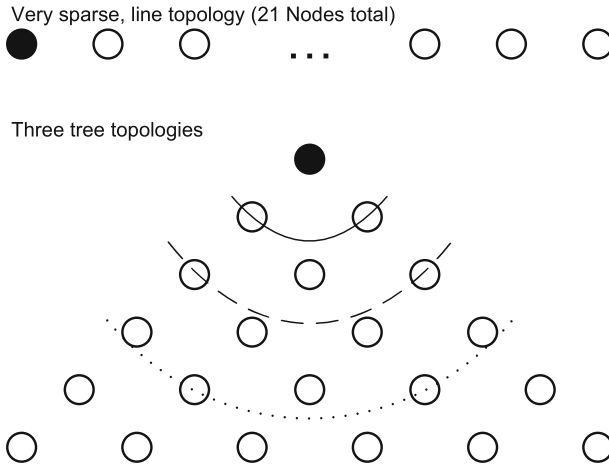
The trade-off in order to achieve the aforementioned improvements, is a decrease in packet delivery ratio (increased packet loss), compared to TM which is by design more reliable. Packet delivery ratio is scrutinised for different traffic rates under multiple network topologies in Sect. 5.2.

## 5 Simulation Results

In order to evaluate the algorithms, we performed a series of experiments in Contiki's Cooja simulator, with common parameters outlined in Table 2. Our discussion in the following paragraphs uses the term *network density*. In this context, network density  $ND$  is defined in the same way as the density of an undirected graph with edge set  $E$  and set of vertices  $V$  (Eq. 1).  $ND$  can take values between 0 and 1 inclusive ( $ND = 0$  for an *edgeless graph* and  $ND = 1$  for a *complete graph*).

**Table 2** Simulation configuration

Nodes	21 Sky notes (1 traffic source, 20 sinks)
Radio medium	Unit disk graph medium (UDGM)
Ranges	TX: 50 m, interference: 60 m
PHY and MAC	IEEE 802.15.4 with CSMA
Duty cycling	ContikiMAC (CCI 125 ms) & NullRDC
Iterations	10 for each parameter permutation
RNG seeds	New seed each iteration
Duration	5 min of actual time each iteration
Traffic pattern	CBR (exact rate discussed in text)
Message size	4 app. layer bytes (variable size on link)
TM Params	$I_{min} \in \{125, 250, 375, 500, 625, 750\}$ $k = 3, I_{max}$ variable (see text)
SMRF params	$F_{min} = 0$ ms, $Spread = 1$ $F_{min} = 31.25$ ms, $spread \in \{2, 4, 8\}$



**Fig. 3** Simulated topologies differentiated by network density. Very sparse line topology with  $ND \approx 0.09$ . Three tree topologies: *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

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

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). This definition only makes sense if radio links are symmetric, which is true for Cooja’s UDGM environment (Table 2) but not always the case for real deployments. In case of non-symmetric links (e.g. when A can hear B, but B cannot hear A), the link layer topology would have to be modelled as a directed graph. Investigating the behaviour and performance of multicast algorithms in an environment with non-symmetric links is part of our future plans.

## 5.1 Simulation Topologies and Configuration

We ran our experiments in four different topologies, as illustrated in Fig. 3: (1) A topology where all devices are placed in a line 40m apart, (2) Three different tree topologies differentiated by network density. In all scenarios, all network nodes are configured to join the multicast group in order to provide us more detailed measurements.

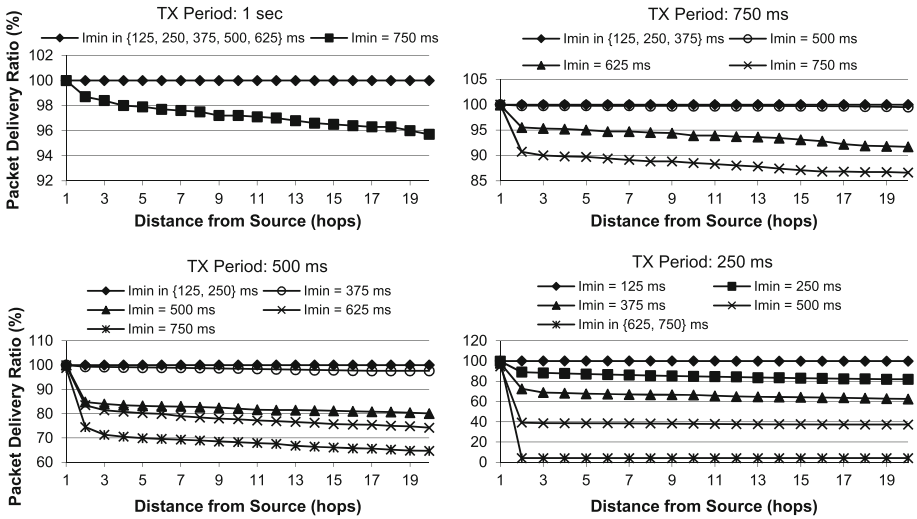
In the case of the line topology, the chosen maximum transmission range of 50m (Table 2) meant that each node could directly exchange radio messages with a maximum of two nodes. This topology helps us examine the algorithms in an extremely sparse topology ( $ND \approx 0.09$  for 21 nodes). Another useful feature is that the shape of the RPL DODAG is predictable: it always matches the physical topology. Previous works evaluating network protocols on a per-hop basis have been limiting their investigation to seven or eight hops [38]. Thus, the selected maximum distance of 20 hops is considered adequate.

For each of those topologies, we experimented with two radio duty cycling algorithms: NullRDC and ContikiMAC. Choosing these two mechanisms brings out potential performance differences between a duty cycling network (ContikiMAC) and a network where nodes keep their radios always on (NullRDC). ContikiMAC typically operates with a channel sampling rate of 16Hz [37] or 8Hz [36], resulting in wake-up intervals of approximately 62.5 and 125ms respectively. In our simulations we used a sampling rate of 8Hz, which is the default value used by Contiki's port for Sky notes.

For TM, we used six different configurations of  $I_{min}$ , also listed in Table 2. For each value of  $I_{min}$ , we set  $I_{max}$  so that the longest possible trickle interval ( $I_{min} * 2^{I_{max}}$ ) would not overflow the boundaries of the variable holding its value. For SMRF, we simulated four different ( $F_{min}$ ,  $Spread$ ) pairs (Table 2). 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: (1) packet delivery ratio, (2) end to end delay and (3) energy consumption. For TM only, we also investigated the ratio of out of order datagram deliveries.

From an application layer perspective, our multicast traffic was constant bit rate (CBR) with a payload of 4bytes. 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 1s). As a result of 6LOWPAN header compression, the number of actual bytes on link would vary per hop: a message leaving the source has a shorter on-link length than when copied beyond the first hop. Furthermore, TM adds 8 bytes to each datagram in the shape of a HBHO, which also prohibits UDP header compression. As a result, layer two frames varied in size between 35 and 61bytes. For these reasons, we use the inter-packet interval to refer to the flows, instead of bytes/s.

Operating the network under very heavy traffic load is intentional; it brings out algorithm advantages and drawbacks, allowing us to draw conclusions on their performance. Additionally, as 6LOWPANS progress towards general-purpose multi-service deployments, protocols are expected to cope with traffic originating from internet hosts and it is not uncommon to encounter applications with higher throughput requirements. For instance, to adapt to these changing requirements, traditional duty cycling algorithms which relied on the assumption of low data rates are now evolving in order to perform efficiently under high and bursty traffic patterns [39]. It is thus becoming increasingly common to conduct WSN protocol evaluation under conditions of heavy network load [39,40].



**Fig. 4** Packet delivery ratio for TM over NullRDC in the line topology. Each graph illustrates results for a different Multicast Traffic rate. Lines correspond to an individual or a group of *Imin* values, as indicated in the legend. Notice the different Y-axis scale for each subfigure

### 5.2 Packet Delivery Ratio and Loss

For each multicast group, we calculate packet delivery ratio (*PDR*) with Eq. 2, where *N* is the number of unique multicast datagrams sent by traffic sources to the group’s IPv6 address, *M* is the number of multicast group members and *R<sub>i</sub>* is the number of unique multicast datagrams sent to this group and received correctly by node *i*. This equation is valid under the assumption that multicast group membership remains unchanged throughout the multicast flow’s entire lifecycle, which holds true in the experiments presented here.

$$PDR = \frac{\sum_{i=1}^M R_i}{N \times M} \tag{2}$$

Packet delivery ratio (*PDR*) is an application layer metric and can take values in [0, 1]. Packet loss is calculated as 1 – *PDR*, thus packet loss 0% is the equivalent of 100% *PDR*, which means that all multicast datagrams were received correctly by all multicast group members.

#### 5.2.1 Line Topology

Investigating the behaviour of TM for different *Imin* values, we observe that packet delivery ratio can vary between perfect (0% loss) and extremely poor. The four graphs in Fig. 4 illustrate the results for four different datagram transmission rates in the line topology over NullRDC. When configured with *Imin* = 125 ms, TM achieves 100% delivery across all 20 hops regardless of traffic rate. Delivery ratio drops rapidly with higher *Imin* values and with higher transmission rates. Losses occur when node caches are full and a new packet arrives, overwriting an older one before the latter gets copied further down the line. Since there is no path redundancy in this topology, a packet lost in this manner cannot be recovered. In all scenarios, a very heavy packet loss increase is observed when *Imin* becomes higher than the

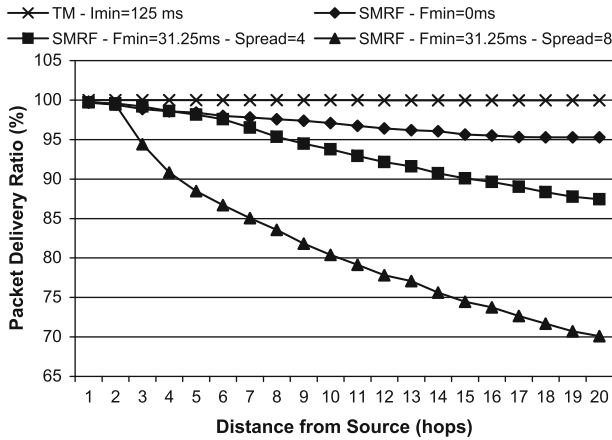
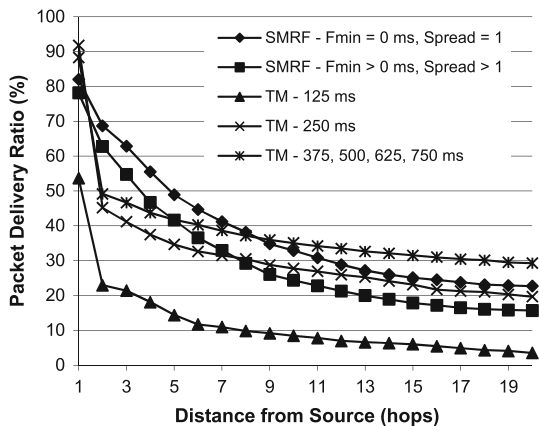


Fig. 5 Packet delivery ratio over NullRDC (Line Topology)

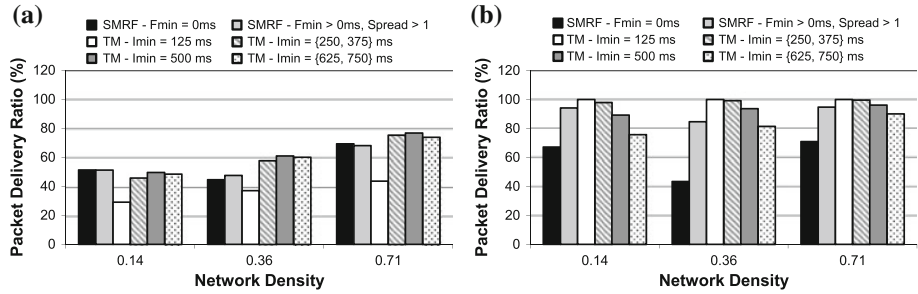
Fig. 6 Packet delivery ratio with ContikiMAC for both algorithms and various configurations (Line Topology)



multicast flow’s inter-datagram interval. The reason is that, even with constantly resetting trickle timers, ICMPv6 control packet exchange is not nearly quick enough to keep up with the multicast datagram arrival rate. Thus, the loss phenomenon described above is guaranteed to occur. Heavy packet losses mostly happen at the first hop, which acts as a *filter* for the remainder of the line: The first node throttles traffic to a rate which is more tenable for the rest of the network, which is why the decremental trend is a lot smoother beyond the first hop.

We cherry-picked TM’s *Imin* value of 125 ms, (which is the best choice for NullRDC), compared its packet delivery ratio with SMRF and plotted the results in Fig. 5. We include results with three *Fmin*, *Spread* configurations, each corresponding to a different line in the plot: *Diamonds*: (0, 1) respectively, *Squares*: (31.25, 4) and *Triangles*: (31.25, 8). Results for SMRF were comparable for different multicast traffic rates, we have thus combined all four rates in this single figure. Since SMRF only forwards each datagram once, it was anticipated to demonstrate a higher packet loss rate. Results confirm this and also indicate that losses increase as the forwarding delay increases.

We performed the same measurements over ContikiMAC and results are significantly different, as illustrated in Fig. 6. The first observation is that with ContikiMAC, packet loss rates are higher across the board. This is due to the fact that multicast datagrams are



**Fig. 7** Packet delivery ratio in the tree topologies for different algorithms, over both RDC layers and various traffic rates. **a** Over ContikiMAC. **b** Over NullRDC

transmitted as broadcast frames at the link layer, which is rather inefficient as discussed in Sect. 4.1. Another observation is that over ContikiMAC, lowering TM’s  $I_{min}$  has an adverse result: delivery ratio decreases instead of increasing.  $I_{min}$  values of 125 and 250 ms stand out as having underperformed, compared to the remaining four values which yielded comparable results and are thus averaged out into a single line. SMRF-related results are similar to those observed over NullRDC: (1) Packet loss is slightly higher compared to TM and (2) Increased forwarding delay (by increasing  $F_{min}$  and  $Spread$ ) results in higher losses.

### 5.2.2 Tree Topologies

In tree topologies, TM has a comparative advantage since there is path redundancy: a node may receive a multicast datagram from any of its potentially multiple neighbours. We thus anticipated its packet losses to be considerably lower. Multicast traffic transmission rates once again appeared to have minimal impact on the results and are thus averaged out. Based on our experience from the line topology, we singled out TM’s two best  $I_{min}$  candidates: 125 and 500 ms and compared results with SMRF’s two best  $F_{min}, Spread$  pairs: (0, 1) and (31.25, 4). Results are illustrated in Fig. 7, with the graph on the left corresponding to ContikiMAC and the one on the right to NullRDC.

When operating over NullRDC, the (0,1) SMRF configuration which performed very well in the line topology now severely under-performed its (31.25, 4) counterpart. The reason is that it is very susceptible to the hidden terminal problem, which was the original motivation behind introducing  $Spread$ . The results over ContikiMAC also confirm the finding that configuring TM with a sub-optimal  $I_{min}$  value can lead to very poor delivery ratios. Since packet trains indirectly mitigate the hidden terminal problem, the performance of SMRF’s (0, 1) configuration is comparable to the delivery ratio exhibited when  $Spread > 0$ . Depending on network density, SMRF can actually deliver more packets than TM, despite the fact that it is less reliable by design. This occurs when the latter is configured with a sub-optimal  $I_{min}$ .

## 5.3 End to End Delay

### 5.3.1 Line Topology

As discussed in Sect. 4, after our initial analysis we anticipated TM to exhibit low speed compared to the very straightforward SMRF. Figures 8 and 9 illustrate the results over NullRDC and ContikiMAC respectively.

For TM in the case of NullRDC we observe similar results as those for our analysis of Packet Delivery Ratio. Reducing  $I_{min}$  yields significant performance improvements: As  $I_{min}$



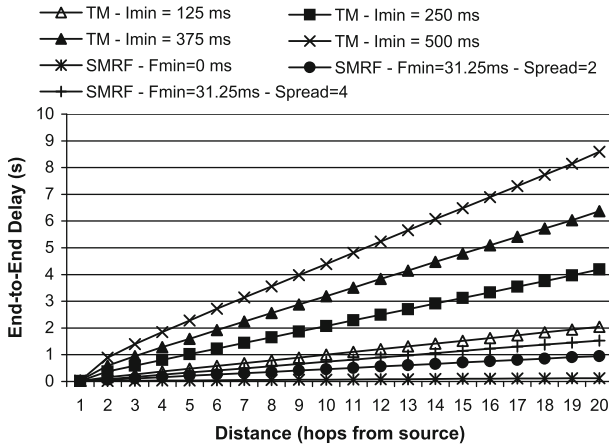


Fig. 8 End-to-end delay over NullRDC (Line Topology)

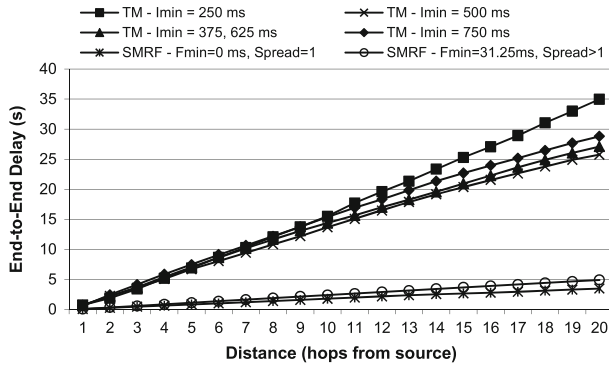
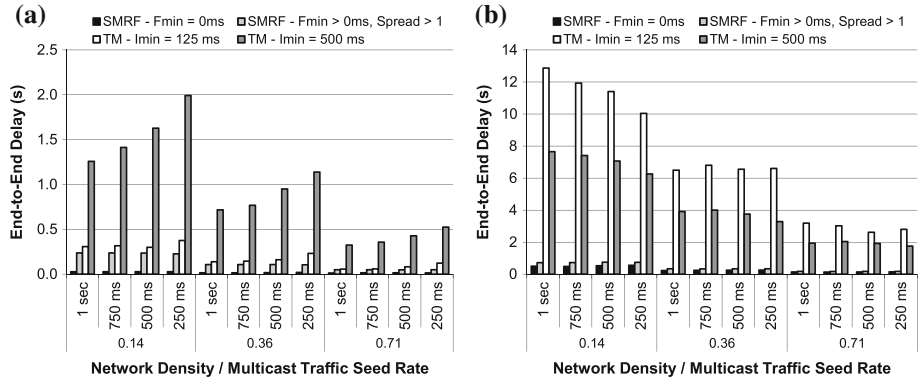


Fig. 9 End-to-end delay over ContikiMAC (Line Topology). Lines which would otherwise overlap are summarized into a single one (triangles and circles)

decreases, after resetting their trickle timers nodes exchange cache content information more frequently and inconsistencies are detected earlier, leading to significantly lower hop-by-hop forwarding delay. A similar observation applies to SMRF: Increasing the value of *Spread* effectively increases forwarding delay. The effect is augmented over twenty hops, leading to longer end to end delays. However, as anticipated by our discussion in Sect. 4, SMRF configurations with low *Spread* (2 and 4) are considerably faster than TM with *Imin* = 125 ms.

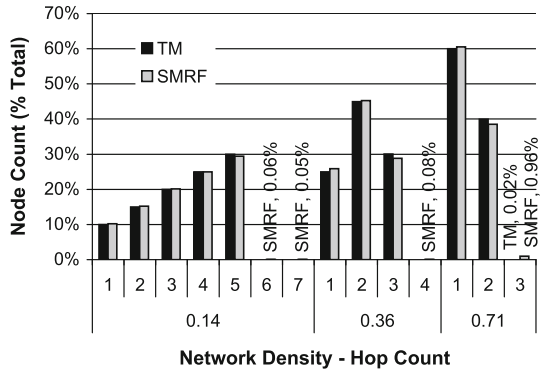
We approximated the three most representative lines from Fig. 8 (*stars*, *circles* and *white triangles*) with linear regression trend lines (not displayed in the diagram for clarity). All three trend lines demonstrated very good fit (co-efficient of determination was very high:  $R^2 > 0.998$ ). The slopes of the three lines were 0.006, 0.049 and 0.105 respectively. These three values represent an approximation of the estimated per hop forwarding delay. In other words, SMRF with *Fmin*, *Spread* (31.25, 2) is about 2.15 times faster per hop than TM with *Imin* = 125 ms.

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 500 ms. Any further reductions cause a radical performance degradation, with *Imin*=125 ms severely under performing. The difference between the two algorithms is even more extreme



**Fig. 10** End to end delay for different algorithm configurations, over both RDC layers. **a** Over NullRDC. **b** Over ContikiMAC

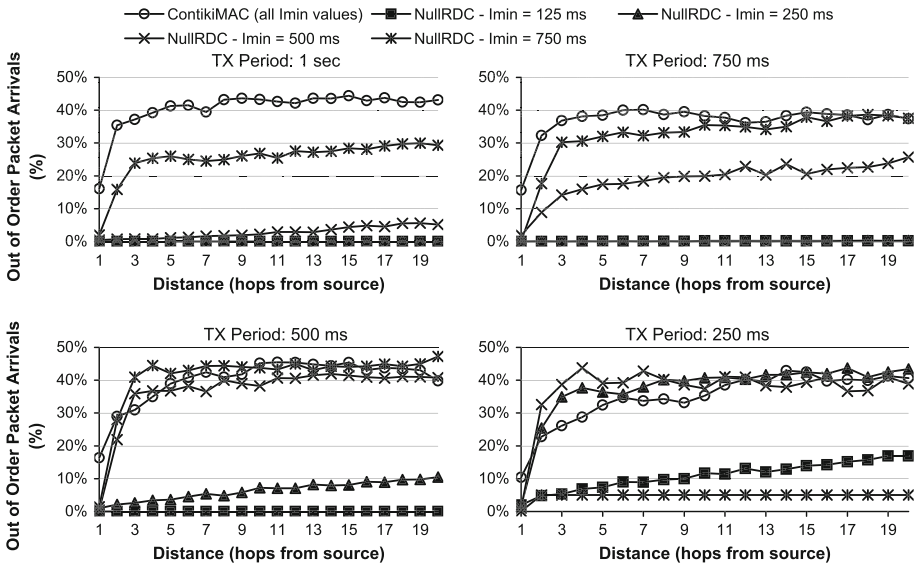
**Fig. 11** Hop count distribution for tree topologies



than in the case of NullRDC, as shown in Fig. 9, with SMRF being over 5 times faster than TM, on a hop-by-hop basis. This is caused by the fact that TM relies heavily on link-local multicast messages (link layer broadcasts) to exchange cache content information between nodes. As discussed in Sect. 4.1, link layer broadcasts are fundamentally inefficient with ContikiMAC.

5.3.2 Tree Topologies

We repeated the measurements in tree topologies, with results illustrated in Fig. 10. Network density is a very important factor for both algorithms. As network density increases, RPL’s DODAG tends to become *shallow* and *wide*: Maximum path length decreases, while the number of direct ancestors for a single node and the number of leaf nodes increase. As a result, with SMRF datagrams have to traverse fewer hops to reach their destinations. Similarly, TM has been designed to be density-aware [18]: High density and path redundancy result in fewer message exchanges before a datagram can reach all its intended recipients. In Fig. 11, the Y axis illustrates the number of nodes observed to be reachable after a number of hops, as a percentage of the total number of nodes across all experiments for this network density. Notice how, for both algorithms, the maximum observed hop count decreases as network density increases. The choice of duty cycling did not have significant impact so the figure combines results from both RDC algorithms.



**Fig. 12** Packets received Out-of-Order as a percentage of total packets received (Line Topology). Over ContikiMAC different *Imin* values have minor impact and are thus combined into a single line

Results regarding TM's *Imin* and end to end delay confirm the previous findings: Low values of *Imin* perform very well over NullRDC, while over ContikiMAC the optimal choice is a value of 500 ms. In all scenarios, irrespective of duty cycling and multicast traffic injection rate, SMRF outperforms TM significantly.

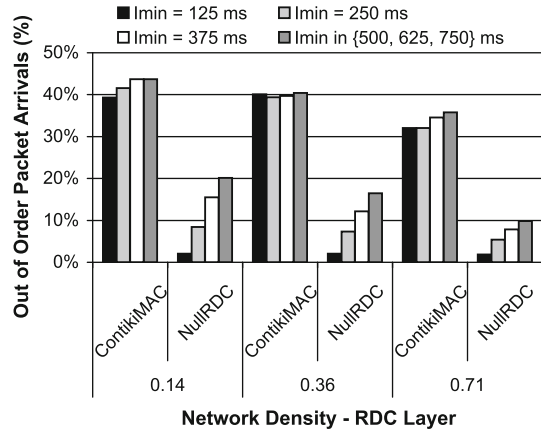
In the case of TM over ContikiMAC in a sparse topology, increasing traffic rate decreases end to end delay. This may seem confusing in the first instance but there is a very reasonable explanation: Increased traffic rate becomes the most frequent trigger for trickle timer resets, thus reducing per hop forwarding delay. In more dense topologies, maximum hop count is lower and mitigates this phenomenon.

### 5.4 Out-of-Order Arrivals

As discussed in Sect. 4, SMRF's operation does not entangle datagram ordering. In all our experiments, all messages delivered by SMRF were in the correct order. Thus, in this section we present evaluation results pertaining to TM only.

In Fig. 12 we present out of order datagram arrivals as a percentage of total transmitted datagrams, with each sub-figure corresponding to a different multicast traffic rate. Interestingly, in the case of ContikiMAC, changes in *Imin* do not appear to influence packet ordering significantly and are therefore combined into a single line. Beyond the fourth hop, out of order datagram delivery ratio fluctuates between about 35 and 45 % for all traffic rates. Compared to NullRDC, packet order is considerably worse with ContikiMAC due to higher error rate on a hop-by-hop basis. At each trickle timer reset, a node will copy and forward multiple datagrams in a row. However, the algorithm offers no guarantees that datagrams forming this *batch* will be transmitted in the correct order. Furthermore, some of them may get lost and get re-transmitted at a subsequent pass, thus arriving late. Conversely, with NullRDC, hop-by-hop transmission is more reliable and as a result TM delivers fewer datagrams out of order.

**Fig. 13** Out-of-order datagram delivery in tree topologies



For slow traffic rates (top two sub-figures) over NullRDC, low  $I_{min}$  values (125 or 250 ms) both achieve 0% out of order arrivals (*triangles* in the graph coincide with *squares*). This happens because the algorithm has a chance to copy and forward packets before the queue in each node's cache can build up. With a very good hop-by-hop successful transmission rate, need for re-transmissions is infrequent. As  $I_{min}$  increases, cache queues start building up and the batch phenomenon described above occurs, increasing out of order deliveries.

Experimentation in tree topologies (Fig. 13) confirms the findings: The percentage of datagrams getting delivered out-of-order is lower over NullRDC than over ContikiMAC, with low  $I_{min}$  values performing better than their high counterparts. Over ContikiMAC, the choice of  $I_{min}$  does not play a significant part, while the phenomenon is mitigated by increasing network density (as a result of the fact that each datagram has to cross fewer hops before reaching all intended recipients).

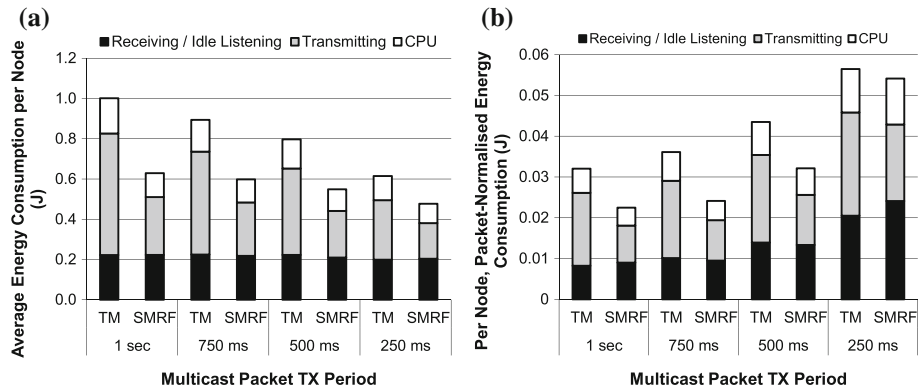
## 5.5 Energy Consumption

Through the facilities provided by Contiki's energy consumption estimation module (energest) [41,42], we measured the time each node spent in each of the following three states over the duration of each experiment: (1) MCU active, (2) RF listening / receiving, (3) RF transmitting. Since we are simulating sky motes, we then converted these time values to estimated energy consumption based on typical datasheet power levels at an operating voltage of 3.0 V [43].

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 other components contributing insignificantly. For this reason, we only consider ContikiMAC for the evaluation of the two algorithms in terms of energy consumption.

### 5.5.1 Line Topology

Figure 14a illustrates average (per-node) energy consumption for each of the two algorithms under different multicast traffic rates. Observe how consumption attributed to radio reception remains relatively constant across different experiments. The reason is that Radio



**Fig. 14** Energy consumptions over ContikiMAC for both algorithms in the line topology. **a** Average per node. **b** Per node average, normalised for the number of received packets

on/off cycles are controlled by the duty cycling algorithm and are thus unrelated to the behaviour of upper layers in the stack. 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 yet another indication of the algorithm’s increased complexity compared to SMRF.

Values displayed in this figure are averages across all nodes in the network. On a hop-by-hop basis, total energy consumption decreases linearly with distance from the traffic source, with nodes close to the source demonstrating an approximate 20% higher consumption than the average, while nodes near the end of the line consume as little as 20% less than the displayed average.

Contrary to our anticipation, network-wide energy consumption decreases as the multicast traffic rate increases, a side-effect of increased packet loss rate. Figure 14b depicts the same results normalised by the number of received datagrams for each scenario. In this context, the line drawing plots average energy consumption per successfully delivered datagram. This normalisation brings out the anticipated (incremental) trend with increasing multicast traffic rate.

### 5.5.2 Tree Topologies

Previous conclusions are confirmed under tree topologies, with results illustrated in Fig. 15. Average, per-node energy consumption is lower with SMRF, with radio transmissions being the most significant factor in the case of TM. Consumption due to listening and reception fluctuates slightly more than in the line topology, while network density has a positive effect in total consumption. The effect of network density is more significant in the case of TM, which was anticipated since the algorithm is *density-aware* by design [18].

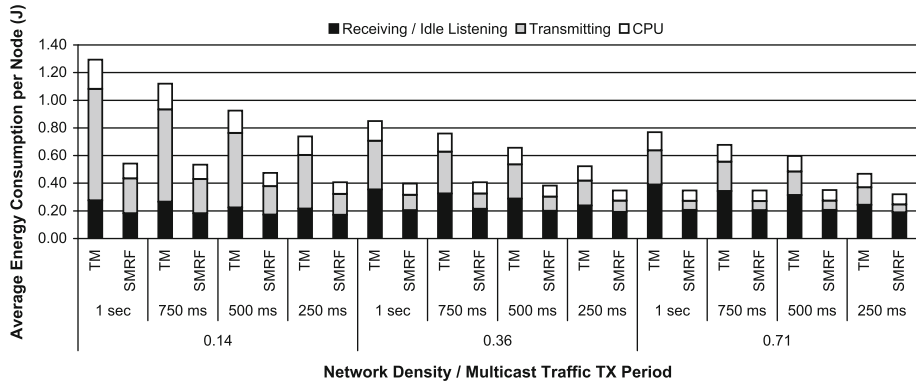


Fig. 15 Energy consumption for various network densities (tree topologies)

Table 3 Hardware testbed configuration

Nodes	11 Sensinode N740 NanoSensors
PHY and MAC	IEEE 802.15.4 with CSMA
Duty cycling	NullRDC
Iterations	10 for each parameter permutation
Message size	4 app. layer bytes (variable size on link)
Traffic interval	$\in \{62.5, 125, 250, 500, 1,000\}$ ms
TM params	$(I_{min}, I_{max}) \in \{(125\text{ ms}, 11), (500\text{ ms}, 9)\}$ $k = 3$
SMRF Params	$F_{min} = 0\text{ ms}, Spread = 1$ $F_{min} = 31.25\text{ ms}, Spread = 4$

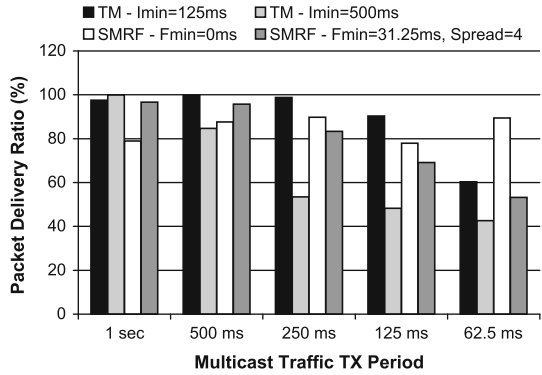
## 6 Hardware Experiments

In order to evaluate the validity of simulated results, we conducted similar measurements on a hardware testbed formed by 11 Sensinode N740 NanoSensors, each one equipped with an IEEE 802.15.4 RF transceiver. Similar to simulated experiments, one of the nodes acted as multicast traffic source and RPL root, while the remaining nodes would join the same multicast group and act as multicast forwarders and traffic sinks. Our nodes were installed in two different rooms, with enough distance and physical obstacles (full height concrete walls and closed wooden/glass doors) to guarantee a multi-hop topology at link-layer. Thus, the traffic source could only communicate with 5 of the 10 participating nodes.

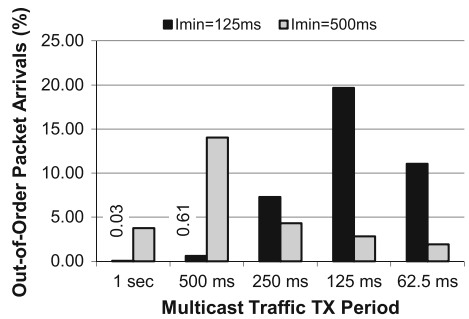
Under the assumption that radio links are symmetric, the network density of this deployment is  $ND = 0.72 (|E| = 40, |V| = 10$  in Eq. 1, Sect. 5). We used devices of identical specification, all transmitting with the same power and configured with the same receiver sensitivity level, thus the assumption is not unrealistic. Deviations can still be caused by spatial interference and by differences between devices introduced during manufacturing.

From a network layer perspective, the topology would alter from time to time and from experiment to experiment due to decisions made by the RPL routing protocol. On most occasions, all nodes were within up to three hops away from the source, with distances of four or five hops observed on very few occasions.

**Fig. 16** Datagram delivery ratio during hardware experiments



**Fig. 17** Out-of-order datagram delivery for TM during hardware experiments. With SMRF, all delivered datagrams are in the correct order



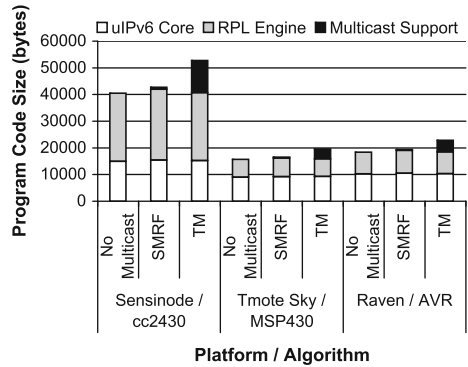
In this deployment, we evaluated packet delivery ratio and out of order packet arrival for the configurations listed in Table 3. The port of the Contiki OS for our hardware platform [19] does not support ContikiMAC yet. As a result all experiments were conducted over NullRDC.

In Fig. 16 we illustrate average datagram delivery ratio for both algorithms under different configurations and traffic rates, with results being a good match of simulated observations. For each experiment permutation, averages were calculated over all nodes and all iterations. Configuring TM with  $I_{min} = 500$  ms severely under performs all other scenarios and exhibits heavy packet losses under most traffic rates. As was the case during simulations, TM with  $I_{min} = 125$  ms exhibits higher delivery ratios than SMRF in most cases.

The results are inverted when the traffic rate is very high (1 message per 62.5 ms). We observe that TM’s packet loss increases abruptly when traffic inter-packet interval is lower than the value of  $I_{min}$ . The reason for this behaviour is that TM’s trickle timers are not refreshing frequently enough. As a result, node caches can overflow before all messages have been forwarded, resulting in dropped datagrams.

As discussed in Sect. 3.2, TM can re-arrange packet order resulting in out of order datagram deliveries. In Fig. 17, we illustrate the percentage of datagrams delivered out of order by TM for each experimental configuration. SMRF delivered all packets in order during all hardware experiments and is thus not displayed in the figure. The most notable observation is that incorrect packet order frequency peaks when the traffic rate is equal to  $I_{min}$ , with observed frequencies on either side of that value being significantly lower.

**Fig. 18** Code sizes for both algorithms and different hardware platforms in a typical configuration with support for 1 multicast group and TM cache size of 6 messages



## 7 Code Sizes and Memory Requirements

For both algorithms, we evaluated code size when compiled for three different hardware platforms: (1) Sensinode N740 NanoSensors (cc2430 System-on-Chip [44], 8bit 8051-based MCU, 8 MB RAM) with the Small Device C Compiler (SDCC), (2) Tmote Sky (16bit MSP430 MCU, 10 MB RAM) with the msp430-gcc toolchain, (3) 8bit AVR MCU (avr-gcc).

Figure 18 illustrates Contiki image sizes when configured to use each of the two algorithms and when built without multicast support. In both our implementations, changes to the TCP/IP core are minimal and this is reflected in the diagram for all platforms (white bar portions). In order for SMRF to operate, it requires RPL group management, as specified in the RPL RFC. Implementing this functionality for Contiki results in an increase of the RPL engine code size. This is only necessary for correct SMRF operation and is automatically disabled when building with TM support.

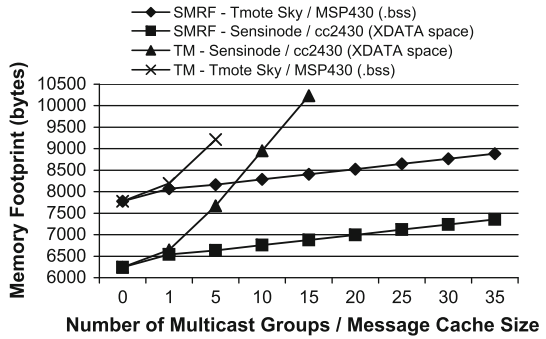
As discussed in Sect. 3.2, TM's design is of high complexity compared to SMRF. Each of the two algorithms is implemented as a separate code module, with respective code sizes depicted with the label *Multicast Support* (bars with black fill in the chart). The size of the module implementing TM is about ten times larger for all platform/compiler combinations. For instance, when building with SDCC, TM's code size is 12028 bytes whereas SMRF's module only occupies 718 bytes of code. With msp430-gcc, the respective sizes are 3,676 and 326 bytes in the .data segment.

We also investigate algorithm memory requirements and scalability with their operational parameters. For TM, we evaluate scalability with message cache size and for SMRF we research into the impact of an increase in the number of supported multicast groups. For SMRF, each multicast group requires additional storage space in the node's routing table. The memory requirement increase incurred by each additional group is reflected in Fig. 19 for two different hardware platforms (lines with diamonds and squares). The increase equals 24 bytes per group for both platforms. The vertical offset between the two lines is caused by two factors: (1) platform differences in terms of low level drivers and configuration and (2) different hardware architectures and compiler optimization capabilities.

The same figure illustrates TM's memory requirements as message cache size increases. We observe that the increase is very abrupt, with each additional message requiring an additional 256 bytes of RAM. This amount is necessary in order to cache the entire message headers and payload, alongside local control information required by the algorithm. Without altering Contiki's configuration for the Sensinode platform, images with TM support will only fit



**Fig. 19** Memory requirements and scalability



**Table 4** Algorithm selection criteria and optimal configuration

Criterion	TM	SMRF
Reliability	✓	
Low delay		✓
Code size	High	Low
Delivery order		✓
Energy efficiency		✓
<i>Configuration</i>		
NullRDC	Low $I_{min}$	$F_{min} = 0$ ms (lower delay)
(no duty cycling)	e.g. $I_{min} = 125$ ms	$F_{min} > 0$ ms (lower loss)
ContikiMAC	$3 * CCI \leq I_{min} \leq 4 * CCI$	$F_{min} > 0$ ms
(duty cycling)	e.g. $(CCI, I_{min}) = (125, 500)$ ms	

available memory when cache size is lower than 7 messages. For the Sky platform, memory restrictions are satisfied when the cache is configured to hold fewer than 9 messages.

### 8 Conclusions

In this work we have disclosed design and implementation details of the stateless multicast RPL forwarding (SMRF) algorithm for IPv6-based wireless sensor networks. We have also presented the outcomes of an in-depth comparison between SMRF and the Multicast Forwarding Using Trickle (TM) algorithm. We presented evaluation results obtained by simulations as well as from experiments conducted on a multi-hop hardware testbed. The latter result set validated simulated findings.

We have demonstrated that TM’s performance and energy consumption are very sensitive to changes in the value of its configuration parameter  $I_{min}$ , with the optimal depending on the choice of underlying duty cycling algorithm. In the case of deployments without duty cycling, decreasing the value of  $I_{min}$  results in better performance. In the more realistic scenario of duty-cycled networks we observe that, in order to minimize delay and packet loss  $I_{min}$  should be configured to a value higher than the CCI. Our experiments were conducted with a CCI value of 125 ms, in which case  $I_{min} = 500$  ms is the optimal configuration, performing slightly better than when  $I_{min} = 375$  ms. Lower values have a negative impact on packet loss, while higher values increase the ratio of packets delivered out of order. From

our experiments we also observed that the values of  $Imax$  and  $k$  do not have a significant impact on performance.

On the other hand, SMRF is less susceptible to variances of this nature, has lower end-to-end delay and is more energy efficient in exchange for an occasional slight drop in reliability. SMRF is also a lot less complex and has lower memory requirements, making it suitable for severely constrained devices.

Ultimately, the choice of multicast forwarding algorithm should be based on the anticipated usage of a sensor deployment. In Table 4 we highlight a summary of selection criteria and recommended configuration values for both algorithms.

The sources for both implementations are available on github, as a fork of Contiki's source tree<sup>5</sup> and are distributed under the terms of the 3-clause BSD license.

## References

- Hui, J. W., & Culler, D. E. (2008). IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys'08)* (pp. 15–28).
- Montenegro, G., Kushalnagar, N., Hui, J. W., & Culler, D. E. (2007). Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944.
- Hui (editor), J., & Thubert, P. (2011). Compression format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282.
- Winter (editor), T., Thubert (editor), P., Brandt, A., Hui, J., Kelsey, R., Levis, P., et al. (2012). RPL: IPv6 Routing Protocol for Low power and Lossy Networks. RFC 6550.
- Butt, T. A., Phillips, I., Guan, L., & Oikonomou, G. (2012). TRENDY: An adaptive and context-aware service discovery protocol for 6LoWPANs. In *Proceedings of the third international workshop on the web of things (WoT 2012)*. Newcastle, UK.
- Klauck, R., & Kirsche, M. (2012). Bonjour contiki: A case study of a DNS-based discovery service for the internet of things. In *Proceedings of the 11th international IEEE conference on ad-hoc networks and wireless (ADHOC-NOW 2012) Lecture Notes in Computer Science (LNCS)* (Vol. 7363, pp. 316–329). Berlin: Springer.
- Cheshire, S., & Krochmal, M. (2013). Multicast DNS. RFC 6762.
- Lynn, K., & Sturek, D. (2012). Extended multicast dns. Internet Draft (version 01). (draft-lynn-homenet-site-mdns-01).
- Cheshire, S., & Krochmal, M. (2013). DNS-based service discovery. RFC 6763.
- Klauck, R., & Kirsche, M. (2013). Enhanced DNS message compression—optimizing mDNS/DNS-SD for the use in 6LoWPANs. In *Proceedings of the 9th international workshop on sensor networks and systems for pervasive computing (PerSeNS 2013)*.
- Shelby, Z., Hartke, K., & Bormann, C. (2013). Constrained application protocol (CoAP). Internet Draft. (draft-ietf-core-coap-15).
- Sanchez, J. A., Ruiz, P. M., Liu, J., & Stojmenovic, I. (2007). Bandwidth-efficient geographic multicast routing protocol for wireless sensor networks. *IEEE Sensors Journal*, 7(5), 627–636.
- Sanchez, J. A., Marin-Perez, R., & Ruiz, P. M. (2012). Beacon-less geographic multicast routing in a real-world wireless sensor network testbed. *Wireless Networks*, 18(5), 565–578.
- Carzaniga, A., Khazaei, K., & Kuhn, F. (2012). Oblivious low-congestion multicast routing in wireless networks. In *Proceedings of the thirteenth international symposium on mobile ad hoc networking and computing (MobiHoc 2012)*.
- Feng, C. H., Zhang, Y., Demirkol, I., & Heinzelman, W. B. (2012). Stateless multicast protocol for ad hoc networks. *IEEE Transactions on Mobile Computing*, 11(2), 240–253.
- Koutsonikolas, D., Das, S. M., Hu, Y. C., & Stojmenovic, I. (2010). Hierarchical geographic multicast routing for wireless sensor networks. *Wireless Networks*, 16(2), 449–466.
- Song, S., Choi, B. Y., Kim, D. (2010). MR. BIN: Multicast routing with branch information nodes for wireless sensor networks. In *Proceedings of the 19th international conference on computer communications and networks (ICCCN 2010)* (pp. 1–6).
- Hui, J., & Kelsey, R. (2012). Multicast forwarding using trickle. Internet Draft (version 01). (draft-ietf-roll-trickle-mcast-01).

<sup>5</sup> <https://github.com/g-oikonomou/contiki-sensinode/tree/mcast-forward>.

19. Oikonomou, G., & Phillips, I. (2011). Experiences from porting the contiki operating system to a popular hardware platform. In *Proceedings of the 2011 international conference on distributed computing in sensor systems and workshops (DCOSS)*. Spain: Barcelona.
20. Oikonomou, G., & Phillips, I. (2012). Stateless multicast forwarding with RPL in 6LoWPAN sensor networks. In *Proceedings of the 2012 IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops)*. Switzerland: Lugano.
21. Das, S. M., Pucha, H., & Hu, Y. C. (2008). Distributed hashing for scalable multicast in wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(3), 347–362.
22. Song, S., Kim, D., & Choi, B. Y. (2009). Agsmr: Adaptive geo-source multicast routing for wireless sensor networks. In *Proceedings of the wireless algorithms, systems, and applications, Lecture notes in computer science* (Vol. 5682, pp. 200–209). Springer:Berlin/Heidelberg.
23. Okura, A., Ihara, T., & Miura, A. (2005). Bam: branch aggregation multicast for wireless sensor networks. In: *IEEE international conference on mobile ad hoc and sensor systems conference* (pp. 354–363).
24. Flury, R., & Wattenhofer, R. (2006). MLS: An efficient location service for mobile ad hoc networks. In *Proceedings of the 7th ACM international symposium on mobile ad hoc networking and computing (MOBIHOC)* (pp. 226–237).
25. Li, J., Jannotti, J., De Couto, D. S. J., Karger, D. R., & Morris, R. (2000). A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th annual international conference on mobile computing and networking, MobiCom'00* (pp. 120–130).
26. Sá, Silva J., Camilo, T., Pinto, P., Ruivo, R., Rodrigues, A., Gaudêncio, F., et al. (2008). Multicast and ip multicast support in wireless sensor networks. *Journal of Networks*, 3(3), 19–26.
27. Royer, E. M., & Perkins, C. E. (1999). Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom'99)* (pp. 207–218).
28. Bhattacharyya, S. (Ed.) (2003). *An overview of source-specific multicast (SSM)*. RFC 3569.
29. Clausen, T., & Herberg, U. (2010). Comparative study of RPL-enabled optimized broadcast in wireless sensor networks. In *Proceedings of the sixth international conference on intelligent sensors, sensor networks and information processing (ISSNIP 2010)*. Brisbane, Australia: IEEE.
30. Levis, P., Patel, N., Culler, D., & Shenker, S. (2004). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the first USENIX/ACM symposium on networked systems design and implementation (NSDI)* (pp. 15–28).
31. Levis, P., Clausen, T.H., Hui, J., Gnawali, O., & Ko, J. (2011). The trickle algorithm. RFC 6206.
32. Levis, P., Brewer, E., Culler, D., Gay, D., Madden, S., Patel, N., et al. (2008). The emergence of a networking primitive in wireless sensor networks. *Communications of the ACM*, 51(7), 99–106.
33. Hui, J., & Culler, D. (2004). The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)* (pp. 81–94).
34. Lin, K., & Levis, P. (2008). Data discovery and dissemination with DIP. In *Proceedings of 7th international conference on information processing in sensor networks, IPSN'08* (pp. 433–444). Washington, DC, USA: IEEE.
35. Hui, J., & Vasseur, J. P. (2012). The routing protocol for low-power and lossy networks (RPL) option for carrying RPL information in data-plane datagrams. RFC 6553.
36. Dunkels, A. (2011). *The ContikiMAC radio duty cycling protocol*. Technical Report T2011:13, Swedish Institute of Computer Science.
37. Dunkels, A., Mottola, L., Tsiftes, N., Österlind, F., Eriksson, J., & Finne, N. (2011). The announcement layer: Beacon coordination for the sensornet stack. In *Proceedings of the European conference on wireless sensor networks (EWSN)*.
38. Österlind, F., & Dunkels, A. (2008). Approaching the maximum 802.15.4 multi-hop throughput. In *Proceedings of the fifth ACM workshop on embedded networked sensors (HotEmNets 2008)*.
39. Duquennoy, S., Österlind, F., & Dunkels, A. (2011). Lossy links, low power, high throughput. In *Proceedings of the 9th ACM conference on embedded networked sensor systems (SenSys 2011)*.
40. Michopoulos, V., Guan, L., Oikonomou, G., & Phillips, I. (2012). DCCC6: Duty cycle-aware congestion control for 6LoWPAN networks. In *Proceedings of the 2012 IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops)*. Switzerland: Lugano.
41. Dunkels, A., Österlind, F., Tsiftes, N., & He, Z. (2007). Demo abstract: Software-based sensor node energy estimation. In *Proceedings of the fifth ACM conference on networked embedded sensor systems (SenSys 2007)*. Sydney, Australia: ACM.
42. Dunkels, A., Österlind, F., Tsiftes, N., & He, Z. (2007). Software-based on-line energy estimation for sensor nodes. In *Proceedings of the fourth workshop on embedded networked sensors (Emnets IV)*. Cork, Ireland: ACM.

43. Tmote sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module. Moteiv Corporation (2006). Tmote Sky Datasheet Revision 1.0.2.
44. A True System on Chip solution for 2.4 GHz IEEE 802.15.4/Zigbee®. (2007). CC2430 Data Sheet (rev. 2.1).

## Author Biographies



**George Oikonomou** received the M.Sc. and Ph.D. degrees in computer science from the Athens University of Economics and Business, Athens, Greece, in 2002 and 2009 respectively. He is currently a Research Associate with the Queen's School of Engineering, University of Bristol, UK. Previously, he worked as a Research Associate at the Computer Science department, Loughborough University, UK. His current research focuses on wireless sensor networks and the Internet of things, with emphasis on IPv6 networking for low-power, severely constrained devices. Dr Oikonomou is an active developer of the Contiki open source embedded operating system for the internet of things.



**Iain Phillips** has been involved in computing research for over 20 years, with over 100 publications in this area. His main work has been in network architectures considering performance and algorithms for the Internet and Wireless Sensor Networks. He graduated from Manchester University with a BSc in Computing and Information Systems and a Ph.D. in Computer Science—Thesis title Workload Distribution on Massively Parallel Machines. After 7 years post-doctoral research at Loughborough in the High Speed Networks group in Electrical Engineering (Research Assistant/Research Fellow), he moved to Loughborough Computer Science as a Lecturer in 1999, Senior Lecturer in 2005 and Head of Department 2008–11. From 2009–11 he was Vice Chair of the Council of Professors and Heads of Computing (CPHC) and from 2011 Chair. He is a Chartered Fellow of the BCS and a member of the ACM.



**Theo Tryfonas** is a senior lecturer in systems engineering at the University of Bristol. His work is in the areas of software engineering, network security and forensic investigations, systems architecture as well as risk and resilience of systems across a number of sectors including telecommunications, transport and healthcare. His current research interests include looking into uses of wireless and mobile technologies for improving sustainability of future cities, developing policies and frameworks for delivering smart transport and game-theoretic approaches to understanding end-user behaviour implications of smart urban technologies. He is a Chartered IT Professional member of the BCS and a Certified Information Systems Auditor.