Resilient Coding Algorithms for Sensor Network Data Persistence

Daniele Munaretto[1], Jörg Widmer[1], Michele Rossi[2], and Michele Zorzi[2]

[1] DoCoMo Euro-Labs, Landsberger Strasse 312 – 80687 Munich, Germany
[2] DEI, University of Padova, via Gradenigo 6/B – 35131 Padova, Italy

Abstract. Storing and disseminating coded information instead of the original data can bring significant performance improvements to sensor network protocols. Such methods reduce the risk of having some data replicated at many nodes, whereas other data is very scarce. This is of particular importance for data persistence in sensor networks. While coding is generally beneficial, coding over all available packets can be detrimental to performance, since coded information might not be decodable after a network failure. In this paper we investigate the suitability of different codeword degree distributions with respect to the dynamics of the underlying wireless network and design a corresponding data management algorithm. We further propose a simple buffer management scheme for continuous data gathering. The performance of the protocols is demonstrated by means of simulation, as well as experiments with an implementation on MICAz motes.

1 Introduction

Data collection is the primary task of a wireless sensor network. To this end, the sensed data has to be transported to the sink node(s) or should be stored within the network in case no sink node is currently available. Due to the power and memory constraints of the sensor nodes, this has to be done as efficiently as possible. Network coding [1] was shown to provide significant benefits in such networks. Several papers have analyzed the benefits of random network coding [2] for information dissemination and data persistence [3] (the amount of information that can be decoded at any give time). These methods reduce the risk of having some data replicated at many nodes, whereas other data is very scarce (in analogy to the coupon collector's problem [4]). The robustness that can be achieved through the diversity of available information by coding at intermediate nodes can be crucial in sensor networks, where node failures may be common.

While coding is generally beneficial, coding over all available packets might leave coded information undecodable after a network failure, thus reducing data persistence. Algorithms such as Growth Codes (GC) [5], a variant of LT codes [6], address this issue by using low complexity coding algorithms together with a code degree distribution that maximizes data persistence. These concepts are generalized in [7], considering coding over multiple snapshots of data and more general random mixed coding schemes. Growth Codes are designed for networks where the information available at neighboring nodes is uncorrelated, i.e., very sparse networks with a topology that changes significantly from one transmission to the next.

In this paper we investigate the suitability of different codeword degree distributions with respect to the dynamics of the underlying wireless network. In particular, we also investigate more static settings than those analyzed in previous research and discuss their implications on the optimum degree distribution. We then design a corresponding data dissemination algorithm that works well over a wide range of different network scenarios. To allow autonomous operation over an extended period of time in the face of a small amount of available RAM, nodes usually use their on-board flash memory. Since writing to (and to a lesser degree also reading from) the flash is very energy consuming, the coding algorithm has to make sure that the data required for encoding and decoding is available in RAM, and only data that is unlikely to be used again in the near future is written to the flash. We propose a simple buffer management scheme that allows for continuous data gathering, without using an excessive amount of writes to the flash memory.

The protocol is implemented on the MICAz mote platform. We perform a range of experiments to demonstrate its performance and compare it to previously proposed solutions. We further use simulation to investigate the scalability of the proposed approach in larger networks.

The paper is structured as follows. In Section 2 we review related work. Section 3 gives a brief overview of network coding. In Section 4 we present our novel algorithm based on network coding, analyzing suitable degree distributions for coding. Section 5 provides detailed simulation and experimental results on real sensor nodes for the different coding algorithms and degree distributions. In Section 6 we present a buffer management scheme to handle multiple temporal generations of data and Section 7 concludes the paper.

2 Related Work

The usefulness of network coding for data storage was investigated in [3], where the authors showed that a simple distribution scheme using network coding and only based on local information can perform almost as well as the case where there is complete coordination among nodes. Similar considerations also apply to sensor networks.

Growth Codes [5] were specifically designed to enhance data persistence, i.e., to maximize the amount of information that can be decoded at any time instant. Sensor nodes send out codewords that can be coded over multiple original information units. Nodes exchange codewords with their neighbors and combine received codewords with the existing local information, such that the stored information is coded over more and more information units over time.

The number of original information units a stored codeword is coded over is referred to as codeword degree. The authors in [5] propose to gradually increase the codeword degree with the amount of received information, hence the name “Growth Codes”. This codeword degree distribution optimizes sensor network data persistence in the presence of node failures, as it allows to decode the joint information of any subset of nodes with high probability. Intuitively, a high degree increases the probability that transmissions are innovative in that they bring new information to neighbors, while a low degree increases the probability that the information can be decoded immediately upon reception,

thus decreasing the likelihood that nodes will be left with undecodable information in case parts of the sensor network fail. As mentioned in the introduction, Growth Codes work well in case the information available at neighboring nodes is uncorrelated. As shown in [8], performance degrades in less dynamic situations, which are more likely to be found in sensor networks scenarios.

An extension of the Growth Codes work is described in [7], where the problem of collecting multi-snapshots spatial data in a resource constrained sensor network is addressed. Starting from [5], which provides an example of single snapshot data collection, the authors of [7] combine coding and scheduling to maximize the system's utility. They implement two algorithms, with and without mixing the snapshots, where only in the latter case a schedule is needed to improve the total utility gain. The scheduling problem is modeled through the Multi-Armed Bandit theory and solved optimally using Gittins Indices. They also demonstrate that there exists an optimal degree for the snapshots-mixed coding, which achieves maximum utility gain and data persistence.

We observe that without coding, data collection becomes equivalent to a coupon collector problem [9] which takes $\mathcal{O}(N \log N)$ coupons (symbols) for recovering the N original symbols. Existing coding techniques help avoiding the related heavy tail collector effect. However, channel codes such as LT Codes [6] and Reed-Solomon Codes [10] start decoding only after accumulating a large number of received packets, which is not suitable for resource constrained sensor nodes (due to, e.g., limited memory) and for data persistence. Persistence and reliability of cached data can be improved through Fountain Codes, as shown in [11]. The authors use Belief Propagation (BP) for a low decoding complexity. Random walks are used to disseminate coded data in a scalable way. The paper is related to our work, addressing the problem of data persistence when sinks are not available, but it uses the Robust Soliton degree distribution, which limits the range of applicable scenarios. Close to this work, [12] proposes a decentralized implementation of fountain codes. Erasure codes lead to reduced communication, storage and computational cost over random linear coding. One main drawback is to consider only one data packet stored in each node, and then multiplied in loco with incoming new symbols, which wastes the capability of the sensor nodes and increases rapidly codeword degrees without taking into account the network topology. Another drawback is the data dissemination process via pre-routing, in particular geographic routing, which requires each node to know its own location. Pre-routing is the process by which each node, before the data collection can take place, routes its data packet to d randomly selected nodes, which will be XORing what they receive. In [13], the authors extend this approach by showing that if these conditions are slightly relaxed, a constant pre-routing degree suffices.

The main difference of our work with respect to previous research is that we specifically analyze codeword degree distributions providing a high degree of resilience to node/network failures for a much wider range of scenarios (than, e.g., in [5]). In addition, we present a thorough discussion on the degree distributions that work well, designing a full data dissemination algorithm, which we complete with a buffer management scheme. Finally, we provide a performance analysis through experiments with a real-world implementation of the algorithm on sensor motes.

3 Network Coding

With network coding, nodes transmit packets coded over multiple original packets, instead of uncoded data. Coded packets can contain information from many different data sources. For coding, sets of s consecutive bits of a packet are treated as a symbol over the Galois field $GF(q)$, with $q = 2^s$, and an L bits long packet consists of L/s symbols. Note that coded packets have the same length as uncoded data packets. Due to its simplicity, usually *linear* network coding is used, where packets are linear combinations of the original packets.

For random linear network coding, a packet Y coded over the original packets $X^1, ..., X^n$ is generated by multiplying each with a random coding coefficient g_i to obtain $Y = \sum_{i=1}^{n} g_i X^i$. This is done individually for each symbol in the data packet. It is not necessary to first decode received data in order to create new coded packets, but the same operations can be applied recursively to already coded data.

Decoding requires knowledge of the coding coefficients. For simplicity, assume that a packet contains both the coefficients $g = (g_1, ..., g_n)$ and the encoded data [14]. Assume a node has received $(g^1, Y^1), ..., (g^m, Y^m)$. Decoding requires solving the system of equations $\{Y^j = \sum_{i=1}^{n} g_i^j X^i\}$ to retrieve the original X^i. In case $m \geq n$ and n of the equations are linearly independent, all data can be recovered.

The special case $GF(2)$ with a field size of 2 is very appealing for sensor networks since it only requires addition over a finite field (which corresponds to a simple `xor`) and no multiplication. Also the coding coefficients are only a single bit which reduces overhead and decoding complexity. Different algorithms are suitable for decoding. Network coding schemes often use Gaussian elimination to invert the matrix of coding coefficients, but also methods with lower computational complexity (e.g., message passing [6]) can be effectively used in some cases.

To cope with the limited node memory, it is necessary to limit n, the number of packets that can be coded over at the same time. Packets are grouped into generations [14], and only packets from the same generation can be combined in the encoding process.

4 Coding Degree Distributions for Static and Mobile Networks

4.1 Description of the Algorithm

In this section we present a novel network coding (NC) algorithm, called *adaptive network coding* (ANC). In contrast to previous schemes ANC uses specific degree distributions (specified in detail later) in order to adapt its behavior to the type of mobility in the network. Each node has a buffer of limited and known size, which should be however larger than the size of the current generation (i.e., of the number of packets that are to be processed together and eventually distributed to all nodes). This buffer may contain encoded as well as original information packets, which may be combined to produce additional encoded packets through random linear coding. Whenever a transmission opportunity occurs, NC schemes usually (see, e.g., [15]) code over all packets in the buffer. We however advocate that coding over *all* available linearly independent

packets (maximum degree encoding) at all times is not always optimal in terms of performance. This was observed in [5], where the authors showed that depending on the number of decoded or *recovered* packets r at a specific node there exists an optimal number of packets to combine to get close to optimal performance in terms of number of decodable packets at each instant in time. The optimum degree distribution however depends on the dynamics of the underlying network and in [8] the authors show the deficiencies of Growth Codes in scenarios other than the very specific one they were designed for.

In this section, we design suitable network coding algorithms based on our findings on the impact of network dynamics such as node mobility and channel conditions. We use the following definitions. The *packet degree* is the number of original information packets which are combined together to form a packet. The degree distribution gives the degree that a packet to be sent should have to give maximum performance under certain network conditions: if r is the number of packets recovered at a given node, the degree distribution returns $\mathcal{D}(r)$, i.e., the degree of the next output packet.[1]

We further say that a transmission opportunity for a node occurs when this node is selected for transmitting a new encoded packet. The actual transmission schedule is not specified in detail here and could be, e.g., either TDMA based or event based. In the former case, a distributed or centralized TDMA schedule is assumed, whereas in the latter a new encoded packet is usually sent as the node receives innovative information from its neighbors (see [15]). An example of a fully distributed approach for the selection of transmission schedules can be found in [16], where the authors propose Proactive Network Coding (ProNC). According to this strategy, every node infers transmission times as well as the data rate to use, based on incoming innovative information and on messages it receives from its neighbors. This scheme has been proven to perform very close to a mechanism exploiting partially centralized and optimal transmission schedules.

ANC works as follows. When a transmission opportunity occurs, the node randomly combines a number of packets in its buffer, so that the resulting packet has a degree that is as high as possible while being less than or equal to $\mathcal{D}(r)$. If this degree can not be obtained, the algorithm combines all packets in the buffer, obtaining a degree strictly lower than $\mathcal{D}(r)$. It is easy to determine the degree of a packet from the number of non-zero entries in the corresponding coding vector. At the first transmission, the buffer is empty and the node generates and transmits a packet containing only the node's own information. Upon receiving a new packet, a node first checks whether this packet increases the rank of the decoding matrix (which is formed by the packets in the node's buffer). If this is the case, this packet is stored in the first empty position of the buffer. Otherwise, the packet is discarded as it is useless for data recovery purposes. In the decoding process, early decoding of some information packets may occur before all packets have been recovered, thereby increasing r, the number of packets recovered by a given node so far. This often happens in practice due to the manner in which information propagates and is *gradually* coded over more and more other information (i.e., there

[1] The name degree *distribution* is used to recall the stochastic nature of the encoding process by which $\mathcal{D}(r)$ packets are randomly and uniformly picked among those in the node's buffer. There is, however, an abuse of notation here. In fact, differently from [6] the number of packets to encode $\mathcal{D}(r)$ is deterministic once we know r.

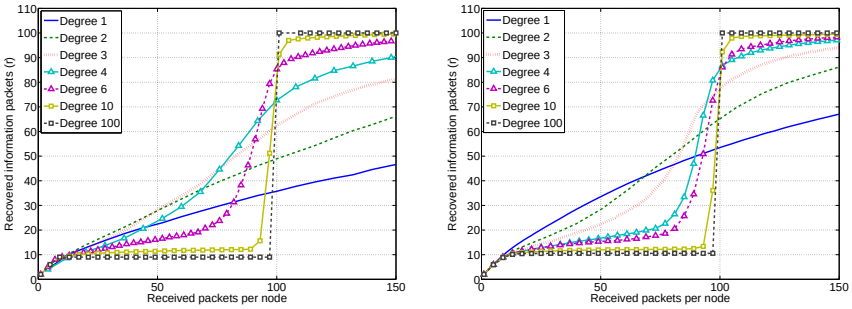

Fig. 1. Average number of packets recovered per node vs. number of packets received for a network of size $N = 100$, static (left) and moderate mobility (right) scenario.

is a codeword degree distribution inherently given by the information dissemination process).

The use of a topology-dependent degree distribution is the main difference between the present algorithm and previous schemes. This modification has a significant impact and the improvements in terms of dissemination time and total number of decoded packets sent are substantial, as can be seen from the experiments later on. In the next section we discuss the properties that a good degree distribution should have as a function of the network dynamics.

4.2 Discussion on Degree Distributions

We use the network coding scheme presented in the previous section for our analysis of the degree distribution's impact on the performance of the dissemination algorithms. For the analysis, we first consider a static grid topology (referred to as STATIC in the experimental results of Section 5). Subsequently, we study the effects of node mobility in a moderate random mobility scenario and a so called *random encounter* mobility scenario (RE in Section 5). In the former, nodes move according to a random way point mobility model with speeds uniformly distributed in the interval $[2, 4]$ m/s, whereas in the latter they move in a completely uncorrelated fashion such that the neighbors of a given node at any time instant are independent of the neighbors of the same node at any other instant. This latter case corresponds to the scenario analytically investigated in [5]. It is somewhat unrealistic in practice, except for extremely sparse and highly mobile networks with very sporadic data exchange.

We start our investigation with the static network case. We consider $N = 100$ nodes in a grid, where every node has exactly 8 neighbors. For the degree distribution we keep the encoding degree $\mathcal{D}(r)$ at a fixed value, independent of r, during the entire dissemination process. Hence, we run simulations by varying the encoding degree from 1 to N in steps of one unit.[2] The simulation results for this case are shown in Fig. 1, on the left side, where we only plot results for a few selected degrees for the sake of clarity.

[2] A customized C++ simulator was written to this end.

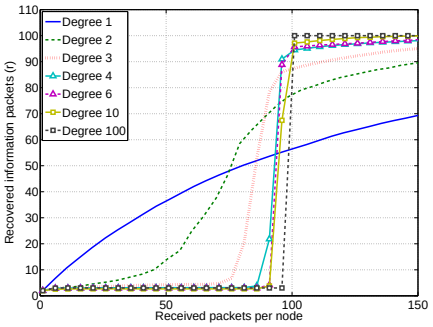

Fig. 2. Average number of packets recovered per node vs. number of packets received for a network of size $N = 100$, random encounter mobility scenario.

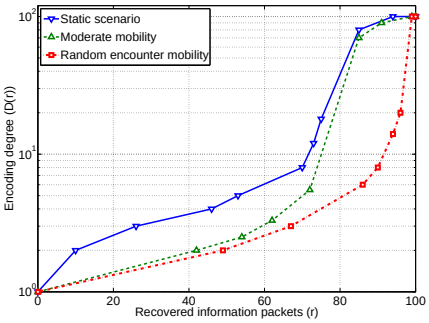

Fig. 3. Comparison among the optimal degree distributions in the three mobility scenarios: $\mathcal{D}(r)$ vs recovered packets, r.

In particular, this figure shows the average number of packets recovered per node, r, as a function of the number of packets received. This plot, as well as the plots that remain to discuss in this section, were obtained through a large number of simulations, so as to get sufficiently tight confidence intervals (within $\pm 3\%$ of the plotted values). These intervals, for the sake of readability, are not shown inside the figures. Fig. 1 clearly emphasizes that the actual degree in use does matter. Very large degrees (e.g., $N = 100$) tend to have good performance, in that they allow full recovery very early on. However, they typically present a step behavior, i.e., very little can be decoded up to a certain point, and then the recovery rate suddenly jumps to 100%. By contrast, smaller degrees give a smoother recovery curve. Due to the static nature of the network and the fixed node density, there is little difference for very high degrees (around 50 and above). The early recovery of useful information through a lower degree coding is preferable in some cases. In particular, in case of a network failure or partition before full recovery, nodes with such heavily coded information cannot make any use of what they retrieved so far.

In the right side plot of Fig. 1, we show similar results for the moderate mobility scenario (again with an average node density of 8 neighbors per node). From the figures we can see that mobility helps to disseminate information more quickly, in the same way as a higher degree distribution does. For example, the curve for degree 6 in the static scenario coincides with the curve for degree 4 in the mobile one. In particular, the performance of low coding degrees is improved through mobility. Finally, in Fig. 2 we report the same results for the random encounter case. Here, the trend is even more pronounced, and very low degrees of 1, 2, and 3 perform extremely well, compared to their performance in the static case.

In all of these graphs, the curves intersect at specific points. Using these crossing points it is therefore possible to define an "optimal" degree distribution by moving along the x-axis of each graph and selecting the curve (i.e., the degree) which maximizes the

number of packets recovered, r. Such a distribution is plotted in Fig. 3 for all of the three scenarios considered here.

Note that the "optimal" distributions we obtain in this way only approximate the true optimal curves. Our distributions were obtained offline through the analysis of the simulation results we obtained for fixed degrees and, in turn, we neglected the dynamics involved in varying the codeword degree during the dissemination process. Nevertheless, we observe that the distribution for the random encounter scenario very closely matches the true optimal distribution in this case, see [5, 7]. This provides evidence about the validity of our approach. An exact analysis for the static and moderate mobility cases is still missing in the literature and is one of the objectives of our future research.

Notably, these optimal distributions increase $\mathcal{D}(r)$ slowly for small r. However, their degree $\mathcal{D}(r)$ increases sharply as r approaches N. This makes sense as when the number of recovered packets becomes sufficiently large, it is convenient to code over packets with large degree (large $\mathcal{D}(r)$) in order to maximize the probability that the few missing packets are included with high probability in the new encoded packets. Moreover, in the random encounter case, this sudden increase of the encoding degree $\mathcal{D}(r)$ occurs for higher values of r. This is mainly due to the fact that mobility contributes to the redistribution of data in the network. Such a redistribution is however absent in the static case and should be compensated for by the dissemination protocol through a more aggressive encoding (i.e., a larger $\mathcal{D}(r)$). Moreover, we verified that the seemingly small difference, for small values of r, between the distributions in the static and in the moderate mobility scenarios is however very important in terms of performance. For the moderate mobility case which, in terms of mobility, lies between the static and the random encounter scenario, we observe a further interesting fact. In particular, its distribution is very close to that of the random encounter case up to a certain value of r ($r \approx 70$ in Fig. 3), while it approaches the optimum distribution of the static case for larger r. The mobility in this scenario provides a sufficient mixing of the information in the initial delivery phase, whereas this is insufficient to ensure a prompt complete recovery when there is only a small number of packets left to recover.

Finally, given the importance of picking the right distribution as a function of the type of mobility, we may think of a distributed algorithm to monitor the dynamics in the set of neighbors in order to select, and possibly change, the degree distribution in use. This scheme is also part of our future research.

5 Experimental Results

In this section we discuss and present our experimental results on real sensor nodes. Our tests are run on MICAz XBow motes with a CC2420 radio chip (working at 2.4 GHz) [17] and an MPR2400CA processor based on the Atmel ATmega128L. The transmission range of these sensor nodes for indoor transmission is about 25 meters. As we obtained our experimental results in a laboratory or 36 square meters, we had to scale down the transmission power so as to get a proper multi-hop environment. Because of the limited number of sensors available, TOSSIM simulations were run to analyze our protocol in networks with a larger number of nodes, e.g., $N = 100$ (see below for a

short description of the TOSSIM simulator). Hence, we measure the average number of recovered information packets as a function of the number of packets received per node for grid (with four neighbors per node), line and random topologies. For each scenario we obtain the performance of our algorithm, ANC. For comparison, we also plot results for the scheme proposed in [5], referred to here as *growth codes based dissemination GC* and the pure network coding based scheme in [15], referred to here as *network coding*. ANC is then evaluated considering the optimal distributions discussed above for the random encounter (ANC-RE) and the static scenario (ANC-STATIC). Note again that *network coding* encodes data through the linear combination of all packets (of a given generation) in the node's buffer. *Growth codes based dissemination* uses the RE distribution, i.e., the optimal distribution in the random encounter scenario. However, in this scheme only one packet, which must contain the node's own information packet, can be used to increase the encoding degree at any given time (if allowed by the RE distribution). Hence, even though the distribution in use is the same as in ANC-RE, the time instants in which the encoding degree is increased differ. Hence, the distribution for Growth Codes almost always returns lower degree packets than what the optimal encoding policy would do, even for the random encounter scenario. This, as we show shortly, leads to substantial differences in terms of performance. These schemes, as well as the different distributions we consider for ANC, are selected to isolate the impact of the adopted degree distribution and of the coding strategy in use, respectively.

In our experiments, interference due to the channel access mechanism, temporal and spatial modifications of the transmitting/receiving radio range, energy consumption due to transmissions and memory usage are all accounted for. Also, for each setting of the involved parameters we repeated a number of experiments so as to get sufficiently tight confidence intervals about our performance measures. In particular, the confidence intervals for the subsequent plots are all within $\pm 10\%$ of the values we show in the graphs. Once again, these intervals are not shown in the graphs for improved readability. In this section, we show results for the case where each sensor generates a single information packet. Hence, we focus on the network-wide dissemination of a single generation of data. Algorithms for more complex scenarios, where nodes generate multiple information packets, are given later in Section 6, where we discuss schemes for *generation and buffer management*.

Before proceeding with the description of the obtained results, we give a short introduction to the TOSSIM TinyOS simulator, which we used to validate our experimental findings and to obtain results for large networks. TOSSIM is the simulator that is distributed with TinyOS. It is used to run TinyOS software thereby emulating the behavior of actual sensor nodes, their timers, the wireless channel, etc. It can be used to test the code to be eventually run on actual sensors, as well as to simulate the behavior of a given protocol in large networks. Packets are transmitted according to a standard CSMA channel access scheme, channel errors can be emulated through any (user defined) channel model and the correctness of received packets is assessed through a CRC check. Errors on acknowledgments, missed start symbols, noise, etc., are also accounted for [18].

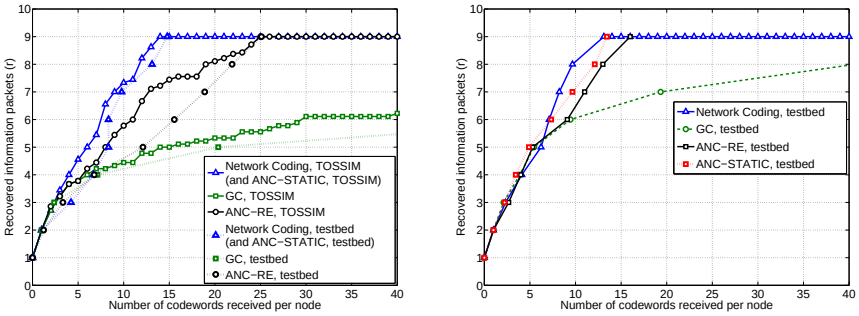

Fig. 4. Average number of packets recovered, r, for 9 sensor nodes placed on line (left) and grid topologies (right). Results are obtained for GC [5], *network coding* [15] and the proposed ANC encoding schemes.

5.1 Small Scale Experiments

In this section we consider a small network of 9 sensor nodes, showing results for grid, line and random topologies. The transmission power is set to the same value for all sensor nodes. In the experiments each node broadcasts either uncoded or coded packets to its neighbors, where coding is executed by means of one of the above algorithms. To access the channel, we use a standard CSMA scheme. At the end of each experiment, all sensors communicate the collected statistics transmitting a *trace file* to the sink node. Trace files contain information about received/transmitted packets as well as their degrees.

9 sensor nodes, line and grid topologies: Positioning the nodes so as to exactly obtain line or grid topologies is difficult in practice, due to the dynamics of the wireless channel. MICAz motes are in fact quite sensitive to antenna positioning and interference. Transmitting and receiving radio ranges change significantly in space and time, see [19]. Given these facts, the actual connectivity graph we get in our experiments does not perfectly match the corresponding graph we would obtain in a simulation with a deterministic channel model. To smooth out part of these variations, we average the results over a sufficient number of experiments.

In Fig. 4 we show the results for line (left plot) and grid (right plot) topologies. For the grid case we considered four neighbors per node. Notably, GC gives unsatisfactory performance in both cases. In particular, the fact that the degree can only be increased adding the node's own information packet to the received packets is insufficient for a proper mixing of the information in static networks. This problem was not observed in [5] as in this paper the authors only focused on extremely dynamic topologies, where the growth codes encoding strategy performs well. *Network coding* performs well and very close to ANC-STATIC. For this reason, only one curve is plotted for both mechanisms. ANC-RE does not perform equally well due to the conservative degree distribution, but it clearly outperforms Growth Codes with the same distribution. Performance in the simulations is slightly higher than in real networks for all algorithms, due to the "more well-behaved" channel mode, but overall TOSSIM simulation data points are

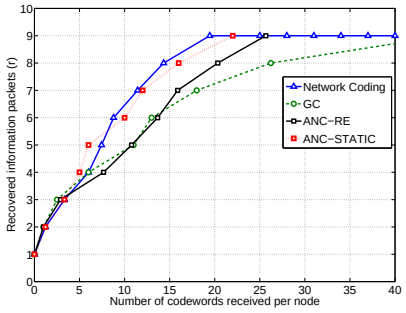

Fig. 5. Average number of packets recovered for *network coding*, ANC and GC. Experimental results for $N = 9$ nodes, random topology.

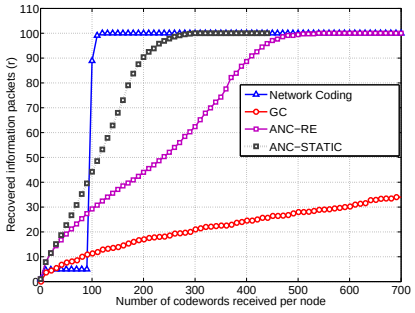

Fig. 6. Average number of packets recovered for *network coding*, ANC and GC. TOSSIM simulations with $N = 100$ nodes on a grid.

reasonably close to the outcomes of our experiments (as seen in the left plot of Fig. 4). A similar agreement between TOSSIM and experiments was found for the results in Fig. 4, plot on the right (and thus they were omitted for the sake of readability). When comparing the two curves, one can see that the less connected the topology is, the more important the high node degree becomes (as done by pure network coding [15]). In the line scenario, pure network coding outperforms all other algorithms. The increase in the number of neighbors in the grid topology helps information dissemination and allows ANC to perform as well as network coding. Growth Codes have worse performance here.

9 sensor nodes, static random topology: Next, we present our experimental results for a simple random topology. This topology is somewhere in between the grid and and the line network and is set up so as to obtain a connected graph. The results for this scenario are given in Fig. 5. A comparison with TOSSIM is not shown due to the difficulties in reproducing the exact random scenario in the simulator. Once again, ANC (ANC-STATIC) performs very close to *network coding*. We can also observe the gap between ANC-STATIC and ANC-RE: the performance of the latter suffers as its degree distribution is not optimal in a static scenario. Finally, GC still gives the worst performance among the considered schemes.

At this point, one might observe that there is little reason for using ANC, as standard *network coding* (i.e., encoding over all available packets) performs very well. However, note that this comes from the fact that nodes gradually accumulate information, and even with full network coding will send out lower degree packets in the beginning, due to the unavailability of further information. This is not the case for larger networks, as we show below.

5.2 Large Scale Experiments

In this section we show results for a network with $N = 100$ nodes. Simulation points are obtained with TOSSIM, the TinyOS simulator. In this simulator, the training

sequence (start symbols) of every packet is transmitted at 10 Kbps, whereas the payload is transmitted at the higher rate of 40 Kbps. Hence, sending a packet of, say, 128 bytes will take about 25.6 ms. Assuming that nodes take turns to transmit, an ideal TDMA would require 2.56 s to schedule the transmission of all nodes. Since we use a random access protocol, we let nodes pick a random transmission time within an interval of $[0, 10]$ s for each transmission slot. In detail, when a node receives an information packet, it stores such packet in its buffer. Hence, before sending a new packet, it waits for the next transmission slot. Transmissions are finally triggered by picking a random transmission time within each time slot. This results in a reasonably low collision rate. We used these settings in all our TOSSIM simulations. The results for 100 nodes are shown in Fig. 6 where we report the performance of ANC, *network coding* and GC. It shall be observed that in this case *network coding* presents the same step-like behavior we discussed previously. This is however detrimental to the performance in terms of *data persistence* as discussed in [5]. In fact, assume that some nodes (or even the whole network) stop working after the dissemination of, e.g., 80 packets so that the network becomes fragmented. In this case, *network coding* would give a recovery rate close to zero. ANC-STATIC, instead, would provide a delivery rate of about 33% (i.e., one third of the packets to be delivered). The same applies if the network generates data faster than it can be transported to the sink nodes. In this case, coding over all packets will prevent the sink node from decoding since it cannot gather enough data for each generation. Also in settings where requests for aggregate information occur at random times and at random nodes in the sensor network, always being able to decode most of the received data is beneficial. Otherwise, the request could only be served after all data has been decoded. Thus, in practical settings a smooth recovery is advisable. Moreover, encoding over *some* packets, as we do in ANC, leads to sparse decoding matrices. These can be inverted with a lower complexity through, e.g., heuristic algorithms, thus leading to lower energy consumption. In addition, with sparse matrices early decoding occurs with higher probability (which is the actual reason for the smooth recovery of ANC).

As a further remark, we note that there is a substantial difference between ANC-STATIC and ANC-RE for a static network with $N = 100$ nodes. This is a further indication of the importance of the selected degree distribution, especially for large networks. Finally, we observe that GC still gives unacceptable performance. Once again, the selected encoding strategy matters and its importance becomes apparent with increasing network size.

6 Handling multiple generations via buffer management

In this section we consider a more general scenario, where nodes generate a large number of observations, at different time instants. We use the concept of *generations* as introduced in Section 3. How packets are subdivided into generations can be decided for example based on spatial or temporal criteria. With the term *spatial generation* we refer to the data generated within the same cluster of nodes, i.e., within a well characterized geographical region (over a certain period of time). With *temporal generation* we refer to the data generated within the same time interval by all the nodes in the network: for the sake of simplicity we consider that the first readings of all sensor nodes belong to

the first generation, the second readings to the second generation and so on. Here, we present a simple buffer management scheme based on temporal generations. This buffer management scheme is included in the real-world implementation used in the previous section.

MICAz nodes have very limited RAM capabilities. Sensor nodes may only be able to hold a single generation in their memory at a time. This is especially true when the number of nodes in the network, N, is large. Moreover, energy consumption is an important consideration in wireless sensor networks and we thus need to devise energy efficient solutions to handle multiple generations with the given memory constraints. In case multiple generations can be stored in the RAM, the following discussion still holds with minimal modifications.

Let N be the number of nodes in the network, which we consider to be connected. Our ANC protocol is used to select the degree distribution for encoding. Moreover, assume that our nodes are processing the first generation of data. As per our ANC algorithm, as nodes receive new codewords (i.e., new packets), these are stored in the main buffer in RAM (one codeword, one row in the buffer). When a transmission opportunity occurs, the rows in the buffer are randomly combined as dictated by the degree distribution in use, following the procedure we described in the previous sections.

For the sake of explanation, consider now the instant in which a given node recovers the first generation. Note that any further packet this node may receive for this generation will be discarded since the packet can not be innovative. Also, the packets currently stored in the buffer (i.e., the recovered generation) can be safely copied to the flash memory as the decoding process is complete for this generation at this specific node. However, these packets are not deleted right away from the buffer, but are rather kept there to further assist the node's neighbors that have not yet recovered the generation.

Upon the complete decoding of the old data, the node can start processing a new generation (i.e., encoding new readings) but the neighbors of this node that are still using the old generation might need additional packets to be able to decode. In order to help these nodes, we propose a novel algorithm referred to here as *cooperative distribution management*. According to this scheme, the node successively overwrites the old data in the RAM, with data from the new generation, from the top row of the buffer to the bottom one. When information belonging to the new generation is received, the node in question stores it in the first row of its buffer. The old packet that previously occupied this position in the buffer is not deleted, but it is rather combined with the packet in the last position of the buffer, as shown in Fig. 7, figure on the left. Upon the reception of subsequent packets belonging to the new generation, the node sequentially stores them in the second, third, etc. position of the buffer, by combining the old packets in these positions with the old packets that are still in the buffer. This combination is done in such a way that the degrees of the old packets that are retained in the buffer resemble, as close as possible, the power of two sequence $1, 2, 4, 8, \ldots$, see Fig. 7, figure in the middle. This specific sequence allows a sufficient variety of degrees among the retained old packets to be able to send out packets of different degrees if necessary. As long as neighbors require packets from the old generation, the node may alternately send out packets from the new and the old generation.

We now consider the nodes which may receive packets belonging to the new generation without, however, having switched to it yet. These nodes may just store these packets in an extra buffer of small size for later use (in case such additional storage space is available). This allows a prompt switch to the new generation, as soon as the recovery of the old generation is complete.

We observe that the above mechanism extensively uses the RAM memory, whereas the flash memory is written only upon the complete recovery of a given generation of data. Limiting the access to the flash is an important consideration as it is characterized by rather long writing times as well as substantial energy consumption (compared to writing to the RAM).

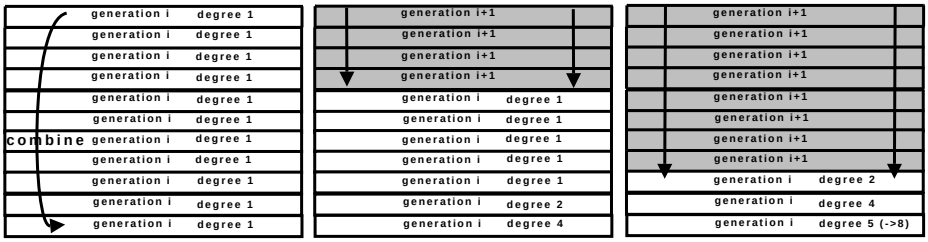

Fig. 7. Example of buffer management. Generation i is fully recovered, the new generation $i+1$ is being processed. Packets belonging to generation i are still retained, through their linear combination, upon the reception of packets pertaining to the new generation.

We intend to investigate further buffer management schemes in future work. In particular, we believe that a more flexible scheme that is able to handle multiple generations simultaneously, while adhering to the same memory constraints, may improve the performance. Writing coded data to the flash is undesirable since it needs to be rewritten once it has be decoded. However, it is not necessary to only write full generations to the flash in a single pass. As partial data is recovered, it can be written to the flash in case it is not needed very often for the decoding of further packets for a given generation. Note that reading from the flash is substantially less expensive than writing to it. We also intend to explore spatial generations, coded over the packets of a specific region.

7 Conclusions

In this paper we proposed a novel network coding-based algorithm with adaptive degree distribution, with the aim of achieving a high degree of data persistence in static and mobile networks. We believe this is an important first step towards a practical self-adaptive coding algorithm for sensor networks. We provide insights into the relationship between degree distributions and network mobility. We also characterized the performance of different encoding schemes and related degree distributions through simulations and experiments on real nodes.

A more thorough analysis on how to adapt the degree distribution to the specific dynamics of the network (and how to “measure” these dynamics) is still necessary and

kept under study. Moreover, buffer management is another important avenue for future research, which we briefly introduced here and which needs to be addressed further to successfully distribute multiple snapshots of data in large networks.

References

1. R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.
2. T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On Randomized Network Coding," in *41st Annual Allerton Conference on Communication Control and Computing*, Monticello, IL, US, Oct. 2003.
3. S. Acedanski, S. Deb, M. Medard, and R. Koetter, "How good is random linear coding based distributed networked storage?" in *NetCod*, Riva Del Garda, Italy, Apr. 2005.
4. S. Deb and M. Medard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," in *42nd Annual Allerton Conference on Communication Control and Computing*, Monticello, IL, Oct. 2004.
5. A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence," in *ACM SIGCOMM*, Pisa, Italy, Sept. 2006.
6. M. Luby, "LT Codes," in *43rd Ann. Symp. on Fundations of Computer Science*, Vancouver, Canada, Nov. 2002.
7. J. Liu, Z. Liu, D. Towsley, and C. H. Xia, "Maximizing the data utility of a data archiving and querying system through joint coding and scheduling," in *IPSN 2007, Cambridge, MA, US*, Apr. 2007.
8. D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi, "Network coding strategies for data persistence in static and mobile sensor networks," in *WNC3*, Limassol, Cyprus, Apr. 2007.
9. R. Motwani and P. Raghavan, *Randomized Algorithms*. New York, NY, US: Cambridge University Press, 1995.
10. S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1982.
11. Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *INFOCOM 2007, Anchorage, AK, US*, May 2007.
12. A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized Erasure Codes for Distributed Networked Storage," *IEEE/ACM Trans. on Networking*, vol. 52, no. 6, pp. 2809–2816, June 2006.
13. ——, "Distributed Fountain Codes for Networked Storage," in *IEEE ICASSP*, Toulouse, France, May 2006.
14. P. A. Chou, T. Wu, and K. Jain, "Practical network coding," in *41st Annual Allerton Conference on Communication Control and Computing*, Monticello, IL, US, Oct. 2003.
15. J. Widmer, C. Fragouli, and J.-Y. LeBoudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *NetCod*, Apr. 2005.
16. E. Fasolo, J. Widmer, M. Rossi, and M. Zorzi, "A Proactive Network Coding Strategy for Pervasive Wireless Networking," in *IEEE GLOBECOM*, Washington, DC, US, Nov. 2007.
17. "CC2420 data sheet." [Online]. Available: http://www.ti.com/
18. P. Levis and N. Lee, *TOSSIM: A Simulator for TinyOS Networks*, June 26, 2003.
19. "Tinyos community forum." [Online]. Available: www.tinyos.net