Performance Evaluation of Reliable Overlay Multicast in Wireless Sensor Networks

Gerald Wagenknecht, Markus Anwander, and Torsten Braun

Institute of Computer Science and Applied Mathematics
Universität Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
wagen@iam.unibe.ch, anwander@iam.unibe.ch, braun@iam.unibe.ch

Abstract. Using multicast communication in Wireless Sensor Networks (WSNs) is an efficient way to disseminate code updates to multiple sensor nodes. For this purpose a multicast protocol has to support bulky traffic (typical traffic pattern for code updates) and end-to-end reliability. In addition, we are interested in energy-efficient operations due to the limited resources of WSNs. Currently no data dissemination scheme fits the requirements mentioned above. Therefore, we proposed the SNOMC (Sensor Node Overlay Multicast) protocol, an overlay multicast protocol, which supports reliable, time-efficient, and energy-efficient data dissemination of bulky data from one sender to many receivers. The protocol's performance in terms of transmission time, number of totally transmitted packets and energy consumption is compared to other often cited data dissemination protocols. Our results show superior performance of SNOMC independent of the underlaying MAC protocol.

1 Introduction

Wireless Sensor Networks (WSN) consist of wireless sensor nodes, which host different applications for the purposes of event detection, localization, tracking, monitoring. An application needs to be configured and continuously updated throughout the lifetime of the network. Such tasks can occur rather often, especially in the deployment phase. There are several challenges to that the configuration and update process. Configuration and updating should be done over the air [1]. Moreover, code update traffic is bulky in nature and has high reliability requirements. Finally, in contrast to the predominant multipoint-to-point communication in WSNs (data retrieval), code updates follow a point-to-multipoint pattern.

Several strategies can be used for data delivery from one sender node to many receivers. The simplest strategy is flooding, where data is transmitted using broadcast communication mechanisms. It is, however, inherently very inefficient, energy consuming, and unreliable. Current broadcast mechanisms for code dissemination such as TinyCubus [2], Deluge [3], and Trickle [4] unfortunately do not include any reliability mechanism. Another strategy is to deploy multiple unicast connections between the sender and any of the desired receivers. In the context of WSNs, however, redundant transmissions lead to higher probability of collisions and increased transmission times. A distribution strategy that can more efficiently meet the requirements of configuration and code updating, is multicast. Multicast is able to propagate data from a single sender

to many receivers by affecting a smaller number of sensor nodes in the network. It is easily extendable with any kind of reliability mechanism.

Simply porting an existing IP Multicast solution designed for wired networks to wireless sensor networks is impractical or even impossible. There are three main challenges to that. In contrast to wired networks, resources such as energy, memory, and CPU power are limited in WSNs. Therefore, directly porting an existing IP Multicast solution would require memory and processing power, which a sensor node may lack. Even if resources are available, the lifetime of a node may be severely affected. While in wired networks each node has a dedicated role, nodes in WSNs can take the roles of sender, receiver, forwarding node, and branching node, see Fig. 1. Branching nodes duplicate packets and store state information about receivers and/or about other branching nodes. Forwarding nodes have less or no information about the multicast state and just forward the multicast data from one neighbor to the next one. Wireless communication links are more vulnerable to disruptions than wired links, which raises additional concerns about medium availability, collisions, and reliability.

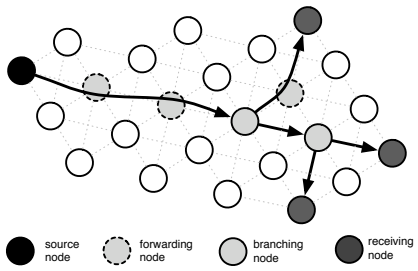


Fig. 1. Roles of the nodes in a multicast scenario.

A multicast solution for WSNs needs to address the above-mentioned issues and in particular, reliability to ensure that code updates are disseminated efficiently. Despite many studies on multicast in WSNs most of them focus on multicast routing and not on reliable and efficient data distribution (see Section 2 for more details). Up to our knowledge, there is not a single multicast protocol able to meet the combined set of requirements for reliability and efficiency in both time and energy consumption for bulky traffic patterns. Moreover, we would like the multicast communication to be IP-based in order to access the WSN via the Internet [5].

To fill in the gap we proposed the SNOMC (Sensor Node Overlay Multicast) protocol [6], which supports the reliable transfer of bulk data in a WSN from one sender to multiple receivers. SNOMC has been designed as an overlay multicast protocol on top of the μIP stack from Contiki [7] and offers time- and energy-efficient data distribution and simple NACK-based mechanism for end-to-end reliability. A complete protocol description is provided in [6] while in this paper we are more interested on the protocol performance and how it compares to other proposed solutions.

The paper is organized as follows: Section 2 introduces related work on data distribution schemes as well as on multicast in WSNs. Section 3 describes the SNOMC

protocol briefly. Evaluation, including simulation scenario, used protocol stack, and results is presented in Section 4. Section 5 concludes the paper.

2 Related Work

A commonly used data dissemination scheme of low complexity but low efficiency in WSNs is broadcasting. A number of protocols have been proposed to improve the efficiency of broadcasting such as Multipoint Relaying (MPR) [8]. Only a subset of nodes (so called multipoint relays) rebroadcast messages. The relays are chosen based on local knowledge at each node of its two-hop neighborhood. MPR does not support any reliability mechanism. Pump Slowly, Fetch Quickly (PSFQ) [9] is a reliable transport protocol and supports broadcast-based code distribution. It transmits data segments relatively slowly ('pump slowly') and uses an aggressive NACK mechanism to fetch missed data segments ('fetch quickly'). The aggressive NACK mechanism can lead to congestion in the WSN. TinyCubus [2] is an adaptive cross-layer framework for wireless sensor networks. It is also broadcast-based and deploys a role-based code distribution algorithm using cross-layer information such as role assignments to decrease the number of messages needed for code distribution to specific nodes. TinyCubus assumes that roles are assigned before code deployment. Directed Diffusion [10] can be used for both multipoint-to-point and point-to-multipoint communications.

In [11] a multicast protocol called BAM (Branch Aggregation Multicast) is presented. It supports single-hop link-layer multicast and multi-hop multicast by doing branch aggregation. Another multicast protocol for sensor nodes with of node mobility support is VLM^2 (Very Lightweight Mobile Multicast) [12]. VLM^2 provides multicast from a base station to sensor nodes and unicast from sensor nodes to a base station, but it has no reliability mechanisms. In [13] the authors present an effective all-in-one solution for unicasting, anycasting, and multicasting in wireless sensor and mesh networks. RBMulticast [14] is a stateless, receiver-based multicast protocol, which exploits knowledge on geographic node locations to reduce costly state maintenance. The authors of [15] adapt ADMR (Adaptive Demand-driven Multicast Routing), a multicast protocol for mobile ad-hoc networks, on a real wireless sensor node (MICAz). They show that protocol adaptation is not a trivial task and a number of problems have to be solved. At the same time, the authors of [16] analyze IP Multicast and show that it is possible to be used in WSNs. Further, there are several multicast solutions for WSNs based on the geographical sensor node positions [17, 18, 19]. All of these protocols support neither end-to-end reliability nor energy-saving mechanisms.

3 SNOMC Protocol Description

The main requirements towards the protocol we wish to design are multicast support in WSNs, reliable communication for bulky traffic (e.g. code updates), and protocol operation on top of IP. Multicast solutions for WSNs can be classified in different ways. First, depending on the layer of the protocol implementation, there are IP multicast (network layer) and overlay multicast (application layer) solutions. For IP multicast the distribution tree is built between routers in the Internet. For overlay multicast, the tree is

built between the end systems. Second, we can distinguish between a sender-driven and a receiver-driven formation of the multicast group. Moreover, different transport protocols (UDP or TCP) can be used, depending on requirements towards reliability support. Last, the network entity where caching occurs, can differ: sender nodes, branching nodes or all intermediate nodes. Detailed description can be found in [6], while here a shorter overview is presented.

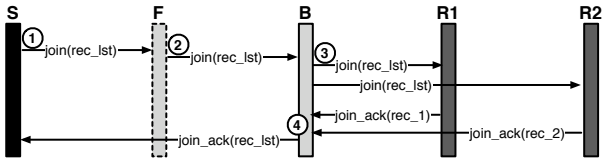


Fig. 2. SNOMC: Joining phase, sender-driven mode.

To meet our design requirements we developed the Sensor Node Overlay Multicast (SNOMC) protocol, an overlay multicast protocol able to operate in both sender-driven mode and receiver-driven mode. We are using UDP as transport protocol and to ensure reliability we are using a simple NACK-based mechanism with all three caching modes. In the sender-driven mode, the sender decides which nodes should be in the multicast group as receivers. The join procedure is shown in Fig. 2. First, the sender creates a *join* message, which contains the list of receivers, the group id, and the address of the sender. This *join* message is transmitted towards all receivers via intermediate nodes. If an intermediate node can reach all receivers via a single one-hop neighbor it is a forwarding node. Otherwise, the intermediate node becomes a branching node and splits the receivers list into partial lists for the respective next-hop neighbors. Each branching node adds its address into the *join* message. When a *join* message reaches a receiver it confirms its role as receiver by transmitting a *join_ack* message back to the last branching node. The branching node waits for the *join_ack* messages of all subordinate receivers, combines them, puts its address as branching node into the message and transmits it back towards the sender node. Thus, the sender node knows the next branching nodes and each branching node knows its predecessor and its successor.

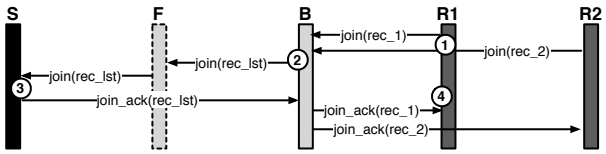


Fig. 3. SNOMC: Joining phase, receiver-driven mode.

In the receiver-driven mode, see Fig. 3, the receivers themselves decide whether they want to be a member of the multicast group by sending a *join* message to the

sender. A node on the path from receiver to sender decides on its role depending on the number of subordinated receivers, one or many respectively. Branching nodes need to notify the sender by adding their identity to the *join* message. The sender node responds with a *join_ack* message containing the receiver list, its own address as sender_id, and the collected branch_id. Branching nodes are responsible to split the *join_ack* message when necessary. Data from sender to receivers are propagated using the overlay network established as result of the join procedure. Overlay links are established between the nodes, which cache the data.

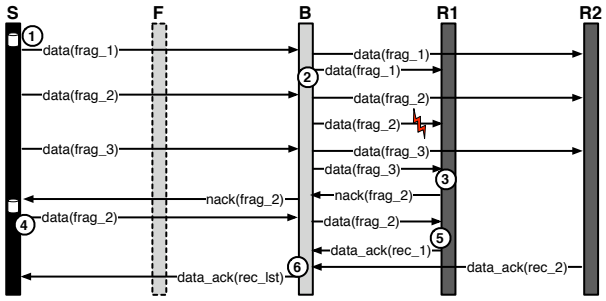


Fig. 4. SNOMC: Transmission phase, caching on sender node.

The caching strategy 'caching on sender node' is depicted in Fig. 4. The sender node fragments the data and caches them. Each branching node duplicates and retransmits *data* messages. If a fragment gets lost, the receiver detects a gap in the fragment sequence and requests the missing fragment. This is done using a *nack* message. Since the sender node has the fragment in the cache, it retransmits it towards the requesting receiver. If the receiver gets all fragments successfully, it confirms this with a *data_ack* message. Branching nodes can combine *nack* and *data_ack* messages. Another strategy is when data can be cached additionally at the branching nodes. The cache size is 10 packets. If more packets are coming in the oldest one will be deleted from the cache. The benefit of this additional caching is that a receiver can request data directly from the branching node if it has detected a lost fragment.

In case the data can be cached at every intermediate node the overlay connections change such that every node knows its predecessor and successor in the distribution tree. If a receiver detects a missing fragment, it requests the fragment directly from its predecessor node. If the predecessor node has the fragment cached it retransmits it; otherwise it forwards the *nack* message up the distribution tree until a node is found where the fragment has been cached.

4 SNOMC Evaluation

This section presents the evaluation of the SNOMC protocol. First, we describe the protocol stack. Then, we move on to introduce the different simulation scenarios. Finally, we discuss the results of our measurements.

4.1 Protocol Stack

To evaluate the performance of SNOMC, we compare it to a number of transport protocols commonly found in wireless sensor networks in combination with different underlying MAC protocols. More specifically, these protocols are: Flooding, Multipoint Relay, TinyCubus, Directed Diffusion, UDP, and TCP. For the description of the protocols we refer to Section 2. All protocols have been implemented in the OMNeT++ simulator [20]. The protocol stack is shown in Fig. 5 and is based on the μIP stack from Contiki. To enable a fair comparison we had to ensure end-to-end reliability for all protocols and implement the same simple NACK-based reliability mechanism used in SNOMC.

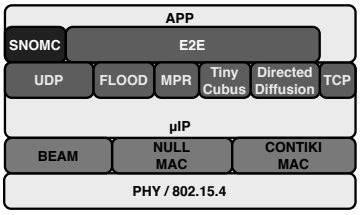


Fig. 5. Simulation protocol stack.

Finally, we compare SMOMC with two unicast-based transport protocols: UDP and TCP. While TCP has a reliability mechanism based on positive acknowledgments, we enhanced UDP with a NACK-based reliability mechanism same as in SNOMC.

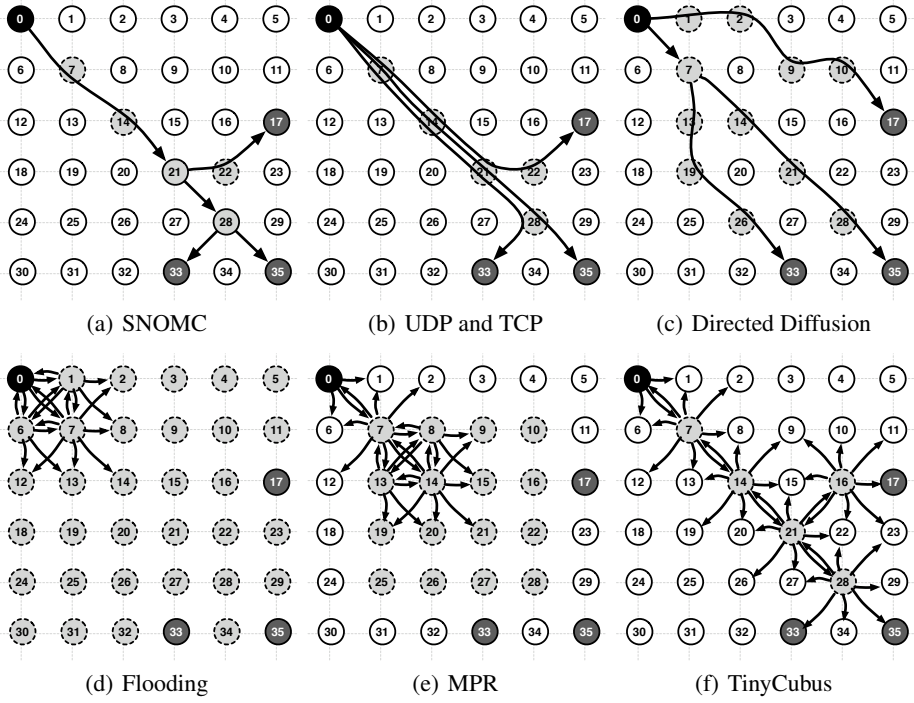
The underlying MAC protocol plays an important role for the transmission of data between neighbor nodes. Hence, different MAC protocols can lead to significantly different results, irrespectively of the used transport or multicast protocol. We chose three MAC protocols with different support mechanisms for reliability and energy-efficient operation. The Burst-aware Energy-Efficient Adaptive MAC Protocol (BEAM) [21] uses an adaptive duty cycle mechanism, which reacts quickly to changes in both traffic loads and traffic patterns and ensures hop-to-hop reliability. ContikiMAC [22], which is part of the Contiki operating system, also supports energy-saving radio duty cycling mechanisms and reliability based on an acknowledgment mechanism. NullMAC, also part of Contiki, has no energy-saving mechanisms and does not support reliability. Table 1 shows an overview of the parameter of the used MAC protocols.

Table 1. MAC Protocol Parameters.

	acknowledgments	retransmissions	energy-saving
BEAM	positive ack	5	yes
ContikiMAC	early ack	2	yes
NullMAC	no	0	no

4.2 Simulation Scenario

We arranged 36 sensor nodes in a grid of 6x6 nodes with a distance of 100 meters between nodes as shown in Fig. 6. Since we are interested in a multicast scenario, we chose for a sender (node 0) with three receivers (node 17, 33, and 35).



(a) SNOMC (b) UDP and TCP (c) Directed Diffusion

(d) Flooding (e) MPR (f) TinyCubus

Fig. 6. Simulation scenarios.

Given the chosen simulation scenario, each of the compared protocols affects a different set of nodes. In the case of SNOMC there are two branching nodes (21, 28) and three forwarding nodes (7, 14, 22) as shown in Fig. 6(a). For UDP and TCP the same nodes are affected but there are three independent connections (cf. 6(b)). As shown in Fig. 6(c) a different set of nodes participates in the distribution tree in Directed Diffusion, which is a result of the different interest message routing compared to the static routing of SNOMC. Flooding is a radical case where all nodes are affected (cf. in Fig. 6(d)). In the chosen grid scenario the Multipoint Relay protocol calculates a rather high number of multipoint relay nodes (cf. Fig. 6(e)). In the case of TinyCubus, the same set of nodes as in SNOMC is affected. Due to design decisions the protocol does not distinguish between receivers and intermediate forwarders (cf. Fig. 6(f)). Hence, all nodes in the set (7, 14, 21, 22, and 28) will rebroadcast the packets.

We created two evaluation scenarios, which differ in the size of the transmitted messages - 20 bytes and 1000 bytes. The size of 1000 bytes is typically associated with

software updates on the sensor nodes; the size of 20 bytes is related to a short configuration message for the sensor nodes. For each scenario, 50 simulation runs are used for evaluation. We measured three parameters: (i) transmission times from the sender to all receivers, (ii) the total number of packets it takes to ensure the successful reception of the data by all receivers, and (iii) the energy consumption of the nodes in the network. The energy consumption is measured according to the CC2420 state machine with real switching times and energy consumption according to [23] and [24] (values for sleeping, receiving, and transmitting) and is calculated per node and per transmitted byte. All parameters are measured only taking into account the data distribution phase. Any initial phases are not considered.

Table 2. Simulation Parameters.

carrierFrequency	bit-rate	sensitivity	thermalNoise	TX power	modulation
2.4E+9 Hz	250 kbps	-94 dBm	-110 dBm	1mW	O-QPSK

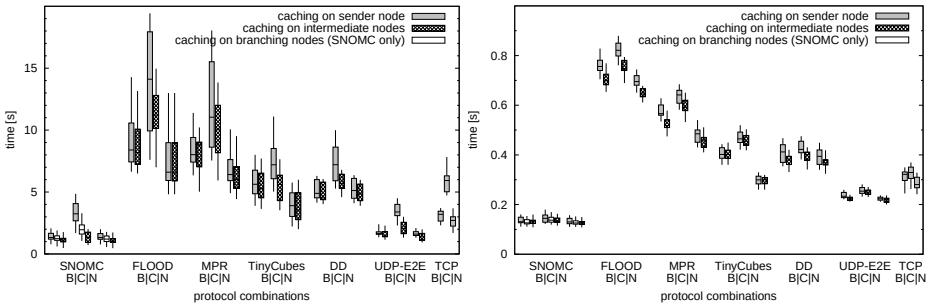
In order to get realistic results a radio model is implemented according to the CC2420 manual [23] and the Castalia Simulator [24]. It is used to calculate the signal to noise ratio (SNR) based on parameters shown in Table 2. Using the SNR and real measurements with a CC2420 radio transceiver the bit error rate (BER) is calculated. In addition, a normally distributed packet error rate of 5% is assumed to represent random noise and external interferences.

4.3 Results on Time Consumption

In this section, we present our findings transmission times. In all figures on the x-axis the combinations of transport and MAC protocols are shown. In our notation **B** stands for BEAM, **C** for ContikiMAC, and **N** for NullMAC.

First, in Fig. 7(a), we discuss results for the time required to transmit 1000 bytes from the sender node to the three receiver nodes. As expected, UDP-E2E require more time due to the redundant unicast flows that need to be established for each of the three receiver nodes. TCP performs even worse since every packet has to be acknowledged. This cross traffic caused by simultaneous data and acknowledgments increases collision probability and hence affects delay negatively. Flooding, Multipoint Relay, and TinyCubus are all broadcast protocols and are much worse in performance. On the one hand, broadcasting affects usually more nodes, which leads to a higher number of transmissions. Consequently, the probability of collisions increases and more retransmissions are necessary pushing delay up. On the other hand, to avoid collisions higher random back-off time, compared to unicast-based, are necessary. This, however, would also lead to longer transmission times. SNOMC requires the lowest time to transmit the 1000 bytes to the receivers.

If we now compare the performance in combination with the MAC protocol, we see in Fig. 7(a) that BEAM has a little lower performance, considering the time needed to



(a) 1000 bytes (b) 20 bytes

Fig. 7. Transmission time.

deliver 1000 bytes, than NullMAC using SNOMC, Directed Diffusion, UDP and TCP. However, BEAM outperforms ContikiMAC, which is the result of two factors. First, BEAM is optimized for bulky traffic while ContikiMAC focuses only on constant (or slowly changing) traffic. Second, BEAM has a better congestion control and better duty cycle mechanism. The latter is also the reason why caching at intermediate nodes affects the performance with both protocols differently, i.e., BEAM has much smaller effects. Together with broadcast-based protocols, NullMAC works better than BEAM. Since BEAM has energy-saving mechanisms, the radio transceiver can be in sleep mode. If so, longer time is needed to transmit a packet from sender to receiver. On the contrary, the radio transceiver in NullMAC is always on and therefore the sender can immediately transmit the packet.

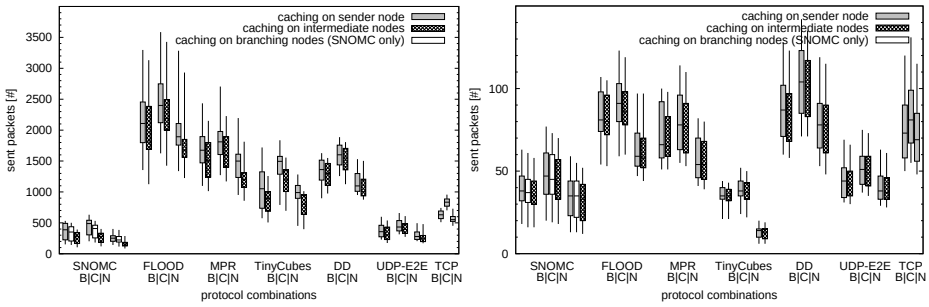
In case of 20 bytes of data a single packet has to be transmitted. Transmission times are shown in Fig. 7(b). We see that SNOMC achieves the best performance. In an ideal case TinyCubus would be better since it requires a smaller number of transmissions compared to SNOMC (due to using broadcast transmissions), but SNOMC has the added benefit of smaller random-back off times. UDP and TCP need more transmissions due to the three independent flows, hence the longer transmission times. Although TCP requires acknowledgments for each packet, in our scenario there will be a single acknowledgment only (due to only one data packet), explaining the much smaller differences between TCP and UDP compared to the scenario with 1000 bytes.

Further, in SNOMC, TinyCubus, Directed Diffusion and UDP collisions among data and acknowledgments generally do not occur and hence no retransmissions are required. Therefore, the corresponding boxplots in Fig. 7(b) are quite compact and have no big outliers. Finally, the differences between Flooding, Multipoint Relay and Directed Diffusion are similar to the 1000 bytes scenario.

4.4 Results on Number of Transmissions

Fig. 8(a) shows the number of total transmissions needed for the successful transfer of 1000 bytes. As we can see, SNOMC requires the fewest number of packets, followed by UDP, TCP and Directed Diffusion. The results of broadcast-based protocols (Flooding, Multipoint Relay, and TinyCubus) are considerably worse and are compliant with

our observations on transmission times. Flooding requires most transmissions (inherent to its communication style), followed by Multipoint relay (result of the disadvantageous set of multipoint relays) and, with the best performance of the three, TinyCubus. Looking at the MAC protocols, BEAM requires more packets to ensure hop-to-hop reliability than NullMAC, irrespectively of the transport protocol. This is due to the fact that the receiver can be in sleeping mode and multiple attempts may be required before the packet is transmitted successfully. Using ContikiMAC always needs more packet retransmissions on link layer due to a worse duty cycle mechanism and thus higher number of necessary end-to-end retransmissions on the transport layer.



(a) 1000 bytes (b) 20 bytes

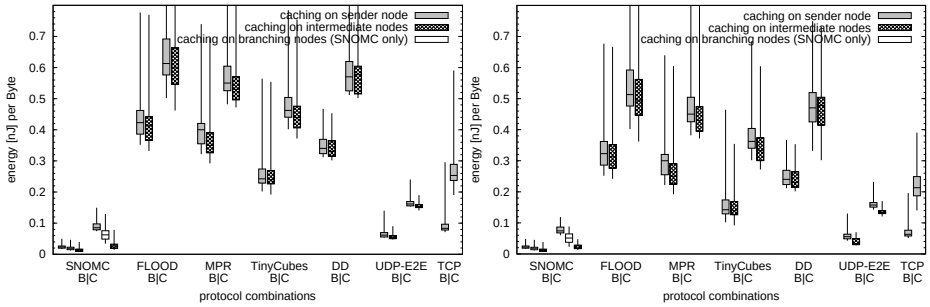
Fig. 8. Number of transmitted packets.

The results for the transmission of 20 bytes are shown in Fig. 8(b). TinyCubus achieves the best performance in combination with NullMAC. It has an optimal set of forwarding nodes and NullMAC keeps the radio transceiver always awake. Hence, this combination reaches the minimal number of packets (6) to ensure the successful transmission of 20 bytes. The other broadcast protocols (Flooding and Multipoint Relay) show an improved performance as well, considering the scenario with 1000 bytes. Just one packet has to be transmitted, which causes less collisions and retransmissions. Out of all non-broadcast-based protocols, SNOMC has the best performance while Directed Diffusion has the worst. Directed Diffusion requires a larger number of packets because of the more extensive hop connectivity, i.e., more connections compared to UDP and TCP. Further, the differences between the three caching modes are quite small; a result of the smaller number of required retransmissions.

4.5 Results on Energy Consumption

We now move on to discuss the energy consumed per node and per transmitted byte. Results for the scenario with 1000 bytes are shown in Fig. 9(a). We compare only the performance with BEAM and ContikiMAC, since NullMAC does not have an energy saving mechanism. In general, it can be seen that for broadcast-based protocols there is a stronger relation between the consumed energy and the number of transmitted bytes.

More specifically, the more bytes are transmitted the higher is the energy consumption per byte due to higher packet loss. The energy consumption of unicast-based protocols is generally good with the exception of Directed Diffusion, which performs rather poor due to additional maintenance messages, e.g., for path reinforcement or the propagation of new interest. Concerning the impact of the MAC protocol, BEAM offers higher energy-efficiency than ContikiMAC since the latter has a worse duty cycle mechanism. Furthermore, caching does not significantly influence the performance of broadcast-based protocols from an energy point of view.



(a) 1000 bytes (b) 20 bytes

Fig. 9. Energy consumption per node and per transmitted byte.

In Fig. 9(b) corresponding results on energy consumption are shown for the scenario with a single packet (20 bytes). As we can expect, energy consumption per transmitted byte is lower due to the smaller size of data to transmit; the transmission of 20 bytes implies a lower number of collisions and retransmissions.

5 Conclusions

We propose the Sensor Node Overlay Multicast (SNOMC) protocol to support a reliable, time-efficient and energy-efficient dissemination of bulky data from one sender node to many receivers. To ensure end-to-end reliability we designed and implemented a NACK-based reliability mechanism. Further, to avoid costly end-to-end retransmissions we propose different caching strategies implemented in SNOMC.

We compared the SNOMC protocol to other common protocols for data dissemination in terms of transmission time, number of transmitted packets and energy consumption. In general, SNOMC outperforms the other protocols. Further, we showed that our protocol performs well with different MAC protocols, which support different levels of reliability and energy-efficiency. We therefore conclude that SNOMC can offer a robust, high-performing solution for the efficient distribution of code updates in a WSN.

References

1. G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, S. Morgenthaler: MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks. *WWIC'08, Tampere, Finland, May'08*.
2. P. J. Marron, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, K. Rothermel: TinyCubus: A Flexible and Adaptive Framework for Sensor Networks. *EWSN'05, Istanbul, Turkey, Feb'05*.
3. J. W. Hui, D. Culler: The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. *SenSys'04, Baltimore, MD, USA, Nov'04*.
4. P. Levis, N. Patel, S. Shenker, D. Culler: Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. *NSDI'04, San Francisco, CA, USA, Mar'04*.
5. M. Anwander, G. Wagenknecht, T. Braun: Management of Wireless Sensor Networks using TCP/IP. *IWSNE'08, Santorini Island, Greece, Jun'08.*
6. G. Wagenknecht, M. Anwander, T. Braun: SNOMC: An Overlay Multicast Protocol for Wireless Sensor Networks. *WONS'12, Courmayeur, Italy, Jan'12*.
7. A. Dunkels, B. Grönvall, T. Voigt: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *EmNetS'04, Tampa, FL, USA, Nov'04*.
8. A. Quayyum, L. Viennot, A. Laouiti: Multipoint relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. *TR, INRIA, Sophia Antipolis, France, 2000*.
9. C. Y. Wan, A. T. Campbell, L. Krishnamurthy: PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. *WSNA'02, Atlanta, GA, USA, Sep'02*.
10. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva: Directed Diffusion for Wireless Sensor Networking. *ACM/IEEE Transactions on Networking, 11(1):2-16, Feb'02*.
11. A. Okura, T. Ihara, A. Miura: BAM: Branch Aggregation Multicast for Wireless Sensor Networks. *MASS'05, Washington, DC, USA, Nov'05*.
12. A. Sheth, B. Shucker, R. Han: VLM2: A Very Lightweight Mobile Multicast System For Wireless Sensor Networks. *WCNC'03, New Orleans, LA, USA, Mar'03*.
13. R. Flury, R. Wattenhofer: Routing, Anycast, and Multicast for Mesh and Sensor Networks. *INFOCOM'07, Anchorage, Alaska, USA, May'07*.
14. C. H. Feng, W. B. Heinzelman: RBMulticast: Receiver Based Multicast for Wireless Sensor Networks. *WCNC'09, Budapest, Hungary, Apr'09*.
15. B. Chen, K. Muniswamy-Reddy, M. Welsh: Ad-Hoc Multicast Routing on Resource-Limited Sensor Nodes. *REALMAN'06, Florence, Italy, May'06*.
16. J. S. Silva, T. Camilo, P. Pinto, R. Ruivo, A. Rodrigues, F. Gaudncio, F. Boavida: Multicast and IP Multicast Support in Wireless Sensor Networks. *Journal of Networks, 3(3),19-26,2008*.
17. D. Koutsonikolas, S. Das, Y. C. Hu, I. Stojmenovic: Hierarchical Geographic Multicast Routing for Wireless Sensor Networks. *SENSORCOMM'07, Valencia, Spain, Oct'07*.
18. J. A. Sanchez, P. M. Ruiz, I. Stojmenovic: Energy Efficient Geographic Multicast Routing for Sensor and Actuator Networks. *Computer Communications, 30(13), 2519-2531, Jun'07*.
19. J. Lee, E. Lee, S. Park, S. Oh, S. H. Kim: Consecutive Geographic Multicasting Protocol in Large-Scale Wireless Sensor Networks. *PIMRC'10, Istanbul, Turkey, Sep'10*.
20. OMNeT++: Discrete Event Simulation System, *http://www.omnetpp.org*.
21. M. Anwander, G. Wagenknecht, T. Braun, K. Dolfus: BEAM: A Burst-Aware Energy-Efficient Adaptive MAC Protocol for Wireless Sensor Networks. *INSS'10, Kassel, Germany, Jun'10*.
22. A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, N. Finne: The Announcement Layer: Beacon Coordination for the Sensornet Stack. *EWSN'11, Bonn, Germany, Feb'11*.
23. CC2420: Datasheet for the Chipcon CC2420 RF Transceiver, *Online, Dec'11*.
24. H. N. Pham, D. Pediaditakis, A. Boulis: From Simulation to Real Deployments in WSN and Back. *WoWMoM'07, Helsinki, Finland, Jun'07.*