NarrowCast: A New Link-Layer Primitive for Gossip-Based Sensornet Protocols

Tomasz Pazurkiewicz, Michal Gregorczyk, and Konrad Iwanicki

University of Warsaw, Warsaw, Poland
{tp277655,mg277528}@students.mimuw.edu.pl,
iwanicki@mimuw.edu.pl

Abstract. Although gossiping protocols for wireless sensor networks (sensornets) excel at minimizing the number of generated packets, they leave room for improvement when it comes to the end-to-end performance, namely energy efficiency. As a step in remedying this situation, we propose NarrowCast: a new primitive that can be provided by asynchronous duty-cycling link layers as a substitute for broadcasting for gossiping protocols. The principal idea behind the NarrowCast primitive is to allow a sensor node to transmit to a fraction of its neighbors, which enables controlling energy expenditures and reliability. We discuss methods of approximating the primitive in practice and integrating it with gossiping protocols. We also evaluate implementations of the approximations with Trickle, a state-of-the-art gossiping protocol, and X-MAC, a popular link layer based on low-power listening. The results show that—without sacrificing reliability—gossiping using even the simplest approximations of NarrowCast can considerably outperform gossiping based on broadcasting in energy efficiency.

1 Introduction

Gossiping is a compelling communication paradigm with numerous applications in sensornets, such as disseminating queries [1], aggregating information [2], or maintaining complex overlays [3], to name a few examples. The essence of gossiping is that each node has a local state, which it repeatedly broadcasts to its neighbors. Likewise, it integrates the states received from the neighbors with the local state. The global effect of these repeated, local state exchanges is that information is disseminated among the nodes, such that they can learn a query, collectively compute an aggregate, or construct an overlay. Importantly, the dissemination process does not require any routing infrastructure and is robust to network dynamics, which is crucial especially under mobility.

The robustness of gossiping comes at a cost, though. The repeated node state broadcasts, which allow gossip-based protocols to tolerate failures, packet loss, and mobility, also introduce a lot of redundancy in the traffic. The redundancy wastes node resources. At best, transmitting, receiving, and processing redundant information drains node energy and reduces the effective channel throughput. In extreme cases, such as a concentration of mobile nodes in an area, the resulting broadcast storms may even lead to a collapse of the entire dissemination process [4]. Gossiping in sensornets thus requires managing redundancy: on the one hand, redundancy must be sufficient to handle network dynamics; on the other hand, its negative performance effects must be minimal.

B. Krishnamachari, A.L. Murphy, and N. Trigoni (Eds.): EWSN 2014, LNCS 8354, pp. 1–16, 2014.

To date, the problem of redundancy management has been addressed at the network layer: in gossiping protocols themselves. For example, a gossiping protocol can make probabilistic decisions on whether to rebroadcast its state or not [5] or can wait listening for and counting its neighbors' broadcasts, so that its own one can hopefully be suppressed [1]. In general, as we elaborate in the next section, multiple techniques exist that allow gossiping protocols to limit the number of broadcast packets.

However, even though reducing the number of packets generated by a gossiping protocol improves the network-layer dissemination performance, we argue that it still leaves a lot of room for improvement with respect to the end-to-end performance, notably energy efficiency. For instance, when analyzed end to end rather than only from the network layer perspective, probabilistically rebroadcasting a packet wastes the energy of potentially many nodes that have already received and processed the packet. Likewise, counting duplicate neighbors' broadcasts requires energy for receiving and processing them. All in all, we argue that while redundancy management mechanisms at the network layer are necessary, if employed alone, they are inherently limited, as it is the link layer below that controls channel access and radio energy expenditures.

In support of our argument, we propose NarrowCast, a link-layer primitive that is a step toward improving the energy efficiency of gossiping in sensornets. NarrowCast targets the suboptimal combination of broadcast communication and gossiping: on the one hand, the link layer spends time and energy on ensuring that a broadcast reaches all neighbors of the transmitter; on the other hand, some of the neighbors discard the received data as redundant at the network layer, thereby wasting this effort. As a countermeasure, the NarrowCast primitive allows a node to transmit to a *fraction* of its neighbors. In effect, assuming that the resulting energy cost is proportional to the fraction, the gossiping protocol gains control over energy expenditures and robustness.

We evaluate NarrowCast in simulation and on a $\sim$100-node testbed. Being conceptually simple a primitive, NarrowCast is not trivial to implement in the real world, especially when aiming at minimal assumptions and maximal performance. For this reason, we present a few implementations that, under different assumptions, approximate NarrowCast for X-MAC [6], a popular sensornet link-layer protocol. We evaluate these implementations with Trickle [1], a state-of-the-art sensornet gossiping protocol. The results confirm that NarrowCast improves the energy efficiency of gossiping.

The rest of the paper is organized as follows. Section 2 surveys related work. Sections 3 and 4, respectively, introduce and evaluate NarrowCast. Section 5 concludes.

2 Related Work

Arguably, the simplest form of gossiping is flooding, that is, rebroadcasting received data once by each node. Flooding lacks any redundancy management. As a result, it does not ensure that data reach all nodes, and may cause broadcast storms [4].

2.1 Managing Redundancy of Gossiping

For these reasons, virtually all gossiping protocols employ techniques for managing redundancy. A popular technique is to have each node locally suppress its broadcast with

a given probability [4,5,7,8]. The probability can be preconfigured globally [4,5,7], but this may be suboptimal in networks with heterogeneous node densities or under mobility. Alternatively, the probability can be adapted by each node based on the perceived neighborhood size [8], which requires additional neighborhood estimation mechanisms. In any case, with an appropriate probability, such proactive techniques of redundancy management considerably limit the traffic without impairing robustness.

In contrast to the proactive probabilistic ones, reactive techniques rely on observing the dissemination process and acting accordingly. For example, before rebroadcasting, each node can count broadcasts from its neighbors [1,4,7,9]. If the number of such broadcasts exceeds a threshold, the node suppresses its own one. Again, the threshold can be fixed [1,4,7] or adapted dynamically [9]. Alternatively, each node can estimate the number of new nodes its broadcast would reach and suppress the broadcast if this number is too low [4,7,9]. The estimation can be based on the nodes' positions [4,7,9] or signal quality [4]. Nodes can also piggyback their neighborhood onto broadcast data [9]. Although estimation-based approaches are potentially more accurate, in practice, counting-based ones perform similarly and are easier to implement.

Whereas the previous techniques focus on limiting traffic, improving the reliability of gossiping typically boils down to broadcasting repeatedly and relying on the traffic-limiting techniques—possibly in combinations—to minimize redundancy. In particular, Trickle [1] uses counting and, in addition, dynamically increases interbroadcast intervals (details in Sect. 3.1). Likewise, GOSSIP3 [5] combines counting with probabilistic suppression. TARP [10], in turn, applies an entire sequence of rebroadcast rules.

All in all, in terms of network-layer packets, such algorithms perform well. Yet, their end-to-end energy efficiency is heavily influenced by the underlying link layer, whose medium access control (MAC) protocol determines the energy expenditures.

2.2 Link-Layer Support for Gossiping

However, sensornet MAC protocols are hardly ever optimized for gossiping. To date, the prevalent traffic pattern in sensornets has been all-to-one data collection, for which unicast communication over a virtual tree dominates. The popularity of this pattern is reflected in some MAC protocols that offer dedicated mechanisms [11]. Gossiping, in turn, assumes no virtual topology and currently relies on MAC support for broadcasting. There are two main approaches to providing such support: synchronizing nodes and probing the wireless channel. None of them is tailored to gossiping, though.

MAC protocols following the first approach [11,12,13,14] aim to ensure that a node mostly sleeps, thereby saving energy, but when it does wake up to broadcast, all its neighbors are awake as well. To this end, the nodes maintain synchronization. This, however, is problematic, especially if they move. In particular, since they mostly sleep, discovering them by a mobile node may take a lot of time [14]. Moreover, the node must decide whether to adopt their wake-up schedule or not, which is again not trivial if the global cost of the decision is to be low [14]. Alternatively, nodes may operate on multiple schedules [12], but this requires more energy and schedule-disposal policies. Finally, mechanisms are necessary for adapting to changing network conditions.

In contrast, MAC protocols following the second approach [6,15,16] do not maintain shared state, but rely on so-called low-power listening. As previously, a node mostly

sleeps and wakes up only periodically to check if another node is transmitting. However, the node wake-up schedules need not be synchronized. Instead, during a period guaranteeing that each neighbor will wake up—the low-power listening check interval—a broadcasting node either transmits data repeatedly [16] or transmits an announcement preamble followed by the data only at the period end [6,15] (details in Sect. 3.1). Although this asynchrony facilitates applications of such protocols in mobile sensornets, the prolonged transmissions incur a significant energy overhead [6,15,16].

All in all, as we argued previously, irrespective of the link-layer MAC scheme and despite network-layer redundancy management, the energy efficiency of gossiping based on broadcasting leaves room for improvement. To date, however, little work has been done in this direction. Gaba et al. [17] suggest that cross-layer mechanisms could provide gossiping protocols with feedback from the link layer on channel utilization, so that the protocols' reaction to network dynamics could be optimized. Yet, to the best of our knowledge, no sensornet cross-layer optimizations target gossiping. The interplay between gossiping and the link layer is in turn touched upon by Dunkels et al. [18] who provide a unified set of gossiping abstractions for different MAC schemes. Dunkels et al. [19] also propose an additional announcement layer that concatenates data broadcast by different applications. While not aimed particularly at gossiping, this solution could potentially improve gossiping in multi-application scenarios. In general, however, we are not aware of any solution that targets gossiping and takes the NarrowCast's approach: to abandon link-layer guarantees that a broadcast reaches all neighbors, as many of the neighbors will ignore the received data anyway at a higher layer.

3 NarrowCast Primitive

The principal idea behind NarrowCast is simple: to allow a node to transmit to a fraction of its neighbors. While solutions that utilize the primitive in an optimal manner constitute an avenue for future research (see Sect. 5), this paper aims to demonstrate that NarrowCast can offer performance benefits even for existing state-of-the-art solutions.

3.1 Assumptions and Prerequisites

To this end, as a gossiping algorithm for our discussion, we assume Trickle [1], as it is a compelling solution employed, among others, by popular dissemination protocols, collection protocols, and even Internet of Things standards. To guarantee that data eventually reach all nodes, every node running Trickle broadcasts the data repeatedly every T_{max} time units. Since for the sake of traffic T_{max} is normally large (on the order of minutes), the dissemination latency would be large as well. Therefore, as a counter-measure, whenever a new version of the data is produced at a node or the node receives such a version from its neighbor, it shrinks its interbroadcast interval to T_{min}. Since T_{min} is in contrast small (on the order of milliseconds), a broadcast storm may occur whenever all neighbors receiving the new data shrink their intervals and attempt to rebroadcast.

To alleviate broadcast storms, Trickle uses two redundancy management mechanisms (see Fig. 1). First, the interbroadcast interval of a node is doubled up to T_{max} with each broadcast by the node, that is, the duration of the i-th interval after learning a new

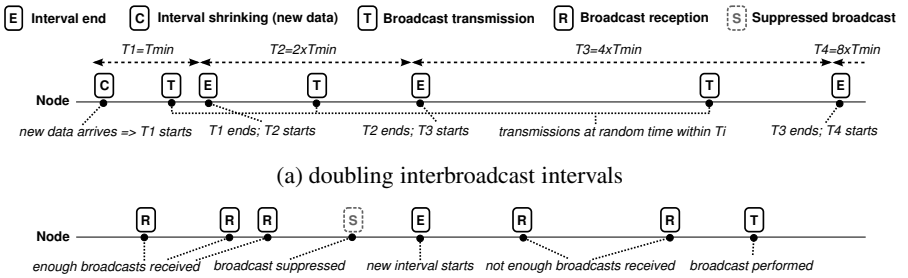
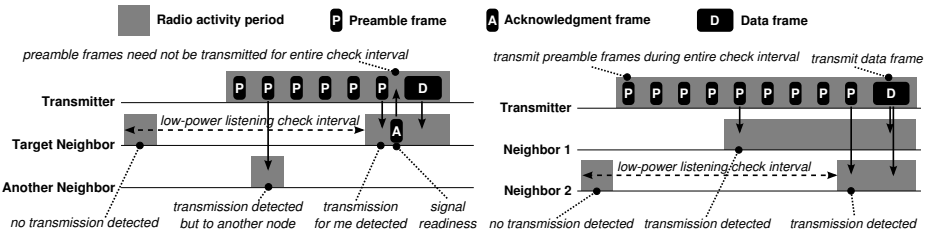


Fig. 1. The redundancy management mechanisms in Trickle

version of data is $T_i = MIN(2^{i-1} \times T_{min}, T_{max})$ [Fig. 1(a)]. This self-regulation mechanism enables recovering from a storm irrespective of the network density. Moreover, instead of transmitting exactly after T_i time units, a node draws a random time from $(0.5 \times T_i, T_i)$, which desynchronizes transmissions. Second, in every interval, each node counts broadcasts received with its version of the data. If their number exceeds a threshold (typically 2–3), the node suppresses its own broadcast in the interval [Fig. 1(b)].

As to the link layer, NarrowCast assumes that the average cost of broadcasting data in terms of energy and channel occupation is proportional to the fraction of neighbors receiving the data. All asynchronous MAC protocols based on low-power listening satisfy this assumption: the more data repetitions or the longer the transmitted part of a preamble, the higher the channel occupation and energy costs, but also the more neighbors awake to receive the data. In particular, our approximations of NarrowCast are built for X-MAC [6], a popular low-power listening MAC, suitable for mobile networks.

In X-MAC, an announcement preamble is a sequence of short frames, a so-called strobed preamble (see Fig. 2). The frames are separated by brief periods in which the transmitting node switches to reception mode. This is useful for unicast packets, as the receiver can acknowledge that it is up, thereby allowing the transmitter to terminate the preamble and send the actual data [Fig. 2(a)]. For broadcast packets, in turn, preambles must be transmitted during an entire low-power listening channel check interval



(a) unicast traffic (b) broadcast traffic

Fig. 2. An example of low-power listening with strobed preambles

[Fig. 2(b)], which is typically preconfigured globally. Again, we would like to stress that while we use X-MAC to illustrate and evaluate our ideas, any MAC protocol satisfying the above cost-proportionality assumption could likely benefit from NarrowCast.

3.2 Main Hypothesis and Idea

Analyzing them in combination, we may observe that gossiping and MAC protocols have independent reliability mechanisms. A MAC protocol bears the cost of waking up all neighbors, so that they can receive each broadcast packet. At the same time, (correctly) assuming that broadcasts are unreliable, a gossiping protocol repeats them. This functionality duplication negatively affects channel utilization and energy expenditures.

To illustrate, consider our combination of Trickle and low-power listening. Suppose that a node broadcasts new data. Its neighbors are awakened by the preceding preamble and receive the data. As a result, they all reset their Trickle intervals to T_{min} and attempt to rebroadcast. Recall that for latency reasons T_{min} is small, on the order of milliseconds. In contrast, to save energy, it is not uncommon for low-power listening preambles to last hundreds of milliseconds. This means that in the initial Trickle intervals multiple nodes want to rebroadcast simultaneously. Hopefully, the MAC protocol ensures that only one succeeds at a time. However, this implies that the others wait, possibly with active radios. Moreover, even though many nodes are already up, the rebroadcasting nodes still have to transmit their preambles, as they may have neighbors that were not in the range of the previous broadcasts, and hence, may yet be sleeping. All in all, the channel congestion and the resulting waiting period may be considerable, which inflates the energy expenditures. The situation is further aggravated when multiple data items are disseminated simultaneously, as is often the case in gossip-based applications.

NarrowCast tries to alleviate these effects by giving control to the network layer over the transmission reliability mechanisms at the link layer, and thereby, over the costs of communication. The idea is to have the link layer provide a communication primitive that allows the network layer (e.g., a gossiping protocol) to transmit, *narrowcast*, to a fraction of neighbors. Under our cost-proportionality assumption on the link layer, such narrowcasts can be proportionally shorter than broadcasts (e.g., have shorter preambles), which reduces channel utilization and energy costs. Moreover, we hypothesize that they need not compromise the reliability of dissemination, as the gossiping protocol will compensate with its own mechanisms (e.g., by repeating transmissions).

3.3 Implementation

To validate our hypotheses, we have implemented NarrowCast for the MiXiM framework [20] of the OMNeT++ simulator [21], a low-level simulation engine for sensornets, and for TinyOS 2.1, an operating system for sensor nodes. While NarrowCast is conceptually simple, its implementation is challenging: whereas unicasting a packet to one neighbor or broadcasting it to all is fairly straightforward, it is difficult to ensure that a packet is received by a *given fraction* of neighbors, especially in the absence of shared state. We have thus implemented only approximations of NarrowCast. They all share the idea of shortening packet preambles, but vary in assumptions and implementation effort. For simulation, we have also created a close-to-ideal oracle-based NarrowCast.

Incomplete Preambles (*IP*). The first approximation requires the least implementation effort. Suppose that the global interval in which every node wakes up to check the wireless channel for an ongoing preamble transmission is T_C. A NarrowCast transmission to a given percentage of neighbors, p, is preceded by an incomplete preamble, lasting $p \times T_C$. The motivation behind this idea is that if we assume that neighbor wake-up schedules are uniformly distributed, such an incomplete preamble lasting $p \times T_C$ should on average wake up p percent of neighbors, albeit without any hard guarantees.

Neighbor Cache (*NC*). The second approximation aims to improve the guarantees, assuming that nodes maintain state. More specifically, each node caches its neighbors' wake-up schedules. Before narrowcasting, it consults the cache to compute a preamble fraction that would wake up p percent of its neighbors. The cache is updated by piggybacking transmitters' wake-up schedules on packets. For eviction, the oldest unrefreshed schedules are chosen. While schedule maintenance is not coordinated among nodes, schedules get outdated, which may be problematic especially under mobility.

Colliding Acknowledgments (*CA-* and *CA+*). The third approximation also aims to improve the guarantees, but by means of acknowledgments. Upon waking up and receiving a preamble frame, a node transmits an acknowledgment frame. The frame is transmitted only once for the preamble of a given packet. When a sufficient number of acknowledgments in subsequent slots are observed by the transmitter, depending on the variant, the transmitter terminates the preamble and sends the actual data frame. In the CA- variant, the preamble lasts *at most* $p \times T_C$: it can be terminated earlier, as soon as acknowledgments in k slots are observed. In contrast, in the CA+ variant, the preamble lasts *at least* $p \times T_C$: it is not terminated before this time expires *and* acknowledgments in at least k slots are observed. CA- thus minimizes costs, while CA+ favors reliability.

A major problem with this approximation is that multiple nodes may wake up simultaneously and transmit acknowledgment frames. Dutta et al. showed that if such frames are identical and well timed, their collisions need not be destructive, but only for a few transmitters [22]. Therefore, for scalability, we do not rely on receiving acknowledgment frames, but merely on sensing a high channel state in acknowledgment slots. Nevertheless, this approximation still requires the most implementation effort.

Oracle (*OC*) [only simulations]. Finally, in OMNeT++, we have also implemented an oracle that informs a transmitter when a given percentage of its previously sleeping neighbors have been awaken by preamble frames. Upon such an event, the transmitter terminates the preamble and proceeds with a data frame.

While there are likely many other ways to implement NarrowCast, we believe the previous ones cover enough various techniques to assess the potential of the primitive well.

3.4 Integration with Higher-Layer Protocols

Likewise, there are several ways in which NarrowCast can be utilized by a gossiping protocol like Trickle. Again for brevity, we focus just on two representative ones.

First, instead of broadcasting, the protocol can decide to always narrowcast to a given percentage, p, of neighbors. We denote such a scheme ***fix***(p). For instance, narrowcasting to 50% of neighbors should intuitively reduce energy expenditures, perhaps even twice, without degrading reliability, unless the network is extremely sparse.

Second, a protocol like Trickle, in which the intervals between subsequent transmissions change dynamically, can also dynamically change the percentage of neighbors to which data is narrowcast. In particular, in every Trickle interval, the percentage can be multiplied by a factor $\alpha > 1$, assuming that the initial percentage (in T_1) is p_I. We refer to such a scheme as ***dyn***(p_I, α). The idea behind the scheme is that in the initial intervals data should be propagated fast, perhaps somewhat sacrificing reliability, hence the percentage can be low. Later intervals are in turn longer, and thus, a larger percentage can be utilized to reliably deliver the data to the rest of the nodes, but at a higher cost.

Finally, the NarrowCast-related interfaces provided by a link-layer are simple. The fraction of nodes to which a packet should be narrowcast can be made part of the packet's meta-data, which allows for per-packet manipulations matching the convention of TinyOS. An alternative solution is to designate separate NarrowCast addresses. For example, address `ffff` is considered a broadcast address in TinyOS. For NarrowCast, in turn, addresses `fff`x could be assigned, where $2^{x-15} \times 100\%$ denotes the percentage of nodes that should receive a packet destined to this address ($15 > x \geq 0$ or more).

4 Evaluation

We evaluate NarrowCast in OMNeT++/MiXiM and on a testbed. As mentioned previously, in all experiments we used Trickle (with T_{min} = 256 ms, T_{max} = 60 mins, and suppression threshold = 3) in combination with the aforementioned X-MAC-based implementations of NarrowCast (with different low-power listening check intervals).

The MiXiM framework [20] for OMNeT++ [21] strives to realistically simulate sensornet communication at the signal level. Furthermore, we tried to match the simulated radios to the radios of our sensor nodes, notably in terms of timing, throughput, encoding, and packet reception rate deterioration with distance. We simulated up to 400 nodes in static and mobile networks. In static networks, the nodes were arranged on a torus, and we varied their density. Likewise, for mobility we varied the density by adopting square playgrounds of different sizes, in which the nodes moved with human-scale speeds according to a random-waypoint model (min. speed = 0.3 m/s, max. speed = 1.66 m/s, max. stop time = 1 s). In addition, we let the mobility patterns to warm up for one hour before starting the experiments. Overall, the static networks allow us to systematically study the impact of node density on NarrowCast, whereas the mobile ones enable illustrating the effects of heterogeneity, borders, and connectivity dynamics.

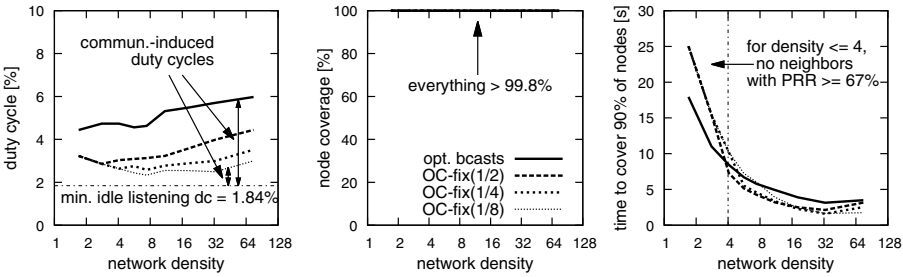
The testbed [23], in turn, spans 10 offices and consists of 102 sensor nodes with CC1101 868MHz radios, 96 of which took part in the experiments. For the employed radio transmission power, the nodes form a network of 7 hops, with a nonuniform density (from 7 to 47 neighbors per node), and more than 15% of asymmetric links. We thus believe that the testbed emulates a medium-scale real-world sensornet well.

Due to space constraints, we focus on experimental results that highlight only the main advantages and drawbacks of NarrowCast, thereby illustrating its potential.

4.1 Attainable Performance

Let us begin with a study of an ideal, oracle-based (*OC*) implementation of NarrowCast with one new data item gossiped every five minutes, such that the average radio duty cycle remains reasonable for sensornets: at a few percent. We study the dissemination of 60 items (300 minutes) in 400-node torus networks with various densities. Such networks are homogeneous, and hence, for each density, we can configure X-MAC with a global check interval that yields a minimal duty cycle. More specifically, we choose an interval optimal for Trickle based on broadcasting, and use it also for Trickle with narrowcasting. This guarantees that the comparisons are not be biased toward NarrowCast.

Figure 3 compares the simplest oracle-based NarrowCast, *OC-fix*, with broadcasting in terms of duty cycle (a), coverage (b), and latency (c) in relation to network density. Network density is defined as the average of local densities of all nodes. Local density is in turn defined probabilistically for each node: as the sum of packet reception rates from all other nodes (measured before the plotted experiments, independently for each node pair to avoid collisions). As such, this definition captures also extremely poor links.



(a) duty cycle (b) coverage (c) latency

Fig. 3. Narrowcasting vs. optimally configured broadcasting when gossiping one data item

Duty cycle—the percentage of time a node's radio is active transmitting, receiving, or idly listening—determines the energy consumption of the node. For all densities, gossiping with even the simplest, *fix*(p), variant of NarrowCast offers a lower duty cycle than optimally configured gossiping based on broadcasting, and the lower p is, the lower the duty cycle [Fig. 3(a)]. In particular, for the lowest plotted values of p, the duty cycle for narrowcasting is lower by up to a factor of 2 from the one for broadcasting.

The reduction is even more noteworthy considering the theoretical minimum of duty cycle (1.84% in the figure), which corresponds to no communication and only periodic low-power listening channel checks. If we subtract from the actual duty cycle the idle listening duty cycle, we obtain what we dubbed a *communication-induced duty cycle*, which describes dissemination efficiency. With this metric, we can observe that, in all but extremely sparse networks, the *fix*(p) variant of narrowcasting disseminates the same amount of information as broadcasting with nearly a factor of $1/p$ less energy. It is not precisely $1/p$, because some costs, such as activating the radio or transmitting actual data, are independent of p. For sparse networks, in turn, the duty cycle grows because *OC-fix* guarantees that p percent of neighbors (rounded up to 1) indeed wake

up to receive the data, to which end the transmitted preamble parts may be longer than $p \times T_C$. For a given p, this phenomenon happens at densities around $1/p$, that is, when the expected number of neighbors that wake up on a preamble part lasting $p \times T_C$ drops below 1. Nevertheless, even in the sparsest networks, NarrowCast reduces duty cycle.

At the same time, it does not impair reliability, measured with a standard coverage metric [Fig. 3(b)]: the percentage of nodes that receive each data item. The coverage is hardly ever below 90% (in the plotted experiments, it was above 99.8%), and in general, we have not observed major differences between narrowcasting and broadcasting. This validates our claim that reliability mechanisms in the communication stack are redundant for gossiping, so we can relax some of them without impairing reliability.

By and large, narrowcasting also improves the pace of dissemination, measured, for instance, as the latency to cover 90% of nodes [Fig. 3(c)]. The only exception is the sparsest networks, with densities below $1/p$. In such networks, due to the shorter low-power listening preamble parts and weak links, fewer (out of already few) neighbors of a transmitter have chances to hear the transmitted packet and start contributing to the dissemination process. In effect, not only does the process bootstrap slower, but also the latency penalties accumulate at each hop. Note, however, that we plotted such networks only for illustration, as their actual density, compared to our definition that also captures poor links, is extremely low. For example, no node has any neighbors with packet reception rates above 67% or even 50% in the sparsest plotted networks. Nevertheless, even in such challenged networks, the latency growth is not dramatic. In networks with a reasonable density, in turn, narrowcasting can reduce latency even twice. Moreover, for each density, there seems to be an optimal value of p that minimizes the dissemination latency. We leave an in-depth study of this phenomenon for future work.

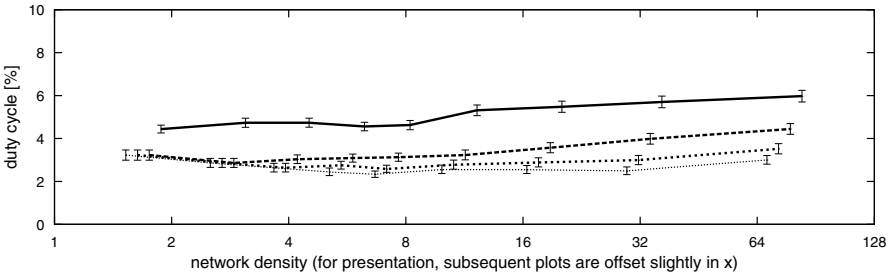
Finally, Fig. 4 presents the divergence of individual duty cycles (a) and latencies (b) from the averages plotted in Fig. 3, more specifically, the 10-th and 90-th percentile values for each of the approximations and densities in the 60 rounds of dissemination. In short, individual values are largely concentrated around the means, which suggests that the performance stability of narrowcasting is comparable to that of broadcasting.

All in all, the results confirm our hypotheses. Gossiping with narrowcasting can be more energy efficient than with broadcasting: it disseminates data with a lower channel utilization, and hence, duty cycle and typically latency. At the same time, its reliability is not impaired, being more than sufficient for gossip-based applications.

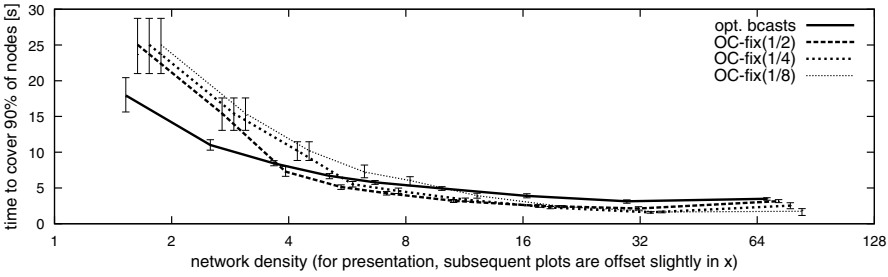
4.2 Quality of Approximations

However, the previous results concern an ideal, oracle-based implementation of NarrowCast. In contrast, in practice, we can rely only on approximations. Figure 5 thus presents the performance of our approximations (apart from *CA*-, as we explain shortly).

In dense and medium networks, the performance of the simplest approximation, Incomplete Preambles [*IP-fix*(p), Fig. 5(a)–(c)], is comparable to the oracle-based NarrowCast. In contrast, in sparse networks, the approximation performs poorly in terms of coverage (and hence, the latency to cover 90% of nodes), especially for low values of p. This is because in *IP-fix*(p) packet preambles are always transmitted for a *fixed* fraction, p, of the low-power listening check interval. In effect, while in dense networks or for large p, there is only a remote probability that insufficiently many neighbors wake



(a) duty cycle [cf. Fig. 3(a)]



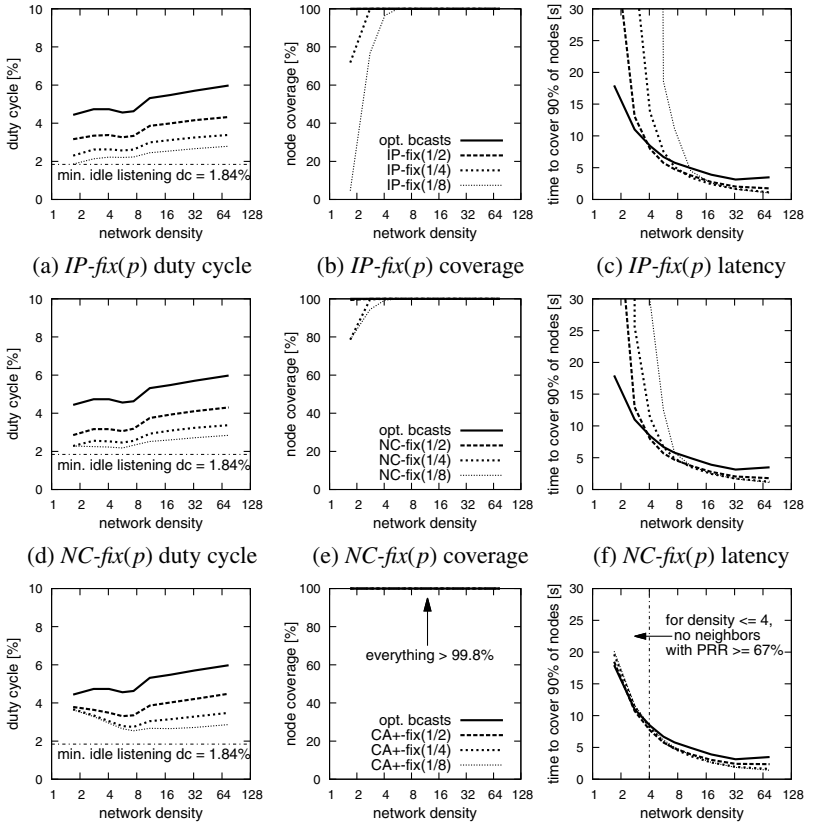
(b) latency [cf. Fig. 3(c)]

Fig. 4. Bars illustrating the 10-th and 90-th percentile of the values from Fig. 3

up in this fraction of time to check the channel, in sparse networks and for low values of p, this probability grows, which impairs coverage. Put differently, unlike *OC-fix*(p), *IP-fix*(p) is unable to compensate for network sparsity by extending the transmitted preamble parts if necessary, which can be observed by comparing for sparse networks the duty cycles of *IP-fix*(p) [Fig. 5(a)] and *OC-fix*(p) [Fig. 3(a)]: for *OC-fix*(p) they are higher because of the longer preambles. Nevertheless, as we discuss shortly, even for this simple approximation, we can improve the coverage with the $dyn(p_I, \alpha)$ scheme.

The second approximation, Neighbor Cache [*NC-fix(p)*, Fig. 5(d)–(f)], has similar drawbacks as *IP-fix*(p). Even though a cache of neighbors' wake-up schedules facilitates ensuring that sufficiently many neighbors have a chance to wake up during a preamble transmission, the cache has to be complete. However, this is hard to guarantee in sparse networks and for low values of p, because to add a neighbor to the cache, a node must be lucky to wake up and hear the neighbor transmitting a shortened preamble. The plots illustrate these effects, because in the corresponding experiments the node caches were deliberately purged every third dissemination. While one may argue that the caches could be maintained out of band, for many applications, especially mobile ones, the benefits of *NC-fix*(p) may still not compensate its drawbacks.

The performance of the final approximation, Colliding Acknowledgments [*CA(-/+)-fix*(p)], depends on the variant. Transmitting preambles for at most p percent of the low-power listening check interval or until k acknowledgments are observed (the *CA-* variant, not plotted) inherits the performance problems in sparse networks from *IP-fix*. In contrast, transmitting preambles for at least p percent of the check interval and until at least k acknowledgments are observed [the *CA+* variant, Fig. 5(g)–(i)])



(a) *IP-fix*(p) duty cycle (b) *IP-fix*(p) coverage (c) *IP-fix*(p) latency

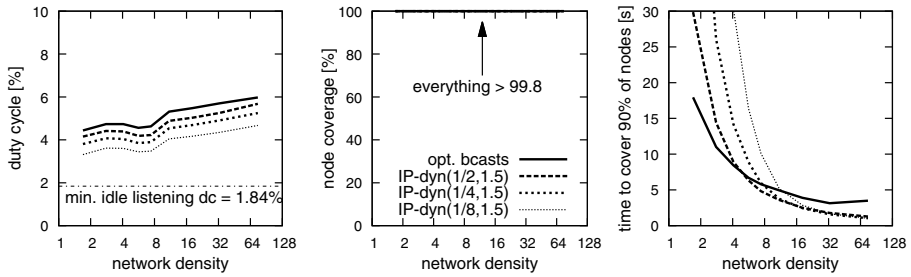
(d) *NC-fix*(p) duty cycle (e) *NC-fix*(p) coverage (f) *NC-fix*(p) latency

(g) *CA+-fix*(p) duty cycle (h) *CA+-fix*(p) coverage (i) *CA+-fix*(p) latency

Fig. 5. The performance of the approximations of NarrowCast (for *CA+*, $k = 2$)

performs best of all the approximations in the *fix* configuration. For all plotted network densities, its coverage is above 99.8% and the duty cycle and latency are low. The additional feedback in the form of k acknowledgments addresses the performance issues of the previous approximations in sparse networks, even for k as small as 2. In fact, due to this additional feedback, in the sparsest networks, the duty cycle for *CA+* is slightly higher than for *OC* [Fig. 5(g) vs. Fig. 3(a)] while the latency is lower [Fig. 5(i) vs. Fig. 3(c)]. All in all, *CA+* evidences that NarrowCast can be effectively approximated.

What is more, however, even with a simple, imperfect approximation, such as *IP*, we can maximize reliability by dynamically controlling the fraction of neighbors to which data is narrowcast: by means of the *dyn*(p_I, α) scheme. To illustrate, Fig. 6 presents the performance of *IP-dyn*(p_I, 1.5) for different values of p_I. In other words, after shrinking the Trickle interval, each node starts with a low NarrowCast fraction, p_I, and with each doubling of the Trickle interval, it multiplies this fraction by 1.5, until it reaches 1.



(a) duty cycle (b) coverage (c) latency

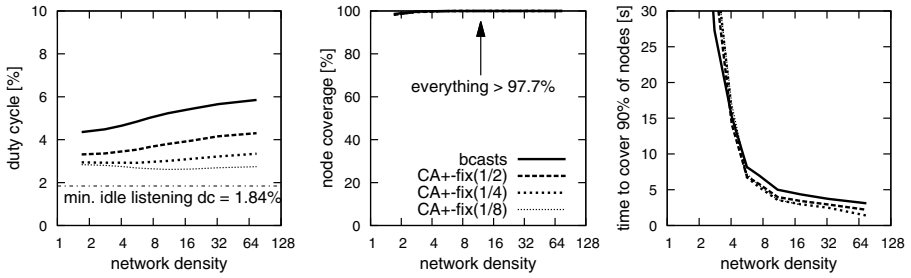
Fig. 6. Improving the reliability of Incomplete Preambles with the $dyn(p_I, \alpha)$ scheme

This scheme provides both a lower duty cycle [Fig. 6(a)] and a high coverage [Fig. 6(b)]. It does not significantly improve the latency in sparse networks [Fig. 6(c)], because in such networks dissemination at a node often progresses only when the node's NarrowCast fraction has grown sufficiently, which takes time. In dense networks, in turn, the latency is lower than for broadcasting. Finally, an additional advantage of *IP-dyn* over *CA+-fix* is simplicity: its TinyOS implementation is just a few lines of code.

In summary, NarrowCast is best approximated with *CA+*. Yet, even simple approximations, such as *IP* can offer a reasonable performance if accompanied with mechanisms for dynamically adjusting the fraction of neighbors to which data is narrowcast.

4.3 Effects of Network Heterogeneity and Dynamics

NarrowCast performs well also under network dynamics. In particular, Fig. 7 shows the performance of *CA+* under mobility on square playgrounds with the same dimensions as the toruses. Since mobility makes determining the optimal low-power listening check interval hard, for each playground, we used the value from the corresponding torus.



(a) duty cycle (b) coverage (c) latency

Fig. 7. Performance of Colliding Acknowledgments (*CA+*) under mobility ($k = 2$)

Again, narrowcasting outperforms broadcasting in duty cycle. At the same time, it offers a lower latency and virtually perfect coverage in all but extremely sparse networks. In the sparse networks, in turn, the coverage drops slightly, but so does the coverage for broadcasting. This is because in such networks, nodes get disconnected easily.

In particular, due to disconnections, the coverage for narrowcasting is sometimes better than for broadcasting. Nevertheless, even in the sparsest networks, the coverage is above 90% (in fact, 97.7% in the plots), and more importantly, we have not observed major differences between broadcasting and narrowcasting. Furthermore, the deviation of individual values from the averages, measured, for instance, by the 10-th and 90-th percentile values (not plotted), is not much higher than in the static torus networks. In other words, network dynamics do not impair the performance of NarrowCast.

4.4 Real-World Behavior

Finally, to show that NarrowCast works in the real world, we present experiments with the TinyOS implementations of the different approximations on our testbed. Although we have conducted several experiments with various applications employing NarrowCast, for consistency, in this paper we present only experiments with the same application as in the simulations, albeit extended to concurrently disseminate multiple data items. More specifically, d items ($d \in \{1, \ldots, 16\}$) are disseminated in the system, and for every such item, each node runs a dedicated instance of Trickle. Periodically, new versions of all items appear simultaneously at random nodes and are started being gossiped. The period is one hour, so that even for $d = 16$, the average duty cycle is low.

Table 1 compares broadcasting and selected approximations of NarrowCast in terms of communication-induced duty cycle and latency to cover 90% of nodes with all items. The coverage is omitted as it was consistently equal to 100%. All depicted values were obtained for the same 96 nodes of the testbed, and the dissemination scenarios were repeated 4 times. The idle listening duty cycles for given values of d were as follows: 0.97% for 1 event, 1.31% for 2 events, 1.64% for 4 events, 3.3% for 8 and 16 events.

Table 1. Testbed results for selected approximations of NarrowCast

	opt. bcast	***IP-fix(1/2)***	***IP-fix(1/4)***	***IP-dyn(1/2,1.5)***	***IP-dyn(1/4,1.5)***	***CA+-fix(1/2)***	***CA+-fix(1/4)***	***opt. bcast***	***IP-fix(1/2)***	***IP-fix(1/4)***	***IP-dyn(1/2,1.5)***	***IP-dyn(1/4,1.5)***	***CA+-fix(1/2)***	***CA+-fix(1/4)***
# concurrent disseminations (d)	**commun.-induced duty cycle [%]**							**latency to cover 90% of nodes [s]**						
1	0.95	0.39	0.15	0.34	0.2	0.56	0.43	4.39	1.37	2.38	2.05	2.03	3.07	2.23
2	1.37	0.58	0.28	0.48	0.32	0.82	0.64	30.5	3.99	3.82	3.35	2.23	7.9	7.91
4	1.62	0.55	0.31	0.59	0.32	1.13	0.91	42.8	8.22	3.96	7.82	4.97	16.1	9.97
8	1.53	0.81	0.42	0.78	0.54	1.4	1.08	65.3	14.5	6.58	16.1	8.62	33.9	22
16	3.52	1.25	0.67	1.36	0.75	2.6	2.12	121	51.4	34	57.8	31	61.2	60.1

As is typically the case in sensornets, the experimental results differ somewhat from the simulations, despite the fine-tuning of the simulated radios. In particular, the absolute numbers for duty cycle and latency vary due to the differences in the network topologies and wireless communication phenomena not modeled by the simulator. Likewise, colliding acknowledgments perform worse in the real world than in simulation.

This is because whereas in the simulator clear channel assessment is nearly perfect, in our TinyOS implementation, it is configured conservatively. In effect, transmitted acknowledgments are sometimes ignored, which slightly inflates duty cycle and latency. While we were not compelled to address this issue, devising mechanisms optimized for particular radio chips constitutes an interesting problem. There are also a few other minor differences that can be attributed to the testbed topology and experimental settings.

Nevertheless, despite these differences in absolute values, the basic trends from the simulations remain valid. Narrowcasting to a fraction of neighbors instead of broadcasting to all reduces the communication-induced duty cycle by a factor almost inversely proportional to the fraction. By and large, it also reduces the dissemination latency. Finally, it hardly affects the coverage. The testbed experiments thus confirm that the approximations of NarrowCast can in practice improve the performance of gossiping.

5 Conclusions and Future Work

In summary, the results demonstrate that NarrowCast—a link-layer primitive allowing a node to transmit to a fraction of its neighbors—can be effectively implemented in the real world and can indeed improve the end-to-end performance of gossiping protocols. In particular, for Trickle running on top of X-MAC, even the simplest implementations of NarrowCast can reduce the energy consumption and latency by a significant factor, without sacrificing reliability. In general, this reinforces our initial argument that the efficiency of sensornet gossiping protocols leaves room for improvement.

Therefore, since gossiping is a compelling communication paradigm in sensornets, especially under mobility, we believe that our work will inspire novel solutions, designed from the end-to-end perspective. In particular, we are currently working on protocols tailored specifically to NarrowCast. Furthermore, since applications of NarrowCast stretch beyond gossiping, investigating such applications constitutes another research avenue. For instance, we are studying the use of NarrowCast for routing. Finally, we are also working on proving the properties of NarrowCast analytically.

Acknowledgments. This research was supported by the (Polish) National Science Center within the SONATA program under grant no. DEC-2012/05/D/ST6/03582. The authors would also like to thank Albana Gaba for inspiring discussions on cross-layer optimizations.

References

1. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proc. USENIX NSDI 2004, San Francisco, CA, USA, pp. 15–28 (2004)
2. Xiao, L., Boyd, S., Lall, S.: A scheme for robust distributed sensor fusion based on average consensus. In: Proc. IPSN 2005, Los Angeles, CA, USA, pp. 63–70 (2005)
3. Iwanicki, K., van Steen, M.: Multi-hop cluster hierarchy maintenance in wireless sensor networks: A case for gossip-based protocols. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 102–117. Springer, Heidelberg (2009)

4. Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. In: Proc. ACM/IEEE MobiCom 1999, Seattle, WA, USA, pp. 151–162 (1999)
5. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. IEEE/ACM Trans. Netw. 14(3), 479–491 (2006)
6. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proc. ACM SenSys 2006, Boulder, CO, USA, pp. 307–320 (2006)
7. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proc. ACM MobiHoc 2002, Lausanne, Switzerland, pp. 194–205 (2002)
8. Gaba, A., Voulgaris, S., Iwanicki, K., van Steen, M.: Revisiting gossip-based ad-hoc routing. In: Proc. IEEE ICCCN 2012, Munich, Germany (2012)
9. Tseng, Y.C., Ni, S.Y., Shih, E.Y.: Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. IEEE Trans. Comput. 52(5), 545–557 (2003)
10. Gburzynski, P., Kaminska, B., Olesinski, W.: A tiny and efficient wireless ad-hoc protocol for low-cost sensor networks. In: Proc. DATE 2007, Nice, France, pp. 1557–1562 (2007)
11. van Hoesel, L., Havinga, P.: A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In: Proc. INSS 2004, Tokio, Japan, pp. 205–208 (2004)
12. van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proc. ACM SenSys 2003, Los Angeles, CA, USA, pp. 171–180 (2003)
13. Ye, W., Silva, F., Heidemann, J.: Ultra-low duty cycle MAC with scheduled channel polling. In: Proc. ACM SenSys 2006, Boulder, CO, USA, pp. 321–334 (2006)
14. Dobson, M., Voulgaris, S., van Steen, M.: Merging ultra-low duty cycle networks. In: Proc. IEEE/IFIP DSN 2011, Hong Kong, China, pp. 538–549 (2011)
15. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proc. ACM SenSys 2004, Baltimore, MD, USA, pp. 95–107 (2004)
16. Sun, Y., Gurewitz, O., Johnson, D.B.: RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In: Proc. ACM SenSys 2008, Raleigh, NC, USA, pp. 1–14 (2008)
17. Gaba, A., Voulgaris, S., van Steen, M.: Towards congestion-aware all-to-all information dissemination in mobile ad-hoc networks. In: 2010 IEEE GLOBECOM Workshops, Miami, FL, USA, pp. 1690–1695 (2010)
18. Dunkels, A., Österlind, F., He, Z.: An adaptive communication architecture for wireless sensor networks. In: Proc. ACM SenSys 2007, Sydney, Australia, pp. 335–349 (2007)
19. Dunkels, A., Mottola, L., Tsiftes, N., Österlind, F., Eriksson, J., Finne, N.: The announcement layer: Beacon coordination for the sensornet stack. In: Marrón, P.J., Whitehouse, K. (eds.) EWSN 2011. LNCS, vol. 6567, pp. 211–226. Springer, Heidelberg (2011)
20. MiXiM Homepage, `http://mixim.sourceforge.net`
21. OMNeT++ Homepage, `http://www.omnetpp.org`
22. Dutta, P., Musaloiu-E., R., Stoica, I., Terzis, A.: Wireless ACK collisions not considered harmful. In: Proc. ACM HotNets-VII, Calgary, Alberta, Canada (2008)
23. Michalowski, M., Horban, P., Strzelecki, K., Migdal, J., Klimek, M., Glazar, P., Iwanicki, K.: A sensornet testbed at the University of Warsaw. Technical Report TR-DS-01/12, University of Warsaw, Warsaw, Poland (2012), `http://www.mimuw.edu.pl/%7Eiwanicki`