Evaluation of Network Coding Techniques for a Sniper Detection Application

Lorenzo Keller Abdulkadir Karaagac Christina Fragouli Katerina Argyraki
{lorenzo.keller,abdulkadir.karaagac,christina.fragouli,katerina.argyraki}@epfl.ch
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

***Abstract*—This paper experimentally studies the reliability and delay of flooding based multicast protocols for a sniper detection application. In particular using an emulator it studies under which conditions protocols based on network coding deliver performance improvements compared to classic flooding. It then presents an implementation of such protocols on mobile phones.**

I. Introduction

In this paper we study the performance gains of network coding when applied to multicast for a particular application: sniper detection.

In this application mobile sensors, carried by soldiers, measure the time at which they hear a shot and exchange these measurements. Thanks to signal processing techniques [12] the nodes can then infer the position of the shooter.

We focus on the communication aspect of the system. Communication happens over wireless and therefore is limited in range and subject to noise. Moreover when nodes transmit at the same time their communication can collide and in this case packets are potentially lost.

A key characteristic of our setup when compared to other studies on multicast for mobile ad-hoc networks is that, instead of having at a given time one source and a set of receivers, all nodes are both generating and receiving traffic at the same time.

In our application two performance criteria are important: reliability and delay of data exchange. Every node must receive within a small amount of time the measurements of at least a subset of the other nodes.

In this paper we perform an experimental evaluation of already proposed coding approaches. Our contribution is to assess the impact of these schemes on delay and reliability. We also study how different parameters such as transmission rate, network topology and computing power of the nodes influence delay and reliability of these protocols.

The questions we try to answer is whether coding can improve delay and reliability regardless of the particular network parameters. In the rest of the paper we will show that coding indeed significantly increases reliability and under some conditions also reduces delay.

The paper is organized as follows: in Section II we discuss relevant work in existing literature, in Section III we describe our setup and in Section IV we introduce the coding schemes under study. In Section V we present the results of our tests and in Section VI we conclude.

This work was funded by ArmaSuisse Wissenschaft + Technologie (W+T), Project no. 8003413832.

II. Related work

In the past decades many multicast protocols for mobile ad-hoc wireless networks have been proposed [2], [13]. The main challenge in this setup are the frequent changes in the network topology that require suitable protocols.

The solutions that have been proposed can be classified in three categories: efficient tree construction protocols (e.g. MAODV [11]), mesh based protocols (e.g. ODMRP [7]) and flooding based protocols (e.g. SMF [9]). The first approach tries to quickly adapt a multicast tree to network topology changes, in the second approach nodes do not simply build a tree but instead build a mesh and therefore link failures can be better tolerated. In the third approach, flooding, each node that receives a multicast message retransmits it. To improve the rate of flooding and reduce network contention protocols can leverage knowledge of their neighborhood to reduce the number of transmitted packets (e.g. E-CDS [10], S-MPR [1] or probabilistic flooding [13]).

In this paper we don't propose a new approach to multicast: instead we propose coding schemes to be used in conjunction with flooding. We work with flooding because this approach doesn't require to proactively maintain information about the network. In our setup this would be extremely costly in terms of energy since events are rare. For the same reason we will also not use any rate reduction technique.

The performance of network coding when used in conjunction to multicast has been studied in other papers. Most of them look at the problem of a single source transmitting to many receivers. Some papers study our setup but either use rate as performance metric [8], [6] or study rate delay trade-offs from a theoretical point of view [14].

III. Problem setup

We now proceed to formalize our setup. A set of N (mobile) sensor nodes σ_1, ..., σ_N distributed in a given area have to exchange their observation $o_1, ..., o_N \in \mathcal{A}$ about an event that happens at time $t = 0$. Sensors σ_i is ready to communicate its observation of the event to the other sensors at a time t_i that can depend on the distance of the sensor from the event and the time necessary for processing the raw sensed data.

Data is exchanged by the nodes via packet based radio communications; the sensors run a communication protocol that defines when and what to transmit. The radio supports communication at a certain data rate R. We don't assume that nodes have directional antennas and or that they know their relative positions.

In our study we assume that the radios are using carrier sense multiple access (CSMA) with back-off to decide when to transmit as specified in the 802.11 standard. Therefore each node before sending checks if another station is already transmitting. If so it backs-off its transmission until the channel becomes free. When the channel is free the node performs an additional random back-off to avoid collisions with other nodes that were also waiting to send.

We assume that node movement is such that during the data dissemination for an event nodes can be considered static while from one event to the next the network topology is likely to have completely changed. This assumption models many situations in which the timescale at which events happen (minutes or hours) is much larger than the timescale at which data is exchanged (milliseconds or seconds).

We study the performance of different protocols in respect of two quality metrics: delay and reliability. Delay measures the average time at which every node receives the first, second, etc. observation from its peers. Reliability measures the average probability for a given node to receive 1, 2, etc. observations from its peers.

IV. Description of protocols

In this section we describe four coding schemes for flooding, one of them is purely theoretical and will be used to give bounds on the performance of any other protocol while four of them are practical and can be used to perform data dissemination in a real network. In Section V we will analyze their performance in different setups.

A. Classic flooding

In classic flooding source σ_i transmits its observation by simply sending a packet (i, o_i). Every node that receives the packet checks if it has already received o_i in another packet. If not it retransmits the packet. All the recipients of this new message repeat the same procedure. At some point all sent packets will be received by nodes that have already seen o_i and therefore forwarding will stop.

In this protocol every observation is potentially retransmitted N times. If there is collisions or poor radio conditions however, some observations may never be received by some nodes and therefore it is possible that they are retransmitted less than N times. In particular if the first transmission of an observation is lost then nobody will ever retransmit it.

B. Aggregation flooding

In classic flooding any given packet contains exactly one observation and therefore some packet transmissions are useless for some nodes. In particular if a node is only missing the observation of a specific source all the packets containing any other observation are useless.

In aggregation flooding every node transmits all observations it has heard up to now in every packet it sends, and not only the new observation that triggered the packet transmission. This means that the "usefulness" for the recipients of the packets is increased.

Implementing such a protocol is not always possible. Packets containing multiple observations grow in size and the more sources are present, the larger the packets will be. When the packet size exceeds the maximal packet size allowed by the link layer in use it will not be possible to send them.

In our comparison we will therefore assume that there exists an ideal algorithm that is able to compress the size of packet containing multiple observations to the size of packet containing only one observation. We will implement the protocol by just sending the source identifiers in the packets.

The performance of this hypothetical protocol will give us a bound to any flooding based protocol.

C. Random network coding

In this protocol, based on the ideas proposed in [3], every packet p instead of containing a source identifier and the corresponding observation it contains a *coding vector* h_p , a vector of length N over the finite field $\mathbb{F}_q$, and a payload d_p, a vector over the same finite field of length $K = \lceil \log_q(|\mathcal{A}|) \rceil$.

Each source σ_i maps its observation o_i to a vector $x_i \in \mathbb{F}_q^K$ and creates a coding vector $c_i \in \mathbb{F}_q^N$ where $(c_i)_j = 0$ if $j \neq i$ and 1 otherwise. It then sends the two in a packet as (c_i, x_i).

Upon reception of a packet $p = (h_p, d_p)$ every node checks if the received coding vector h_p is linearly independent from the coding vectors previously received. If so if creates a random linear combination of the packets received up to now and send it. The packet sent will have therefore the form :

$$\left(\sum_i \alpha_i \cdot h_i, \sum \alpha_i \cdot d_i \right)$$

for some randomly chosen α_i. It is easy to see that all sent packets can be equivalently expressed as:

$$\left(\sum_{i=1}^{N} \beta_i \cdot c_i, \sum_{i=1}^{N} \beta_i \cdot d_i \right)$$

By appropriately combining received packets the nodes can reconstruct the original $x_1, ..., x_N$. In particular they can do so by performing Gaussian elimination on the matrix:

$$\left(\begin{array}{c|c} h_{p_1} & d_{p_1} \\ \vdots & \vdots \\ h_{p_m} & d_{p_m} \end{array} \right).$$

Notice that in this protocol a packet can potentially be used to recover different sources. In this sense it is therefore similar to aggregation. However a big difference is that to decode a packet which is a linear combination of k sources the receiver may need to first receive up to k packets. In aggregation on the contrary packets can be decoded individually.

D. Opportunistic coding

This protocol, based on ideas proposed in [5], is similar to random network coding but nodes instead of randomly picking the coefficients they carefully choose them in a way that allows all the receivers to decode the received packets immediately. This is done by ensuring that if a packet is a linear combination of M observations then every node that receives it already knows at least $M-1$ of the contained observations.

In order to be able to build the packets each node maintains a list of neighbors and for every neighbor a list of observations that the neighbor linearly combined in its packets up to now. We know that since every packet has been sent such that the neighbors could decode it, every neighbor can only linearly combine observations that it has already decoded and therefore this list can only contain observations already decoded by the neighbor.

As in random network coding a packet can be useful to more receivers than in classic coding. Opportunistic coding also solves the problem of having packets that need to wait to be decoded. However depending on the structure of the network the number of coding opportunities can be very small. In those situations opportunistic coding behaves exactly like classic flooding.

Critical for this protocol is the knowledge of the neighborhood of each node. Proactively maintaining this knowledge is too expensive in our application. Our implementation therefore collects this information passively during the packet dissemination. Each node at the beginning of the dissemination has an empty neighbor table and doesn't perform any coding. Every time it hears a packet it adds the transmitting node to his table and start coding its packets. Neighbor table entries expire after some time of inactivity of the neighbor. This can lead to errors in the construction of packets but we observe that they are quite rare.

E. Limited coding

This protocol is an hybrid between opportunistic and random network coding. In this case as in opportunistic coding only packets that have already been decoded are linearly combined in transmitted packets. The difference is that this protocol takes no precautions to make sure that the neighbors can decode the packets immediately. This protocol has the advantage compared to random network coding that it sends linear combination that are potentially easier to decode since every packet contains at most one additional source compared to what was sent up to now by the node and therefore it can potentially reduce the delay. The main drawback of this protocol is that a new packet is forwarded by a given node only when a new observation is decoded. This means that when a node cannot decode most of its received packets, it will forward very little information and therefore negatively impact the network performance.

V. Experimental results

In this section we present the performance measurements for the protocols described in Section IV under different conditions and we explain their performance. In our experiments we vary the connectivity of nodes, their processing power and the transmission rate.

We test the different protocols in the networks described in Figure 1: two one-hop networks (a and e), a two-hops network, a network with a bottleneck and a N-hops network. In the one-hop networks all nodes are within the reception range of all the others. In the two-hop network, every node can communicate to every other node by relaying through at most one other node. In the bottleneck topology every node can reach every other node through at most 3 hops, the network is however fully connected only thanks to two nodes that form a bottleneck. Finally in the N-hop network nodes are arranged in a line and can communicate only with their immediate neighbors.

The test consists in running multiple times the data dissemination. We assume the nodes are measuring the time of arrival of a sound produced periodically at a fixed position. Every node σ_i knows the times at which the sound is emitted and it knows its distance from the source, it can therefore compute the time t_i at which it can generate its observation. The observation is then disseminated accordingly to the protocol in use.

To measure the performance we first measure for every round and node the arrival time of each packet from the moment in which the event happened. To compute the average delay we individually average the arrival time of the first, second, etc. packet over all rounds and nodes. To compute the reliability we compute the percentage of rounds every node received one, two, etc. packets and then we compute the average over all nodes.

We perform the test using an emulated and a real testbed. We use the emulated environment to test the performance of the protocols at a rate $R = 16$ kbps. This rate is typical for tactical radios and in sensor networks. We also run the protocols in a real testbed composed by 6 HTC WildFire cellphones running Android 2.1 and forming a 802.11b adhoc network. This allows us to test the protocols under real network conditions and gives an idea of the applicability of the conclusions found for tactical radios to radio technologies in widespread use. This also allows us to test the protocols with limited computational resources.

The tests in the emulated environment are done using eMANE [4]. This emulator creates on a normal Linux computer a virtual interface for each emulated node and runs in real time MAC and PHY layer as if the interfaces were real network cards. We used the 802.11 radio model provided with the emulator. We changed the code to support the (non-standard) transmission rate of 16 kbps. We also enabled the use of Wireless Multimedia Extensions (802.11e) in order to increase contention widows sizes and therefore reduce the amount of packets that get lost due to collisions. To configure link gains between nodes we used a Log-distance path-loss model. We also used the SNIR to BER curves provided with the emulator for 1 Mbps communications with 802.11b radios. The protocols are implemented in Java and

packet transmissions at the sources are triggered by the system clock which is shared among all nodes. The processor of the computer running the tests is an 3Ghz Intel Xeon.

In the real testbed we use the same Java code used in the emulation tests packaged as an Android application. In order to be able to set the phone radios in ad-hoc mode we had to modify the phone firmware. Synchronized triggering of packet generation on the phones is more tricky to achieve than in the emulated testbed. Phones need to be synchronized. To do so they are running a custom time synchronization protocol that synchronizes them within approximately 5 ms.

The protocols are implemented on top of UDP. To be able to receive the packets at multiple receivers we send them to an IP multicast group to which all the nodes are registered. This implies that packets are sent as link layer broadcasts and therefore are not acknowledged as it usually happens on 802.11 data frames.

To implement limited and random network coding we use operation over $\mathbb{F}_{2^4}$ in the emulator and $\mathbb{F}_{2^8}$ in the phones. Each entry of payload and coding vectors requires half (respectively one) byte to be transmitted. Finite field operations are implemented using table lookups, addition and subtraction when using $\mathbb{F}_{2^8}$ are implemented directly with xor operations. The finite field is chosen using three criteria: it should be sufficiently large to ensure correct operation, as small as possible to reduce coding headers overhead and should allow fast computations. For our phone implementation in particular we noticed that using one byte per field element is the best trade-off between these three criteria. On the emulator we used a smaller field size since the cost of manipulating vectors over $\mathbb{F}_{2^4}$ is not so expensive as on cellphones and we have smaller headers.

To implement opportunistic coding we use operations over $\mathbb{F}_2$. Each entry of the coding vectors requires therefore 1 bit to be transmitted. We choose which observations to code iteratively by selecting one by one the one that will be useful to the largest number of neighbors and that will not cause them not to decode. If due to errors in the neighbor tables a node receives a packet it cannot decode it will discard it. We decided to use this approach after verifying that such packets are very rare.

A. Emulated network

In this section we discuss the performance of different coding schemes when the data rate is small. This is the case in sensor networks and tactical radios. We also discuss which conclusions are also valid for higher data rates.

The observation sent by the nodes has length 51 bytes. Coding vectors have length 4 bytes in random network coding and 1 byte in opportunistic coding. Identifiers in classic flooding have length 1 byte. The coding schemes, in addition to 802.11, IP and UDP headers have 4 additional bytes that store a sequence number that would be used in case of concurrent dissemination of observation about multiple events. Sequence numbers are loosely synchronized by changing the local sequence number in case a packet with a larger sequence number is received.

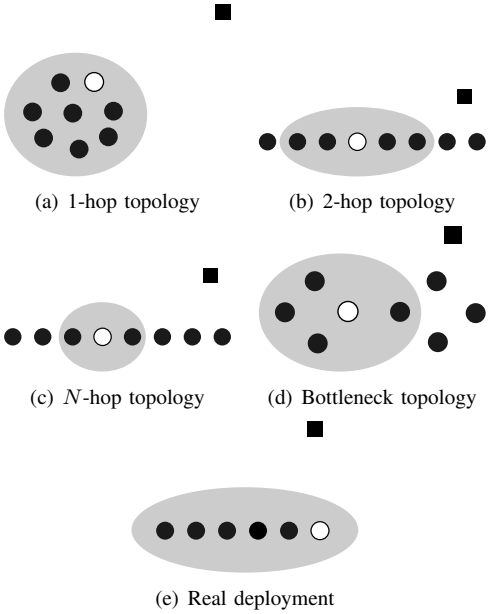

Fig. 1. Diagram of the deployments used for testing. Circles are nodes, the square is the sound source. The gray area around the highlighted node indicates the connectivity.

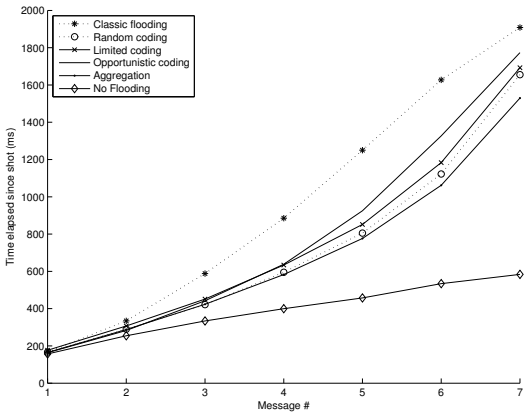

Fig. 2. Average delay in an emulated 1-hop network ($R = 16$ kbps, $N = 8$)

In this section we present results where we run each protocol for 70 events.

a) 1-hop network: Figures 2 and 3 illustrate the performance of the different protocols discussed in Section IV. In addition to those protocols the figures also illustrate the performance when not using any form of flooding.

First observe the performance when no flooding is performed. We can see that the delay is much lower than all the other protocols. This due to two factors. First less packets are sent and therefore less contention happens on the wireless medium. The second reason is due to the fact that less packets are received and therefore the average delay is computed only on "fast" packets. This is an important behavior of the performance metric we use that should always be considered

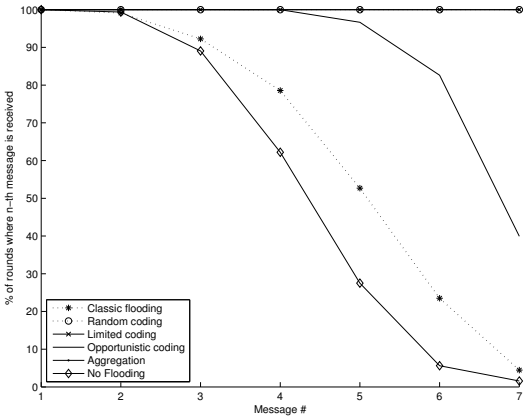

Fig. 3. Reliability in an emulated 1-hop network ($R = 16$ kbps, $N = 8$)

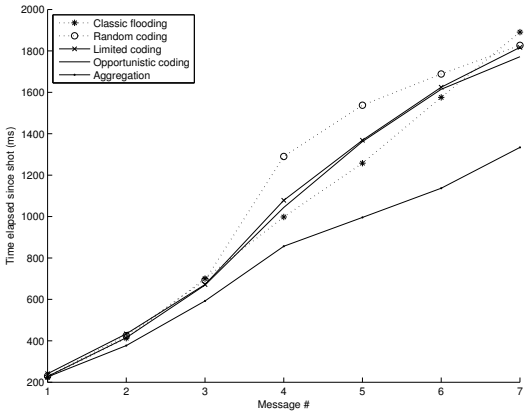

Fig. 4. Average delay in an emulated 2-hops network ($R = 16$ kbps, $N = 8$)

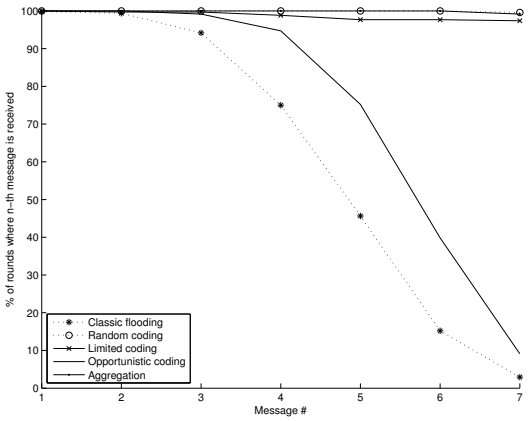

Fig. 5. Reliability in an emulated 2-hops network ($R = 16$ kbps, $N = 8$)

when studying the graphs presented in this paper.

We see that classic flooding has the largest delay. It is interesting to observe that opportunistic coding performs better than classic flooding both in delay and reliability. In this setup packets are lost only if there is a collision, and when a collision occur nobody will receive the packet. This means that each node can have at most one observation not received by all other nodes and this observation is its own. It will be able to code it only when it is forwarding an observation that it knows everybody else has already received. Since this happens rarely we would expect the performance of opportunistic coding to be approximately the same as classic flooding. In our implementation however neighbor tables are not correct at the beginning of the dissemination and therefore some more coding opportunities arise. We ran the same tests with static neighbor tables and we observed that in that case opportunistic coding and classic flooding perform approximately the same.

In this setup coding performs particularly well. Its delay is near what is experienced by aggregation and its reliability is also high. The high reliability is easily explained by the fact that every node in this setup receives a large amount of packets, much bigger than N. Therefore it is likely that every node can decode all the source packets. In particular coding protects from the problem experienced by classic flooding when the first transmission of an observation is lost: nobody will ever receive it because nobody will transmit it again. In coding every packet sent by a source contains information about its own observation and therefore it's much less likely that this will be lost.

Coding has also a surprisingly good delay performance. One would expect that coding across packets would introduce some delay. In this setup however the packets sent are easy to decode. Indeed since every packet is either received by everybody or by nobody (in case of collision) every node has at most one observation unknown to other nodes, namely its own observation. Packets are therefore always decodable by all recipients.

In this topology limited coding doesn't significantly differ from full coding. We will show that in other topologies it instead improves performance.

We ran the same experiment at a higher data rate (1 Mbps). When using the same packet size we observed that almost no packet losses are experienced because almost no collisions occur. Delays are also very similar since all nodes receive the first communication. When we increased the packet size to 1000 byte we could observe a behavior similar to what happens at 16 kbps.

b) 2-hops network: Figure 4 and 5 show the performance of the protocols in a network with diameter 2. In this setup no protocol can approximate the performance of ideal aggregation. All other protocols have the same performance for the first three packets, which are from nodes in their immediate neighborhood. For later observations coding has a higher delay than the other protocols. This can be explained by the fact that in this topology packets are not always lost by all nodes in every case of collision and therefore packets cannot always be decoded immediately as it was happening in the previous scenario.

Regarding reliability we see that coding still performs as well as aggregation, while the performance of classic flooding and opportunistic coding decreases. This is due to the fact that some observations must be successfully transmitted twice

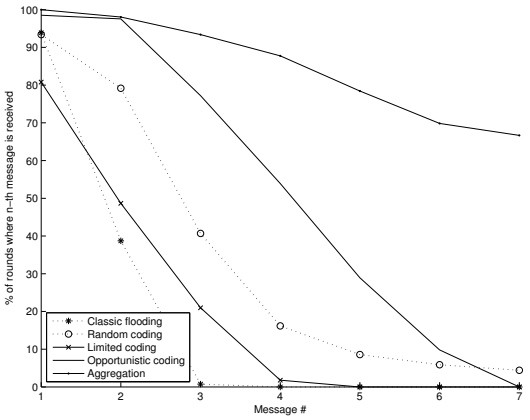

Fig. 6. Reliability in an emulated N-hops network ($R = 16$ kbps, $N = 8$)

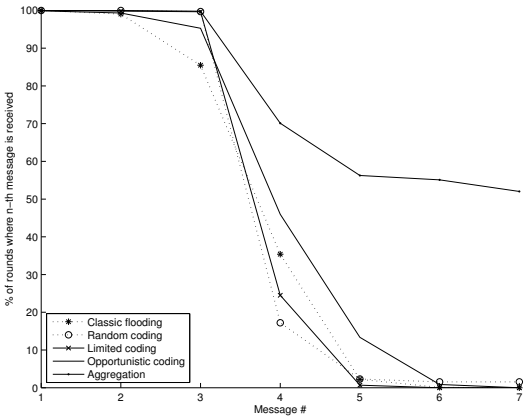

Fig. 7. Reliability in an emulated bottleneck network ($R = 16$ kbps, $N = 8$)

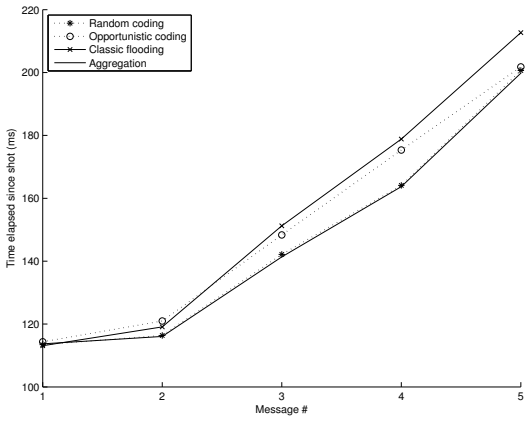

Fig. 8. Average delay in the network composed by six colocated phones ($R = 1$ Mbps, $N = 6$)

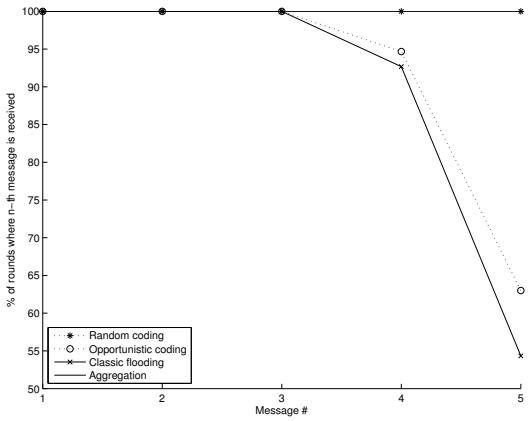

Fig. 9. Reliability in the network composed by six colocated phones ($R = 1$ Mbps, $N = 6$)

before being received by a given node. Opportunistic coding performs better than classic flooding only because at the start of the dissemination neighbor tables are empty. We tested the same protocol with static neighbor tables and the performance of opportunistic coding is the same as classic flooding. The large number of neighbor of each node reduces the number of coding opportunities.

In this network limited coding helps, we can see that its reliability is similar to full coding while its delay is comparable to classic flooding.

Our tests at 1 Mbps with packets of length 1000 bytes show a similar reliability of protocols. Delay of classic flooding and the coded protocols are approximately the same.

c) Bottleneck network and N-hop network: Figure 6 and 7 show the reliability of the different protocols in a network with a bottleneck and in a line network. We can see that in these setups fixed rate protocols discussed in this paper are not adequate. Indeed none of them can ensure a proper dissemination of the data throughout the network. The main reason of these losses is that transmissions are often lost due to the hidden terminal effect. Since the protocols use broadcast communications RTS/CTS cannot be used to mitigate the problem. Notice also that neither end to end retransmission nor FEC here would be sufficient to increase reliability. Since the probability of receiving some of the packets is so low, end to end erasure protection will require an extremely large amount of resources. A much better approach is to perform retransmissions at every node.

The advantage of opportunistic coding compared to classic flooding in the N-hop topology disappears when we increase the rate and packet length. This is due to the fact that timing of packet arrivals is critical to the performance of the protocols. At 1 Mbps classic flooding performs similarly to opportunistic flooding (therefore outperforming random coding).

For the bottleneck topology when we performed experiments at 1 Mbps and packets of length 1000 bytes we observed a behavior similar to what happens at 16 kbps.

B. 1-hop Network of Cellphones

In this setup we test the performance of the different protocols when they are running on cellphones.

There are mainly two differences with the previous section: different rate and different computing power. Phones are transmitting at 1 Mbps and their CPU is a Qualcomm MSM7225 at 528 MHz, much slower than the Xeon processor used in

the emulation.

In this setup decoding and encoding overhead must be carefully handled. We had to optimize the decoding and encoding process used in the protocols to make sure that delay due to them doesn't dominate packet deliver delay. For example we had to reduce at minimum the memory allocations during the coding and decoding process to avoid garbage collection to happen too frequently. Additionally we perform as many operations as possible in place on the received vectors. Finally we also avoid table lookups for additions and multiplication.

At 1 Mbps, dissemination of an observation can be over before the next observation is inserted in the network. Indeed to observe a significant number of collisions we had to increase the observation length to 1000 bytes. Notice that since 802.11 mandates a number slots in the contention window that doesn't depend on the rate, collisions have the same likelihood at 1 Mbps as at 16 kbps provided that there is more than one node waiting to send.

Figure 8 and 9 show the performance of the protocols in this setup. Notice that for aggregation, in this setup, only packets identifiers are sent therefore the packet length is much smaller than any other protocol.

We see that again coding is a good choice to guarantee high reliability. We can also observe that coding and decoding delay do not impact the performance of the coded protocols.

VI. Conclusion

In this paper we empirically studied the performance of different coding schemes for data dissemination. We showed that they can both reduce delay and improve reliability both through simulations and by running the protocols on low-end smart-phones.

Coding can reduce delay in networks where each node is within the range of the all other nodes. In topologies where the hidden terminal problem is present coding loses its advantage but the appropriate coding strategy has a delay comparable to classic flooding.

Coding also improves significantly reliability when nodes are sufficiently well connected. In situations where classic flooding for more than 50% of the rounds doesn't deliver 40% of the packets coded flooding can achieve more than 99% reliability for all packets.

References

[1] T. Clausen and P. Jacquet. Optimized link state routing protocol. RFC 3626. Technical report, 2003.

[2] C. de Morais Cordeiro, H. Gossain, and D.P. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *Network, IEEE*, 17(1):52 – 59, 2003.

[3] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[4] N. Ivanic, B. Rivera, and B. Adamson. Mobile ad hoc network emulation environment. In *IEEE Conference on Military Communications (MILCOM)*, 2009.

[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: Practical wireless network coding. In *ACM SIGCOMM*, 2006.

[6] T. Kunz and L. Li. Broadcasting in multihop mobile tactical networks: to network code or not. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2010.

[7] Sung-Ju Lee, M. Gerla, and Ching-Chuan Chiang. On-demand multicast routing protocol. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 1999.

[8] Weifa Liang, R. Brent, Yinlong Xu, and Qingshan Wang. Minimum-energy all-to-all multicasting in wireless ad hoc networks. *Wireless Communications, IEEE Transactions on*, 8(11):5490 –5499, 2009.

[9] J.P. Macker, J. Dean, and W. Chao. Simplified multicast forwarding in mobile ad hoc networks. In *IEEE Conference on Military Communications (MILCOM)*, 2004.

[10] R. Ogier. MANET extension of OSPF using CDS flooding. In *Proceedings of the 62nd IETF*, 2005.

[11] E. Royer and C. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[12] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *ACM International Conference on Embedded networked Sensor Systems (SenSys)*, 2004.

[13] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2002.

[14] Chi Zhang, Yuguang Fang, and Xiaoyan Zhu. Throughput-delay trade-offs in large-scale manets with network coding. In *INFOCOM 2009, IEEE*, 2009.