Draft version of paper accepted for publication in IEEE/ACM Transactions on Networking, April 2008.

# Efficient Broadcasting using Network Coding

Christina Fragouli, Jörg Widmer, and Jean-Yves Le Boudec

***Abstract*— We consider the problem of broadcasting in an ad-hoc wireless network, where all nodes of the network are sources that want to transmit information to all other nodes. Our figure of merit is energy efficiency, a critical design parameter for wireless networks since it directly affects battery life and thus network lifetime. We prove that applying ideas from network coding allows to realize significant benefits in terms of energy efficiency for the problem of broadcasting, and propose very simple algorithms that allow to realize these benefits in practice. In particular, our theoretical analysis shows that network coding improves performance by a constant factor in fixed networks. We calculate this factor exactly for some canonical configurations. We then show that in networks where the topology dynamically changes, for example due to mobility, and where operations are restricted to simple distributed algorithms, network coding can offer improvements of a factor of $\log n$, where $n$ is the number of nodes in the network. We use the insights gained from the theoretical analysis to propose low-complexity distributed algorithms for realistic wireless ad-hoc scenarios, discuss a number of practical considerations, and evaluate our algorithms through packet level simulation.**

## I. Introduction

Network coding is an area that has emerged in 2000 [1], [2], and has since then attracted an increasing interest, as it promises to have a significant impact on both the theory and practice of networks. We can broadly define network coding as allowing intermediate nodes in a network to not only forward but also combine the incoming independent information flows. Combining independent data streams allows to better tailor the information flow to the network environment and accommodate the demands of specific traffic patterns.

The first paradigm that illustrated the usefulness of network coding established throughput benefits when multicasting over error-free links. Since then, we have realized that we can get benefits not only in terms of throughput, but also in terms of complexity, scalability, and security. These benefits are possible not only in the case of multicasting, but also for other network traffic configurations, such as multiple unicast sessions. Moreover, they are not restricted to error-free communication networks, but can also be applied to sensor networks, peer-to-peer systems, and optical networks. It is in fact advocated that the first applications where network coding will have an impact will be peer-to-peer and ad-hoc wireless networks, as these are environments that offer more freedom in terms of protocol design choices and where information inherently propagates in a distributed manner. For example, ongoing projects investigate the application of network coding ideas to content distribution [3].

In this paper we show that use of ideas from network coding allows to realize energy savings when broadcasting in wireless ad-hoc networks. By broadcasting we refer to the problem where each node is a source that wants to transmit information to all other nodes. Such all-to-all communication is traditionally used during discovery phases, for example by routing protocols; more recently, it has been described as a key mechanism for application layer communication in intermittently connected ad-hoc networks [4]. Moreover, it is directly related to the problem of content distribution. The problem of broadcasting is interesting not only because it abstracts diverse practical applications, but also because this is a situation where information mixing is clearly beneficial and where we thus expect network coding to offer benefits.

Energy efficiency directly affects battery life and thus is a critical design parameter for wireless ad-hoc networks. Optimizing broadcasting for energy efficiency has been extensively studied during the last decade. Applying network coding for wireless applications in general has also been proposed and investigated in the more recent literature. We review both these veins of related work in Section II.

Our interest is on the specific problem of broadcasting, that is, all-to-all communication. As figure of merit we use energy efficiency, calculated as the number of transmissions required for an information unit to reach all nodes in the network. For this specific problem, we derive exact theoretical characterizations of the expected benefits, as well as develop algorithms that allow to realize these benefits in a distributed manner. Our novel contributions can be summarized as follows.

We start by examining fixed networks, that is, networks where the topology and link capacities do not change over time. In this case we show that network coding can at most offer a constant factor of benefits in terms of energy efficiency. We exactly calculate these benefits for a number of canonical configurations, such as the circular network and the square grid network. The same analysis directly extends to all lattices.

Our ultimate goal is not only to investigate possible benefits network coding can offer, but in particular, to deploy network coding ideas in a practical setting and propose simple algorithms that allow to realize the theoretically expected performance. Towards this goal we then focus our attention to decentralized operation, and examine benefits in terms of energy efficiency that use of network coding can bring to this problem without idealized centralized scheduling. We propose distributed algorithms that can be deployed in real networks, and examine different aspects of the proposed system in detail, that are related to and motivated by practical considerations. For example, we investigate the effect of transmission range, the choice of a forwarding factor, possible trade-offs from restricted complexity and memory capabilities, and limited generation sizes. We evaluate the performance of the proposed algorithms through simulation over random networks.

We then examine networks where the configuration dynamically changes, due to nodes moving, turning on and off, roaming out of range, etc. We focus our attention to very simple decentralized distributed algorithms, where nodes do not know the identity of their neighbors. Our motivation is that, in a dynamically changing environment, such updates are costly. Such a configuration is provided for example by very sparse mobile networks where intermittent connectivity is common. Delay-tolerant networking (DTN) architectures [5] are designed to cope with the adverse conditions found in such environments, and existing algorithms in this area are usually based on some form of flooding.

For a number of examples of dynamically changing topologies, we reduce the problem of energy efficiency to simple variations of the coupon collector problem (see for example [6]). This problem was examined in conjunction with network coding in [7], and it was shown network coding can offer benefits that increase as $\log n$ where $n$ is the number of nodes in the network. We thus establish that $\log n$ benefits are also possible in our setting. Simulation results over realistic networks and mobility models demonstrate that our proposed algorithms allow to realize these benefits in practice.

Thus, we conclude that significant benefits of network coding in a wireless environment might manifest in situations where the network operational complexity is restricted.

The paper is organized as follows. Section II provides a review of related work. Section III presents our problem formulation and briefly reviews basic ideas in network coding. In Section IV, we calculate energy efficiency benefits use of network coding can offer over fixed networks, while in Section V, we propose algorithms to realize these benefits in practical networks. We evaluate the proposed algorithms through simulation in Section VI. Section VII examines dynamically changing networks, establishes possible energy efficiency benefits and demonstrates that they can be realized in practice. Section VIII concludes the paper.

## II. Work related to the problem of wireless broadcasting

There exist two main bodies of theoretical work in wireless broadcasting, that do not employ the network coding approach. In both cases, the emphasis is in minimizing the *speed of information dissemination*, which is expressed in terms of rounds of transmissions, with multiple nodes communicating in parallel during each round. We present the results as related to our specific problem. We then briefly review proposed algorithms for flooding in practical networks. Finally, we review related results in the network coding literature, and discuss how our work is positioned in this framework.

### *A. Epidemic Algorithms for Rumor Spreading*

This work focuses on networks represented as graphs, and distributed algorithms, where nodes do not have information about the nodes they are communicating with. At each round, each node randomly chooses a communication partner among the nodes that are connected to it through an edge, and either “pushes” or “pulls” information from it (see for example [8], [9]). Results in the literature establish that $\mathcal{O}(n \log n)$ rounds are required to disseminate the messages. Work in [7] showed that using network coding over a complete graph requires $\mathcal{O}(n)$ rounds, and more recently, a characterization of network coding benefits over arbitrary graphs was provided in [10].

### *B. Broadcasting in Radio Communication Networks*

In this body of work the wireless environment is modeled as a graph, where, when a node transmits a message, it is received by all its neighbors, and where a node successfully receives information if and only if exactly one of its neighbors is transmitting. Again transmissions are divided into rounds, where in each round a subset of the nodes transmits, in a way scheduled to minimize conflicts and maximize information spreading. The goal is to disseminate the information in the smallest number of rounds. Both centralized and decentralized algorithms are presented. Indicative results include that the problem is NP-hard, there exist static networks where the number of required rounds is $\Theta(\log^2 n)$, while there exist mobile networks where the number of required rounds in $\Omega(n)$ [11], [12], [13]. Using a similar model, the problem of minimizing energy consumption over a static wireless network was recently studied in [14].

### *C. Algorithms for Flooding*

Since flooding in wireless networks results in a prohibitively large overhead [15], a substantial number of more efficient algorithms for broadcasting have been proposed. Usually, these are either based on probabilistic algorithms (see for example [15], [16], [17]) where packets are only forwarded with a certain probability, or some form of topology control (e.g., [18], [19], [20]) to form connected dominating sets of forwarding nodes.

### *D. Network Coding for Wireless*

If we allow intermediate nodes to perform network coding operations, the problem of minimizing the energy per bit when multicasting in an ad-hoc wireless network can be formulated as a linear program and accepts a polynomial-time solution [21]. A distributed algorithm to select the minimum-energy multicast tree is proposed in [22]. Minimum cost multicasting using network coding was also examined in [23] for mobile networks and in [24] for fixed networks.

### *E. This Paper*

With respect to the previous work, our work is positioned as follows. We are interested in wireless networks, where a broadcast message is received by all neighbors within a certain radius (as opposed to epidemic algorithms, where communication takes place with a randomly chosen neighbor). While most work in the broadcasting literature looks at the speed of dissemination, which is measured in terms of the required rounds, our measure of performance is energy efficiency, which translates in number of transmissions. Work in [14] also considers optimization for energy efficiency, but over wireless networks modeled as arbitrary graphs. Although this approach has its merit and is interesting, it is not clear how well it applies in practical wireless networks, where the existence of “edges” connecting nodes reflects the positioning of the nodes on the plane and is not arbitrary.

Moreover, our interest is not in worst case bounds, as in [13], but average performance. In this sense, our work is closer to rumor-spreading using network coding [7]. In fact, when we look at dynamically changing topologies, where nodes do not have information about the network topology, we show that our problem reduces to simple variations of the coupon collectors problem, and thus similar results apply. We make this connection precise in Section VII.

Broadcasting is a special case of multicasting, thus the routing algorithms in [21], [22] also apply in our case. For our special case we derive the exact benefits in terms of energy efficiency we expect to realize, and propose very simple distributed routing algorithms that allow to realize these benefits in practice. Although we derive algorithms with the wireless broadcasting application in mind, simple variations of our algorithms may also be used for content delivery over peer-to-peer networks.

## III. System Model

We here present our problem formulation and briefly review the basic ideas for network coding.

### A. Problem Formulation

Consider a wireless ad hoc network with $n$ identical nodes, where each node is a source that wants to transmit the same amount of information to all other nodes. We assume that time is slotted and that during each time-slot a node $v$ can broadcast one unit of information to all its neighbors $N(v)$ within a given transmission range through physical layer broadcast. We also assume that each broadcast transmission is either successfully received by all $N(v)$ neighbors, or else completely fails. All nodes have the same transmission range.

Our performance metric is the total number of successful transmissions required to transmit one unit of information from all sources to all receivers, denoted as $T_{nc}$ for the case of network coding and $T_w$ otherwise. We assume that the energy expenditure is proportional to the number of successful transmissions (we do not explicitly take into account energy required for computation and reception). Hence, we are interested in calculating the “optimal” energy efficiency defined as the minimum number of such transmissions required under all possible strategies and ignoring “time” constraints.

Since we do not try to maximize the number of successful transmissions that occur simultaneously in time (i.e., throughput), we do not investigate involved schedules that ensure transmissions do not collide or interfere. However, the transmission protocols we use for our theoretical analysis can naturally be implemented in a parallel fashion, i.e., resulting in high throughput as well. In fact, we propose and simulate algorithms that operate in practical networks where nodes attempt to transmit simultaneously, and packet loss is taken into account through a probabilistic model. As discussed in Section V-B, these simulation results as well follow the trend predicted by the theoretical analysis.

For practical systems, we are interested in designing algorithms that are distributed, and are not given an priori knowledge of their neighborhood. In a fixed network nodes may be able to infer some information about their neighborhood by observing the number and pattern of transmissions, while in a fast changing network topology, nodes are not able to collect such information, and thus do not utilize such knowledge.

### B. Network Coding Operation

Let $x_1, \ldots, x_n$ denote the source packets associated with the $n$ nodes. These packets[1] are of equal length and contain symbols from a finite field $\mathbb{F}_q$. Linear network coding allows intermediate nodes to combine incoming packets (symbols). Each packet contains a linear combination of the source packets, as described by a vector of coefficients with respect to the source symbols called coding vector, that is sent appended to the packet [25].

The coding vector can be used by network nodes to decode the data, or further encode it. Encoding can be performed recursively, namely, with already encoded packets. Consider a node that has received and stored a set $(a^1, X^1), ..., (a^m, X^m)$, where $X^i$ denotes the information symbols and $a^i$ the appended coding vector to packet $i$. This node may generate a new encoded packet $(a', X')$ by picking a set of coefficients $h = (h_1, ..., h_m)$ and computing the linear combination $X' = \sum_{j=1}^{m} h_j X^j$. The corresponding coding vector $a'$ is not simply equal to $h$, since the coefficients are with respect to the original packets $x_1, ..., x_n$; in contrast, straightforward algebra shows that it is given by $a'_i = \sum_{j=1}^{m} h_j a_i^j$. This operation may be repeated at several nodes in the network.

In the following it is convenient to think in terms of vector spaces, and say that a node has received a vector space spanned by $m$ coding vectors, when the node has received the $m$ corresponding linear combinations of the source symbols. Each node $v$ collects the coding vectors for the packets it receives (or generates) in a decoding matrix $G_v$. A received packet is said to be innovative if its coding vector increases the rank of the matrix of $G_v$. To transmit, the node generates a linear combination whose coding vector lies in the vector space of its decoding matrix. Once a node receives $n$ linearly independent combinations, or equivalently, a basis of the $n$-dimensional space, it is able to decode and retrieve the information of the $n$ sources. Decoding amounts to solving a system of linear equations with complexity bounded as $\mathcal{O}(n^3)$.

For all practical purposes, the size of the matrices with which network coding operates has to be limited. This is straightforward to achieve for deterministic network codes, but more difficult with random network coding. For the latter, packets are usually grouped into so-called generations, and only packets of the same generation can be combined [25]. Possible alternatives for this grouping are to allocate to a generation packets of a given source, packets generated in within a specific area of the network, packets generated in a certain period of time, packets containing a certain type of

[1]We can think of $x_1, \ldots x_n$ simply as symbols, or as packets of symbols of the same size, and apply to each packet the operations symbol-wise. In the following we will talk about symbols and packets interchangeably.

information, combinations thereof, etc. Each source packet is only part of a single generation.

## IV. Fixed Networks

In this section we consider networks where the topology does not change, and evaluate energy efficiency benefits that use of network coding may offer. We consider random networks, where nodes are randomly placed on the network surface, as well as canonical configurations. In both cases we will assume that the common transmission range of the nodes is such that the number of neighbors $N(v)$ of each node $v$ is upper bounded by a constant, i.e., $N(v) \leq N_{max}$.

*Theorem 1:* Consider a fixed ad-hoc wireless network where each node's transmission is received by a constant number of neighbors. For the application of broadcasting network coding offers constant energy efficiency benefits.

*Proof:* The proof follows from two observations:
$(i)$ There exists a routing scheme (not necessarily optimal) that utilizes $n^2$ transmissions. This is because, there exist exactly $n$ messages to be disseminated, and each of the $n$ nodes will need to broadcast a message to its neighbors at most once.
$(ii)$ Any network coding scheme will utilize at least $n^2/N_{max}$ transmissions. This is because each of the $n$ nodes needs to receive $n$ innovative transmissions and each transmission brings innovative information to at most $N_{max}$ nodes. ■
In fact, for canonical configurations such as networks where nodes are placed on a lattice, we can exactly calculate the benefits in terms of energy efficiency that network coding can offer. We illustrate this through two examples, the circular network and the rectangular grid network.

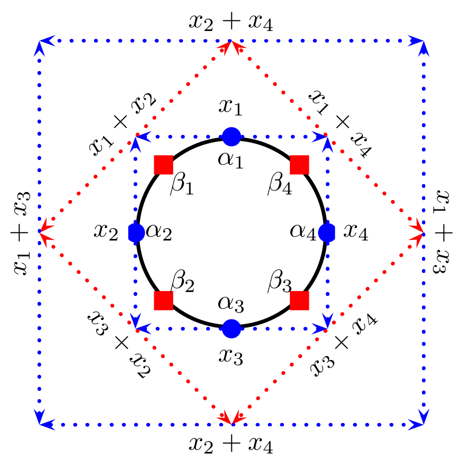


Fig. 1. Circular network with $n = 8$ nodes. Nodes in the set $A$ ($B$) are depicted as circles (squares). Also depicted is the network coding scheme that allows to disseminate the information from nodes in $A$ to all nodes of the network.

### A. Circular Network

Consider $n$ nodes placed at equal distances on a circle as depicted in Fig. 1. Assume that each node can successfully broadcast information to its two neighbors. For example, node $\beta_1$ can reach nodes $\alpha_1$ and $\alpha_2$. The results do not change if we increase the transmission range, as Section VI-C shows.

**Algorithm 1** Network Code for Circular Network

```
Step $k$:
• Phase 1: If $k = 1$, each $\alpha_i \in A$ broadcasts its information symbol $x_i$. If $k > 1$, each $\alpha_i \in A$ transmits the sum of the two information symbols it received in phase 2, step $k-1$.
• Phase 2: Each $\beta_i \in B$ transmits the sum of the two information symbols it received in phase 1, step $k$.
```

*Theorem 2:* Consider a circular network, and the optimal routing and network coding strategies that minimize the number of transmissions for the problem of broadcasting. Then

$$\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{1}{2}.$$

The theorem follows from Lemmas 1, 2 and Section VI-C.

*Lemma 1:* For the circular network it holds that
1) without network coding $T_w \geq n-2$
2) with network coding $T_{nc} \geq \frac{n-1}{2}$.

*Proof:* Since a node can successfully broadcast to its two nearest neighbors, each broadcast transmission can transfer at most one unit of information to two receivers. We have $n-1$ receivers to cover and thus the best energy efficiency we may hope for is $\frac{n-1}{2}$ per information unit. When forwarding w.l.g. we may consider a single source broadcasting to $n-1$ receivers. The first transmission reaches two receivers. Each additional transmission can contribute one unit of information to one receiver. ■

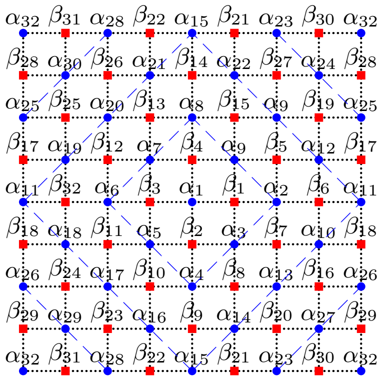
For the case of forwarding, it is easy to see that a simple flooding algorithm achieves the bound in Lemma 1. For network coding consider the following scheme. Assume that $n$ is an even number. Partition the $n$ nodes in two sets $A = \{\alpha_1, \ldots \alpha_{\frac{n}{2}}\}$ and $B = \{\beta_1, \ldots \beta_{\frac{n}{2}}\}$ of size $\frac{n}{2}$ each, such that every node in $A$ has as nearest neighbors two nodes in $B$, as depicted in Fig. 1. It is sufficient to show that we can broadcast one information unit from each node in set $A$ to all nodes in sets $A$ and $B$ using $T_{nc} \geq \frac{n}{2}$ transmissions. We can then repeat this procedure symmetrically to broadcast the information from the nodes in $B$.

Let $\{x_1, \ldots, x_{\frac{n}{2}}\}$ denote the information units associated with the nodes in $A$. Algorithm 1 operates in $\frac{n}{4}$ steps, where in each step first nodes in $A$ transmit and nodes in $B$ receive and then nodes in $B$ transmit and nodes in $A$ receive.

*Lemma 2:* There exist schemes that achieve the lower bounds in Lemma 1. Thus $\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{1}{2}$.

*Proof:* We show that Algorithm 1 achieves the bound in Lemma 1. At step $k$, Phase 1, each node in $B$ is going to receive two new information symbols from the two sources that are $2k-1$ nodes away along the circle[2]. In Phase 2 each node in $A$ is going to receive two information units from the sources that are $2k$ nodes away. Since algorithm 1 concludes in at most $\frac{n}{4}$ steps, and ensures that each broadcast transmission brings new information to two receivers, the result follows. ■

[2]For simplicity of notation, we assume that all indices are $\mod \frac{n}{2}$. Also note that for $n-1$ odd we cannot achieve $\frac{n-1}{2}$ transmissions but $\frac{n}{2}$, however this does not affect the order of the result.



$\alpha_{32}$ $\beta_{31}$ $\alpha_{28}$ $\beta_{22}$ $\alpha_{15}$ $\beta_{21}$ $\alpha_{23}$ $\beta_{30}$ $\alpha_{32}$
$\beta_{28}$ $\alpha_{30}$ $\beta_{26}$ $\alpha_{21}$ $\beta_{14}$ $\alpha_{22}$ $\beta_{27}$ $\alpha_{24}$ $\beta_{28}$
$\alpha_{25}$ $\beta_{25}$ $\alpha_{20}$ $\beta_{13}$ $\alpha_{8}$ $\beta_{15}$ $\alpha_{9}$ $\beta_{19}$ $\alpha_{25}$
$\beta_{17}$ $\alpha_{19}$ $\beta_{12}$ $\alpha_{7}$ $\beta_{4}$ $\alpha_{9}$ $\beta_{5}$ $\alpha_{12}$ $\beta_{17}$
$\alpha_{11}$ $\beta_{32}$ $\alpha_{6}$ $\beta_{3}$ $\alpha_{1}$ $\beta_{1}$ $\alpha_{2}$ $\beta_{6}$ $\alpha_{11}$
$\beta_{18}$ $\alpha_{18}$ $\beta_{11}$ $\alpha_{5}$ $\beta_{2}$ $\alpha_{3}$ $\beta_{7}$ $\alpha_{10}$ $\beta_{18}$
$\alpha_{26}$ $\beta_{24}$ $\alpha_{17}$ $\beta_{10}$ $\alpha_{4}$ $\beta_{8}$ $\alpha_{13}$ $\beta_{16}$ $\alpha_{26}$
$\beta_{29}$ $\alpha_{29}$ $\beta_{23}$ $\alpha_{16}$ $\beta_{9}$ $\alpha_{14}$ $\beta_{20}$ $\alpha_{27}$ $\beta_{29}$
$\alpha_{32}$ $\beta_{31}$ $\alpha_{28}$ $\beta_{22}$ $\alpha_{15}$ $\beta_{21}$ $\alpha_{23}$ $\beta_{30}$ $\alpha_{32}$

Fig. 2. A rectangular grid configuration with 64 nodes enveloping the surface of a torus. For example, the closest neighbors of the node $\alpha_{32}$ are the nodes $\beta_{29}$, $\beta_{31}$, $\beta_{30}$ and $\beta_{28}$.

## B. Rectangular Grid

In this case $n = m^2$ nodes are placed on the vertices of a rectangular grid, and each node can successfully broadcast information to its four nearest neighbors.

*Theorem 3:* Consider the optimal routing and network coding strategies that minimize the number of transmissions for the problem of broadcasting over a square grid network with $n$ nodes. Then

$$\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{3}{4}.$$

The theorem follows from Lemmas 3, 4 and Section VI-C.

*Lemma 3:* For the rectangular grid network it holds that:

1) without network coding $T_w \geq \frac{n^2}{3}$, and
2) with network coding $T_{nc} \geq \frac{n^2}{4}$.

*Proof:*

Each transmission can bring one unit of information to at most four receivers. When forwarding we have an overlap of at least one receiver, i.e., each transmission can bring one unit of information to at most three receivers. ■

*Lemma 4:* There exist schemes that achieve the lower bounds in Lemma 3 and thus $\lim_{n\to\infty} \frac{T_{nc}}{T_w} = \frac{3}{4}$.

*Proof:* For the case of forwarding simply use flooding along one horizontal line and along perpendicular lines.

For the case of network coding, we extend the proof idea in Lemma 2. We partition the square lattice into sub-lattices $A$ and $B$, such that the four closest neighbors for an element in $A$ belong to $B$ (and vice-versa). Let nodes $A$ be sources. We describe a scheme that transmits one information unit from all sources in $A$ to all nodes in $A$ and $B$. Again consider steps divided in two phases, where in the first phase the nodes in $A$ transmit, while in the second phase, the nodes in $B$ transmit. To avoid edge effects, assume that the square grid envelopes the surface of a torus, as depicted in Fig. 2.

We now discuss the connection with the circular network proof. By "distance" between two nodes we refer to the number of hops that separate them. For any node $\alpha$ or $\beta$ in the circular network, the number of neighbors at distance $d$ is two, independent of $d$ (with a possible exception for $d = \frac{n}{2}$, where when wrapping around the circle we may have only one new neighbor). Thus, at every step $k$ in the proof of Lemma 2, it is sufficient for example for nodes $\alpha_i \in A$ to receive two new information units, to learn the information of sources at distance $d = 2k$.

In contrast, in a square lattice, the number of neighbors $N_d$ at distance $d$ can be calculated as $N_d = N_{d-1} + 4$, $N_1 = 4$, $d \geq 2$ (called the coordination sequence of the square lattice). In the case of a grid with $m^2$ points placed on the surface of a torus, the number of new neighbors increases up to a point, and then, because of overlap when wrapping around, starts decreasing. We assume hereafter that $m$ is even, but very similar arguments hold for $m$ odd. Our algorithm for the square grid, in each step, has every node collect the information from all sources that are at a certain (increasing) distance from it. However, since in this case, unlike the circular network, the number of neighbors depends on the distance, the number of transmissions at each step is also not constant.

Algorithm 2 uses this approach. For $k = 1, \ldots, \frac{m}{2}$, each node collects the information from a constantly increasing area. The number of sources collected at step $k$ (and corresponding distances) for nodes in $A$ and in $B$ are provided in Table I. It remains to prove is that there exist linear combinations such that each receiver is able to decode. With a centralized scheme, this amounts to selecting values for the coding vectors such that a product of determinants is nonzero [26]. The sparse-zeros Lemma 6 [26], [27] shows that such values exist. In practice, we can use a randomized approach [28], [29] to find these values (as we do in the next section). ■

TABLE I
NUMBER OF NEW SOURCES AT ALGORITHM 2.

| **step** $k$ | **neighbors** $N_{2k-1}$ for $\alpha_i \in A$ | **neighbors** $N_{2k}$ for $\beta_i \in B$ |
|---|---|---|
| $k = 1$ | 4 | 8 |
| $1 < k < \frac{m}{4} - 1$ | $N_{2k-2} + 4$ | $N_{2k-1} + 4$ |
| $k = \frac{m}{4}$ | $N_{2k-2} + 4$ | $N_{2k-1} + 2$ |
| $k = \frac{m}{4} + 1$ | $N_{2k-2} - 2$ | $N_{2k-1} - 4$ |
| $\frac{m}{4} + 1 < k < \frac{m}{2}$ | $N_{2k-2} - 4$ | $N_{2k-1} - 4$ |
| $k = \frac{m}{2}$ | 4 | 1 |

**Algorithm 2** Network Code for the Square Grid Network

```
Step $k$, $1 \leq k \leq \frac{m}{2}$:
  • Phase 1: If $k = 1$, each node $\alpha_i \in A$ transmits its
    information symbol to their four nearest neighbors. Each
    $\alpha_i \in A$ transmits once. Each $\beta_i \in B$ receives four
    messages. If $k > 1$, each $\alpha_i \in A$ transmits $\lceil \frac{N_{2k-1}}{4} \rceil$
    linear combinations from the $N_{2k-2}$ information units it
    received in phase 2, step $k-1$. Each $\beta_i \in B$ receives the
    information units from all sources at distance $2k-1$.
  • Phase 2: Each $\beta_i \in B$ transmits $\lceil \frac{N_{2k}}{4} \rceil$ from the $N_{2k-1}$
    information units it received in phase 1, step $k$. Each node
    $\alpha_i \in A$ receives the information units from all sources at
    distance $2k$.
($N_k$ calculated as in Table I)
```

## V. Distributed Algorithms

Our goal being to develop distributed algorithms that are well suited for random topologies, we start in this section by developing a distributed algorithm for the square grid network. Given that random topologies with a large number of nodes tend to perform like such a network, we then tune the algorithm to perform well in a random topology, and verify through simulation that we obtain the expected benefits.

### *A. Distributed Algorithms for the Square Grid Network*

The scheduling Algorithm 1 tends to be involved, and thus might be challenging to implement in a practical system. Algorithm 3 employs a much simpler scheduling and still allows us to achieve the optimal benefits in terms of energy efficiency. The algorithm operates in iterations.

**Algorithm 3** Distributed Network Code for the Square Grid

- Iteration 1: Each node broadcasts the information symbol it produces to its four closest neighbors.
- Iteration $k$: Each node transmits a linear combination of the source symbols that belongs in the span of the coding vectors the node has received in all previous iterations.

Let $m_k$ denote the number of innovative packets that node $i$ has received at the end of iteration $k$, and let $V_k^i$ be the vector space spanned by the corresponding coding vectors. That is, $m_k = \dim V_k^i$ is the dimension of the vector space $V_k^i$. Let $A$ be a set of nodes, we denote by $V_k^A = \cup_{j\in A} V_k^j$ the vector space spanned by the union of the vector spaces that nodes in $A$ span.

To show that Algorithm 3 allows to achieve the optimal performance when broadcasting, we need to show that there exists a coding scheme (linear combinations that nodes can transmit) such that each broadcast transmission brings innovative information to four receivers. This implies that Algorithm 3 operates in $k = 1 \dots \lceil \frac{n}{4} \rceil$ iterations as follows. At iteration $k$, each node $i$

1) Transmits a vector from the vector space spanned by the coding vectors the node received at iterations $1 \dots k-1$.
2) Receives four vectors from its four closest neighbors, and increases the dimension of its vector space by four.

Before the iterations begin each node has its own source symbol, and thus $m_0 = 1$. We want to show that for each node $i$ at the end of iteration $k$

$$m_k = m_{k-1} + 4 = 4k + 1. \tag{1}$$

To prove that there exists a coding scheme such that (1) holds, it is sufficient to prove the following theorem.

*Theorem 4:* There exists a coding scheme to be used with Algorithm 3 on the square grid such that at iteration $k$,

$$\dim(V_k^A) \geq \min\{m_k + |A| - 1, n\} \tag{2}$$

for *any* set $A$ of nodes, where $m_k = 4k+1$, $m_0 = 1$.
Indeed, from (2) for $A = \{i\}$ we get that $\dim(V_k^i) \geq m_k = 4k+1$. But node $i$ at iteration $k$ has received $4k$ broadcast transmissions, i.e., $\dim(V_k^i) \leq m_k = 4k+1$. Thus the theorem directly implies that $\dim(V_k^i) = m_k = 4k+1$. For the proof of this theorem we use Lemmas 5 and 6.

*Lemma 5:* Any set $A$ of nodes in the grid, with $4 + |A| \leq n$, has at least four distinct neighbors.

*Proof:* The proof uses the fact that the vertex min-cut between any two nodes in a square grid is four. Let $B$ be the set of nodes in the grid that are not in $A$. From assumption $B$ contains at least four nodes. If all the nodes in $B$ are neighbors of nodes in $A$ we are done. Assume that there exist a node $\beta$ in $B$ that is not a neighbor of any node in $A$. Let $\alpha$ be any node in $A$. Connect $\alpha$ and $\beta$ through four vertex disjoint paths. On each such path there exists a distinct neighbor of $A$. ■

The second result we need is a reformulation of the sparse zeros lemma proved in [26], [27]. Here we present it in a form that is convenient for the proof of our theorem.

*Lemma 6:* Consider a family of $h \times h$ matrices $\mathbf{A}_1, \mathbf{A}_2, \dots \mathbf{A}_N$ whose elements are finite degree polynomials in the coefficients $\alpha_1, \alpha_2, \dots \alpha_\ell$ for some integer $\ell \in \mathbb{N}$. Assume that for each matrix $\mathbf{A}_i$ there exist values $\alpha_1 = p_1, \dots, \alpha_\ell = p_\ell$ over a field $\mathbb{F}_{q_i}$ such that the determinant of $\mathbf{A}_i$ over $\mathbb{F}_{q_i}$ is non zero, i.e., $\det(\mathbf{A}_i) \neq 0$. Then, there exists a finite field $\mathbb{F}_q$, and there exist values in $\mathbb{F}_q$ for $\{\alpha_i\}$ such that $\det(\mathbf{A}_1)\det(\mathbf{A}_2)\dots\det(\mathbf{A}_N) \neq 0$.
For example, if

$$\mathbf{A}_1 = \begin{bmatrix} p^2 & p(1-p) \\ p(1-p) & p^2 \end{bmatrix}, \text{ and } \mathbf{A}_2 = \begin{bmatrix} 1 & p \\ 1 & 1 \end{bmatrix},$$

then for $p = 1$, $\det(\mathbf{A}_1) \neq 0$ over $\mathbb{F}_2$, for $p = 0$, $\det(\mathbf{A}_2) \neq 0$ over $\mathbb{F}_2$, and for $p = 2$, $\det(\mathbf{A}_1)\det(\mathbf{A}_2) \neq 0$ over $\mathbb{F}_3$. In fact, randomly choosing the parameter values over a field $\mathbb{F}_q$ gives us a valid assignment with probability that goes to one as the size of the field $q$ increases [29].
*Proof of Theorem 4:* We use induction.
We use induction.
• For $k = 0$, $m_0 = 1$, since every node has one source symbol.
• For $k = 1$, $m_1 = 5$. Indeed, at the end of the first iteration each node has received the information symbols from its four nearest neighbors. Any $A$ nodes have their own information and moreover the information from their one-step closest neighbors, which, from Lemma 5, amounts to a vector space of size at least $m_1 + |A| - 1 = |A| + 4$.
• Assume that the condition holds for $k = \ell - 1$. It is sufficient to show that it holds for $k = \ell$.

Consider a set $A$. We want to show that $\dim(V_k^A) \geq m_{k-1} + 4 + |A| - 1 = m_k + |A| - 1$. From induction we know that $\dim(V_{k-1}^A) \geq m_{k-1} + |A| - 1$. If $\dim(V_{k-1}^A) \geq m_{k-1} + 4 + |A| - 1$ we are done. The only interesting cases are when $\dim(V_{k-1}^A) = m_{k-1} + i + |A| - 1$, $i = 0 \dots 3$. We will prove here the case where $\dim(V_{k-1}^A) = m_{k-1} + |A| - 1$. For the other three cases the arguments are very similar.

Let $B$ be the set that includes $A$ and all the nearest neighbors of $A$. From Lemma 5 we know that $B$ contains at least four nodes that do not belong in $A$, say $\{\beta_1, \beta_2, \beta_3, \beta_4\}$. We want to show that when the nodes in $\{\beta_1, \beta_2, \beta_3, \beta_4\}$ transmit during iteration $k$, they increase the rank of the set $A$ by four. (And in fact, of every other set they are neighbors.)

But this holds by the following argument. From assumption,

$$\begin{aligned}
&\dim(V_{k-1}^{\{A,j\}}) \geq m_{k-1} + |A|, \text{ for } j \in \{\beta_1, \beta_2, \beta_3, \beta_4\}\\
&\dim(V_{k-1}^{\{A,j,l\}}) \geq m_{k-1} + |A| + 1, \text{ for } j,l \in \{\beta_1, \beta_2, \beta_3, \beta_4\}\\
&\dim(V_{k-1}^{\{A,j,l,z\}}) \geq m_{k-1} + |A| + 2, \text{ for } j,l,z \in \{\beta_1, \beta_2, \beta_3, \beta_4\}\\
&\dim(V_{k-1}^{\{A,\beta_1,\beta_2,\beta_3,\beta_4\}}) \geq m_{k-1} + |A| + 3.
\end{aligned}$$

Thus, nodes $\beta_1$, $_\beta 2$, $\beta_3$, and $\beta_4$ have vectors $v_1$ $v_2$, $v_3$, and $v_4$ respectively such that $v_j \notin V_{k-1}^A$, $j = 1 \ldots 4$, and the vector space spanned by them has dimension four, i.e., $\dim(< v_1, v_2, v_3, v_4 >) = 4$. Then, from Lemma 6, there exist linear combinations that nodes $\beta_i$ can transmit at iteration $k$ such that the vector space of $A$ (and in fact any set $A$ neighboring them) increases in size by four. □

## *B. Distributed Algorithms for Random Networks*

We now extend Algorithm 3 to work over random topologies where the number of neighbors $N(v)$ of a node $v$ is not necessarily constant. Generally, the network is not perfectly symmetric and we cannot assume perfect synchronization among nodes. Moreover, we are interested in very simple protocols where nodes do not have any a priori knowledge about the network topology, and in particular, their neighboring nodes. To account for these factors, and given the randomized nature of our networks, we use a protocol in analogy to probabilistic routing algorithms that forwards packets with a certain probability, according to a *forwarding factor* [16], [17]. The forwarding factor $d$ determines the number of coded packets that will be sent upon reception of innovative packets as described in Algorithm 4. A packet is transmitted by its own source at least once. We combine this approach with randomized network coding [28], [29].

Recall that each node $v$ stores the coding vectors it receives in a decoding matrix $G_v$. In the case of routing the coding vectors are simply the basis vectors $\{e_i\}$ where $e_i$ has one "1" at position $i$ and "0" at all other positions. The matrix has a size determined by the generation size. The matrix of a source $s_i$ that has not yet received information from any other node contains only a single row $e_i$. A received packet is said to be innovative if its coding vector increases the rank of the matrix. Reception of non-innovative packets is simply ignored.

**Algorithm 4** Constant Forwarding Factor $d$

Each node maintains a send counter $s$, that is initially set to zero.

- For each source symbol that originates at a node $v$, the node increases $s$ by $\max(1, \lfloor d \rfloor)$, and it further increases $s$ by one with probability $p = d - \max(1, \lfloor d \rfloor)$ if $p > 0$.
- Similarly, when a node $v$ receives an innovative symbol it increases $s$ by $\lfloor d \rfloor$, and it further increases $s$ by one with probability $p = d - \max(1, \lfloor d \rfloor)$ if $p > 0$.
- If $s \geq 1$, a node attempts to broadcast a linear combination over the span of the received coding vectors. Each transmission reduces the send counter $s$ by one.

Intuitively, good values for $d$ depend on the transmission range and the network topology, such as the neighborhood node density. For example, if a node $v$ forms a vertex cutset for the network, i.e., if removing this node disconnects the network into two components, then this node acts as a bridge that needs to rebroadcast each innovative packet it receives to transfer it between the two components. In contrast, a node in a dense area of the network, that shares each successful reception of an innovative packet with a large number of its neighbors, needs to retransmit more sparingly to avoid overloading the network with redundant transmissions.

Algorithm 5 tries to adapt the number of transmissions of a node according to the node's local neighborhood density, as it is perceived by the packets the node receives.

**Algorithm 5** Forwarding and Receiving Factor

- In addition to updating the send counter as in Algorithm 4, nodes also keep track of received non-innovative packets. For each $c$ non-innovative packets a node receives the send counter $s$ is decremented by one.

To improve the energy efficiency, we can use a *dynamic forwarding factor* $d_v$ different for every node $v$. Such an algorithm can help to adapt to irregularities of the network topology. The value of $d_v$ that would lead to the smallest total number of successful transmissions can only be calculated with perfect knowledge of the network topology. Since we are interested in simple algorithms, we can assume that in a fixed network a node can acquire knowledge about the direct neighborhood as well as the two-hop neighborhood by observing for example the flow of transmissions, while further information is too costly to gather. We therefore investigate the performance of two heuristics to adjust $d_v$. Let $N(v)$ be the set of direct neighbors of node $v$ and let $k$ be a forwarding factor to be used when a node only has one single neighbor. We scale $d_v$ as follows.

**Algorithm 6** Dynamic Forwarding Factor $d_v$

- *Algorithm 6A:* Set $v$'s forwarding factor inversely proportional to the number of 1-hop neighbors
$$d_v = \frac{k}{|N(v)|}.$$
- *Algorithm 6B:* Set the forwarding factor inversely proportional to the minimum of the number of 1-hop neighbors of $v$'s 1-hop neighbors
$$d_v = \frac{k}{\min_{v' \in N(v)} |N(v')|}.$$

Intuitively, if a node $v$ has multiple neighbors but one of the neighbors $v'$ has only node $v$ as a neighbor, $v$ needs to forward all available information to $v'$, no matter how many neighbors $v$ itself has.

The performance of Algorithm 6 also depends on the value of $k$. In essence, $k$ is a cumulative forwarding factor shared between all nodes within a given radio range. It corresponds to

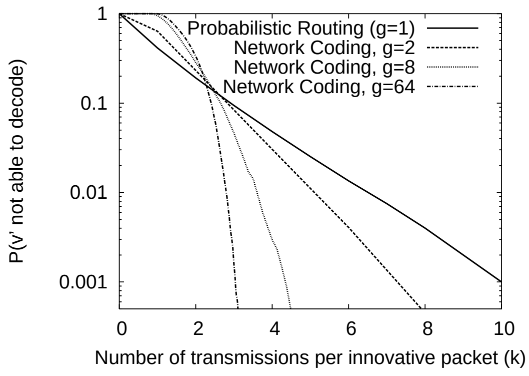


Fig. 3. Probability that a node is not able to decode after $kg$ transmissions for different numbers of information vectors $g$ (i.e., sizes of the decoding matrices).

the number of packets that are transmitted *within this coverage area* as a response to the reception of an innovative packet, independent of the node density.

To determine $k$, we need to compute the probability that a transmitted packet is innovative. In [15], the authors analyze the probability that the broadcast of a given message is innovative for at least one neighbor when this message has already been overheard a certain number of times, for the case of flooding. This probability quickly drops to $0$ for more than ca. $6-8$ overheard broadcasts of the same message. Therefore, $k$ should be set such that the number of broadcasts in an area is close to this value and independent of the network density.

A similar analysis is possible for network coding. As a rough approximation, let us assume that a node $v$ and all but one of its neighbors have all $g$ information vectors, and one neighbor $v'$ has no information. We are interested in the probability that after overhearing $kg$ transmissions, a packet from $v$ will be innovative for $v'$. In other words, $v'$ must have received fewer than $g$ innovative packets from the other nodes and is not yet able to decode.[3]

We compute this probability as follows. Let $D_0$ be a disk of radius $1$ (we can take all transmission ranges equal to 1 since the probability we are interested in is independent of the distance unit chosen). Let $j = kg$, and $D_1, ..., D_j$ be $j$ disks, also of radius 1, with centers in $D_0$, drawn independently and uniformly in $D_0$. Define $Q_j^g$ as the probability that a random point $M$ in $D_0$ is covered by fewer than $g$ of the $j$ disks. Our upper bound is the probability $Q_{kg}^g$. For fixed $g$ and large $k$, we have the approximation

$$Q_{kg}^g \approx \frac{1.72029}{\sqrt{gk}} e^{-0.321021gk}. \quad (3)$$

A more detailed analysis can be found in [30].

The probability of node $v$'s transmission being innovative is depicted in Fig. 3 for the case of probabilistic routing ($g = 1$) and network coding ($g > 1$). With probabilistic routing, this probability decreases exponentially with the number of transmissions, while it drops to $0$ much more rapidly with network coding. The slope of the curve depends on the number of information vectors $g$. In the network scenarios we are interested in, $g$ is on the order of tens to hundreds of information vectors. To achieve probability of not being able to decode below 1%, we have to set $k \approx 3$ for network coding and $k > 6$ for flooding. (Note that this is the probability that $v'$ is not able to decode only using transmissions from nodes in $N(v)$. It might still receive packets via some other neighbors, resulting in a higher overall PDR.) Interestingly, for $k \geq 3$, the probability of not being able to decode tends to 0 in the limit for large $g$, while it is strictly positive for smaller $k$.

[3]In real scenarios, it is extremely unlikely that $v'$ overhears none of the packets that its neighbors received previously to obtain their information. Furthermore, $v'$ may obtain the missing information through a neighbor that is not within $v$'s transmission range. Also this case is not part of the analysis. Therefore, the analysis below is a worst case estimate that gives an upper bound on the probability of $v'$ not being able to decode after $kg$ transmissions.

### C. Distributed Generation Management

Up to now we have assumed that each node is a source that has a single symbol to transmit, and that nodes are able to decode as soon as they receive $n$ linearly independent combinations. Thus, all sources are decoded together at the end of the transmission.

In practice, the node memory and processing capabilities are limited and it might therefore not be possible to keep track of all information in a single matrix. This is especially so since in a random environment there may be benefits in combining symbols not only across space but also *across time* as is observed in the network coding literature. Following the terminology in [25], Algorithms 4–6 can be easily extended to operate over generations. Recall that we define a generation as a collection of packets that we allow to be linearly combined. Dividing packets into generations decreases the decoding complexity, allows to decode data faster (and thus empty the respective memory), as well as use smaller coding vectors. Furthermore, grouping information into generations allows nodes to only decode generations they are interested in, for example based on type of content or local scope. Without central control in the network, nodes have to manage generations based only on their local information. In this section we describe simple distributed generation management methods.

Each node selects the generation for each packet that originates at this node, using a generation size threshold $t$. The node checks which generations it knows having a size that does not exceed the threshold $t$. From these, it randomly picks one generation and allocates the packet to it. If no such generation exists, the node creates a new generation with a random generation ID and inserts the packet. The space of generation IDs has to be large enough so that the probability of having generations with the same ID created by different nodes is relatively small. Alternatively, it is possible to use an ID for new generations that depends on the address of the originating node, which prevents such collisions.

The actual size of generations depends on the threshold $t$ but is not limited by it. Several distant nodes may decide to insert packets into the same generation at the same time. Therefore, $t$ needs to be adapted based on the average size of the matrices at a given node (and can be different for each node). Equivalently, $t$ can be adapted based on the available memory at a node. The higher the probability of nodes inserting many new packets at the same time and the lower the node memory, the lower $t$

needs to be. It is further possible to compose generations that are local in space, while the dissemination of the generation may still be throughout the whole network. For example, we could limit nodes that are allowed to insert packets into a generation to the $\lambda$-hop neighborhood of the node where the generation was created.

## VI. Simulation Results and Practical Considerations

In this section, we present the simulation results for our algorithms, and investigate the effect of parameters such as the forwarding factor, the transmission range, and the generation size.

### A. Description of the Simulator

Unless explicitly stated otherwise, our simulation environment is as described in the following.

Nodes have a nominal transmission range of $\rho = 250m$ and are placed uniformly at random on the simulation area. To avoid edge effects, we let this area envelope the surface of a torus. Transmissions are received by all the nodes within transmission range. We use a custom, time-based network simulator. A packet (symbol) transmission takes exactly one time unit. We assume that a node can either send or receive one packet at a time. The MAC layer is an idealized version of IEEE 802.11 with perfect collision avoidance. At each time unit, a schedule is created by randomly picking a node and scheduling its transmission if all of its neighbors are idle. This is repeated until no more nodes are eligible to transmit.

For finite field operations we select the field $\mathbb{F}_{2^8}$, so that each symbol of the field can be stored in a byte. Addition and multiplication operations can be implemented using two lookup tables of size 255 bytes [31]. The coding vectors are transported in the packet header as suggested in [25]. We use randomized network coding, i.e., combine the received vectors uniformly at random over $\mathbb{F}_{2^8}$ to create the vector to transmit.

We compare our network coding algorithms against probabilistic routing, where received packets are re-broadcasted with a certain probability (similar to our forwarding factor). Our performance metrics are packet delivery ratio (PDR), delay, and overhead. The PDR is measured as the number of packets that can be *decoded* at the destination. For probabilistic routing, this is equal to the number of received innovative packets, whereas with network coding, not all innovative packets can necessarily be decoded. Similarly, delay is counted as the average time between the transmission of a packet by the original source and successful decoding at a node. We also investigate overhead in terms of number of transmissions required to achieve a certain PDR.

### B. Comparison of the Forwarding Algorithms

First we compare the performance of Algorithm 4 against probabilistic routing. The network contains $100$ nodes, randomly distributed on a surface of 1250m $\times$ 1250m. This results in an average number of neighbors of around 12. During the first 100 time units, at each time unit one packet originates at a randomly selected node. Then the simulation continues to run without inserting further packets until no more nodes are eligible to forward. For network coding, we use one single generation that holds the packets from all senders.

As shown in the left graph of Fig. 4, in the static topology network coding achieves 100% delivery ratio for a forwarding factor of 0.25. In contrast, probabilistic routing requires a 3 times larger number of transmissions to achieve the same performance. This difference is more pronounced for intermediate forwarding factors between 0.1 and 0.2, where network coding reaches almost all nodes while probabilistic routing has a PDR below 40%. The average delay from the time a packet originates until it is received (or successfully decoded) at the destination is shown in the right graph in Fig. 4. The decoding delay of network coding does not continue to increase for high forwarding factors, as does probabilistic routing delay. This is due to the fact that with probabilistic routing, multiple duplicates of already received packets may be received before the next novel packet, thus increasing end-to-end delay. With network coding, all of the packets are innovative until the node can decode everything. After this, further received packets are non-innovative, but have no impact on delay. Therefore, decoding delay is only marginally above 100, the minimum number of time units each node needs to receive the 100 packets. For $d$ between $0.1$ and $0.2$, probabilistic routing only reaches nodes that are few hops away, resulting in a small delay.

In more demanding topologies, Algorithms 5, 6A and 6B, are likely to perform better than the simpler Algorithm 4 that uses a fixed forwarding factor for all nodes. We use a network size of 1500m $\times$ 1500m that does not wrap around at the edges and vary the number of nodes between 128 and 1024. The topology has four dense clusters comprising of 70% of the nodes, while the remaining 30% are randomly distributed in the network area. We further ensure that the network is connected. This results in complex topologies with vastly varying node degrees.

For each algorithm we use the lowest forwarding factor that results in a 90% PDR. As can be seen from Fig. 5, Algorithm 4 requires up to twice the overhead of the other algorithms. Algorithms 6A and 6B perform alike for higher node densities, but the fact that Algorithm 6B takes the two-hop neighborhood instead of just the one-hop neighborhood into account helps its performance when node density is low (less than ca. 15 neighbors per node). Here, it is particularly likely that clusters of nodes are connected only via few intermediate connections. Algorithm 5 that limits the overhead when non-innovative transmissions are overheard achieves a performance that is slightly worse than that of Algorithms 6A and 6B. However, it is much more robust with respect to the value of the forwarding factor. It adapts to the “complexity” of the topology more gracefully than the other algorithms, for which a too low forwarding factor results in a reduced PDR, while a forwarding factor that is too high creates unnecessary overhead. The differences between the algorithms are similar but slightly more pronounced when we look at overhead required to achieve a 99% delivery ratio (not shown).

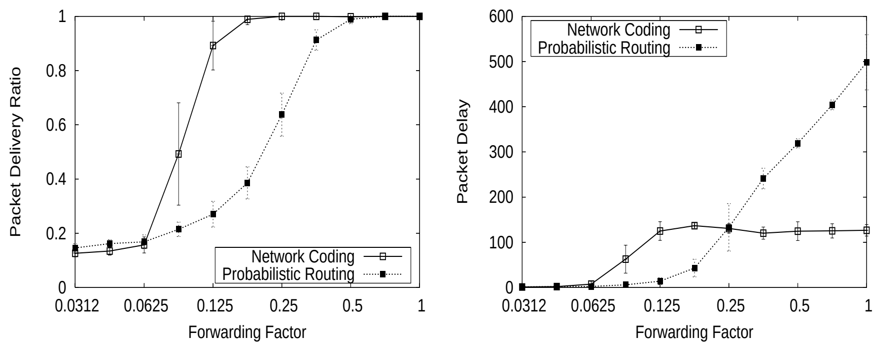


Fig. 4. PDR (left) and end-to-end delay (right) for Algorithm 4 and probabilistic routing. With network coding, the transition to high PDRs occurs for a much lower forwarding factor. It also shows no delay increase for high forwarding factors.

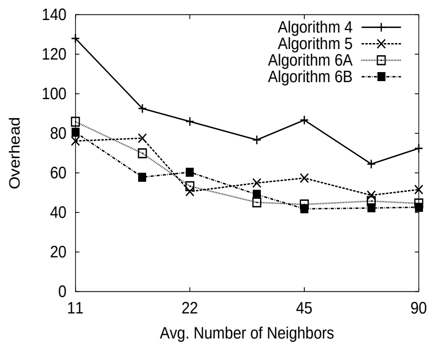


Fig. 5. Comparison of Algorithms 4 - 6 for a clustered topology. Adaptive algorithms outperform Algorithm 4 with a fixed forwarding factor

## *C. Impact of Transmission Range*

In the canonical configurations we have examined up to now we have assumed that each node broadcasts information to its closest neighbors, i.e., to two neighbors in the case of the circular network, and four neighbors in the case of the square grid network. Similarly, in the case of random networks, we assumed that the transmission range is relatively small compared to the size of the network. In this section we investigate how this assumption affects our results. In particular, we assume that all nodes transmit at an identical range $\rho$ (using omni-directional antennas) but that $\rho$ might allow to reach more than the closest neighbors.

In a wireless environment, the transmitted power $P_T$ decays with distance as $\frac{P_T}{\rho^\gamma}$ due to path loss, where typical values are $\gamma \geq 2$. Thus, if a receiver at a distance $\rho$ can successfully receive a signal that has power above a threshold $P_0$, then the transmitted power $P_T$ must increase proportionally to $P_0\rho^\gamma$. Increasing the range of transmission increases $P_T$. On the other hand, increasing the transmission range allows to reach more receivers during each transmission. In the following, we quantify this tradeoff.

*1) Circular Network:* In a circular network, to reach the two closest neighbors, a node needs to transmit at a radius of $2\sin(\frac{2\pi}{n}) = 2\sin(\theta)$. Generally to reach the $2k$ nearest neighbors, $1 \leq k \leq \frac{n}{2}$, a node needs to transmit at a radius of $2\sin(\frac{2\pi k}{n}) = 2\sin(k\theta)$. In the case of network coding, if each broadcast transmission reaches

- *the two closest neighbors*, we need total power

$$P_2 = \frac{n-1}{2}\frac{P_0}{\sin^\gamma(\theta)},$$

- *the 2k closest neighbors*, we need total power

$$P_{2k} = \frac{n-1}{2k}\frac{P_0}{\sin^\gamma(k\theta)}.$$

Thus,

$$\frac{P_2}{P_{2k}} = k(\frac{\sin(\theta)}{\sin(k\theta)})^\gamma = \frac{k}{k^\gamma}\frac{1-\frac{\theta^2}{3!}+\frac{\theta^4}{5!}-\ldots}{1-\frac{(k\theta)^2}{3!}+\frac{(k\theta)^4}{5!}-\ldots},$$

and for large $n$ (small $\theta$) we get that

$$\frac{P_2}{P_{2k}} \approx k^{1-\gamma}. \tag{4}$$

In the case of forwarding, if each broadcast transmission reaches $2k$ neighbors, we need in total power

$$P_{2k}^f = (1+\frac{n-1-2k}{k})\frac{P_0}{\sin^\gamma(k\theta)}.$$

We conclude that in both cases we lose in terms of transmit power when increasing the transmission range, but the ratio $\frac{P_{2k}^f}{P_{2k}}$ remains equal to $\frac{1}{2}$, at least for $k$ much smaller than $n$.

*2) Square Grid:* The square grid can be thought as 2 dimensional lattice $\mathcal{Z}^2$ (enveloping the surface of a torus) that contains all the points of the form $v = xe_1 + ye_2$, where $x$ and $y$ are integers and $e_i$ are the vectors of the orthonormal basis, $e_1 = [1\ 0]$, $e_2 = [0\ 1]$. If we draw a circle in $\mathcal{R}^2$ with radius $k$ around the point $v$ it will contain all points $(x, y)$ satisfying

$$(x-v_1)^2 + (y-v_2)^2 \leq k^2.$$

Thus, if we broadcast at a constant radius $\rho = k \in \mathcal{Z}$,the number of neighbors we can reach equals

$$N_k = \sum_{y=-k}^{y=k}(2\lfloor\sqrt{k^2-y^2}\rfloor+1)-1. \tag{5}$$

If we compare the number of transmissions that we need with and without network coding, we get that

$$\frac{T_{nc}}{T_w} = 1 - \frac{\sum_{y=0}^{y=k-1}(2\lfloor\min\{\sqrt{k^2-y^2},\sqrt{k^2-(y-k)^2}\}\rfloor+1}{\sum_{y=-k}^{y=k}(2\lfloor\sqrt{k^2-y^2}\rfloor+1)-1}$$

Values of this ratio are included in Table II.

TABLE II
CONVERGENCE OF RATIO $\frac{T_{nc}}{T_w}$.

| $k$ | 1 | 2 | 10 | 50 |
|---|---|---|---|---|
| $\frac{T_{nc}}{T_w}$ | 0.7500 | 0.6667 | 0.6013 | 0.6089 |

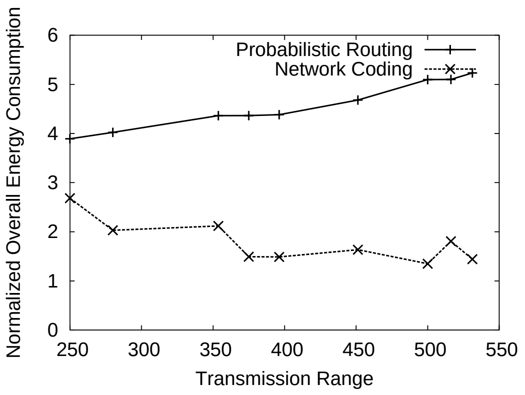
In the case of network coding, if each broadcast transmission reaches

- *the four closest neighbors*, we need total power

$$P_1 = \frac{n-1}{4}P_0.$$

- *the k closest neighbors*, we need total power

$$P_k = \frac{n-1}{N_k}\frac{P_0}{k^\gamma}.$$



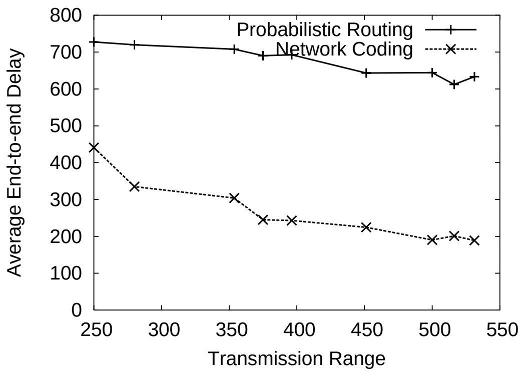


Fig. 6. Forwarding overhead in terms of energy consumption (i.e., number of transmissions $\times$ required transmit power) and decoding delay for different transmit ranges

Thus,

$$\frac{P_1}{P_k} = \frac{N_k}{4k^\gamma}. \tag{6}$$

If $\gamma \geq 2$ and using (5) we can see that $\frac{P_1}{P_k} \leq 1$.

We conclude that for $\gamma = 2$ increasing the transmission range does not affect the energy efficiency. For $\gamma > 2$ the optimal strategy in terms of power efficiency is to transmit to the closest neighbor. Moreover, as the transmission range $\rho$ increases, the benefits network coding offers also increase and converge to approx. $0.609$. This number corresponds to the area of the intersection of two circles with the same radius and centers at distance equal to the radius.

*3) Random Networks:* For the following simulations we also investigate total network energy consumption, which is measured as the sum of transmit power $\times$ transmission time over the duration of the simulation.

In Fig. 6 we show simulation results for a random network with $144$ nodes and a fixed area of 1500m $\times$ 1500m. For each transmission range, we choose the smallest cumulative forwarding factor $k$ for Algorithm 2B that results in an overall PDR of more than 99%. As can be seen from the left graph, with network coding higher transmission ranges even allow to *decrease* the total energy expenditure (assuming a path loss exponent of $\gamma = 2$). Recall that Algorithm 2B is only a heuristic and requires $k$ to be somewhat larger than the optimal value. The intuition behind this result is that, the larger the transmit range, the more "regular" the network becomes in terms of number of neighbors, and the closer $k$ can be set to the optimal value. Note that nodes can trade off the number of transmissions for transmit power, which in turn might allow for simpler MAC layer schedules.

In contrast, for flooding the overall energy consumption increases with the transmit range, since flooding does not allow to reduce the number of transmissions as aggressively as network coding for an increased number of neighbors.

The transmission range might also have an effect on delay. In the right graph of Fig. 6 we see that there is a slight decrease in average (decoding) delay for flooding as well as for network coding, when the transmit range increases. This is the result of two factors: increasing the transmission range implies that more nodes can be reached by a single transmission. On the other hand, scheduling becomes more challenging, as the number of non-overlapping circles that can be simultaneously packed (i.e., transmissions during the same timeslot) is reduced.

### D. Reducing Decoding Complexity

As discussed in Section II, to decode a generation of size $g$, i.e., $g$ linearly independent equations, we need complexity $\mathcal{O}(g^3)$, as we need to perform Gaussian elimination over the $g \times g$ matrix of the received coding vectors.

If at each intermediate node we perform uniform at random combinations over $\mathbb{F}_q$, then the resulting matrix will be a random matrix, with a large fraction of nonzero elements. In [32] it was observed that instead of choosing coding vectors uniformly over $\mathbb{F}_q$, in many cases we get comparable performance by performing sparse linear combinations.This work was motivated by the observation [33] that a sparse random matrix of size $g \times g(1+\epsilon)$ with $lim_{g\to\infty}\frac{\epsilon}{g} = 0$, has with high probability full rank. In particular, this is true if we choose each element of the matrix independently to be nonzero with probability $p = \frac{\log(g)}{g}$, and zero otherwise. Moreover, such a matrix requires $\mathcal{O}(g^2\log(g))$ operations to be decoded. If each node in the graph performs "sparse" linear combinations, we can express the resulting matrix that a receiver needs to decode as a product of sparse matrices which we can solve sequentially. Here we examine the effect of reducing the alphabet size and of forming "sparse" linear combinations through simulation results.

*Reducing the Alphabet Size:* Our simulation results indicated that a relatively small alphabet size is sufficient to achieve good network coding performance. The field of size two, which is much smaller than the average number of neighbors, did not lead to a good performance. However, the field of size $2^2$ performed very similarly to the field of size $2^8$ (which is what we used in all the previous simulations). Further increasing the alphabet alphabet size did provide additional performance gains.

*Reducing the Matrix Density:* We use the following algorithm to generate vectors with a limited number of non-zero entries. As long as the number of non-zero coefficients is lower than a threshold $q$, a row is randomly picked from the decoding matrix, multiplied by a random coefficient, and added to the vector to be sent out. We use a simulation setting similar to that of the previous paragraph. Setting $q = 1$ corresponds to sending out the information of a single row of the decoding matrix which is non-innovative for neighboring nodes with a high probability (in fact, performance degrades to that of probabilistic routing). As soon as $q \approx log(g)$, there is little difference in performance compared to an unrestricted generation of vectors ($q = 100$), as can be seen from Fig. 7.

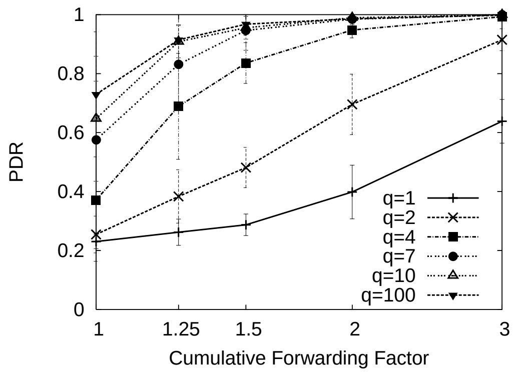


Fig. 7. Impact of reducing the matrix density on PDR

## VII. Dynamically changing topology

In this section we consider networks where the network configuration constantly changes, for example due to mobility. We focus our attention to very simple decentralized distributed algorithms, where nodes do not know the identity of their neighbors. Our motivation is that, in a dynamically changing environment, such updates are costly. We show that use of network coding techniques can offer significant benefits in terms of energy efficiency, through theoretical analysis over a simplified mobility model, and through simulation results over more realistic network scenarios.

### *A. Energy Efficiency Benefits*

For the theoretical analysis we assume a *uniform at random* mobility model. In particular, we divide time into iterations and assume that at the beginning of each iteration nodes are placed uniformly at random on a unit-area disc of radius $1/\sqrt{\pi}$. This corresponds to having a uniform at random mobility pattern, where the iterations are far enough in time, to allow a node to move anywhere on the disc with equal probability since the previous iteration. We use this generous mobility model to simplify the analysis, but examine more realistic models through simulation results in Section VII-B. Moreover, we assume that each node turns off for the duration of each iteration independently at random with probability $p$.

During each iteration each active node transmits within a radius of $r$ with

$$r = \Theta\left(\frac{1}{\sqrt{n}}\right)$$

fixed for all nodes, where $n$ is the total number of nodes. Thus at each iteration each node $v$ has on the average a constant number of $N(v) = N\pi r^2$ neighbors, of which $N(v)(1-p)$ are active.

We compare the energy efficiency in the case where we use forwarding and where we use network coding. We underline our assumption that nodes do not know which are their neighbors, or what information they already have. Thus, in the case of forwarding, without loss of generality we can assume that during each iteration and at each (possibly new) position node $v_i$ always broadcasts $x_i$. In the case of network coding, each node transmits a random linear combination over some finite field $\mathbb{F}_q$ of the symbols it has previously received.

*Theorem 5:* Broadcasting to all receivers can be achieved using on the average
$-$without network coding: $\frac{\Theta(n \log n)}{(1-p)^2}$ iterations,
$-$with network coding: $\frac{\Theta(n)}{(1-p)^2}$ iterations,
where at each iteration occur on the average $(1-p)n$ transmissions. Thus on the average

$$\frac{T_{nc}}{T_w} = \Theta\left(\frac{1}{\log n}\right).$$

*Proof:* Consider first the case of forwarding, and a given node $j$ that would like to transmit its message $x_j$ to all other $n-1$ nodes.

Construct a bipartite graph as follows. The left part consists of the $n-1$ nodes. The right part consists of $M$ nodes $v_i$, where node $v_i$ corresponds to iteration $i$, and is connected to the neighbors of node $j$ during this iteration. Thus the degree of node $v_i$ is a random variable with average $k(1-p)$. We are asking, how many right hand side nodes do we need, i.e., what number of iterations, so that node $j$ transmits its message in all other nodes. This simple analysis has been performed in the context of LT and Raptor codes (see for example [34]-Proposition 1) where it was shown that $M$ should scale as $\Theta(n \log n)$. Since node $j$ is active with probability $(1-p)$, the average number of iterations we need equals $\frac{\Theta(n \log n)}{(1-p)^2}$. This problem can also be viewed as a variation of the coupon collector's problem. The coupon collector's problem in its standard form is described as buying boxes of some product, and in each box there exists one coupon, chosen uniformly at random from a collection of $n$ coupons. We are asking what is the average number of boxes we need to buy to collect all $n$ coupons (see for example [6]). It is well known that in this case the answer is $O(n \log n)$ coupons. Our case is a simple variation, where now each box contains on the average $k(1-p)$ different coupons.

In [7] it was shown that use of network coding with the standard coupon collector problem reduces the number of required iterations to $n$. In our case as well, node $j$ is active on the average $(1-p)m$ out of $m$ iterations. While it is active, it receives on the average $k(1-p)$ transmissions from its active neighbors. Using standard arguments in the network coding literature, and provided that the field $\mathbb{F}_q$ is large enough, each received transmission brings new information to the node $j$. Thus, node $j$ is able to decode all $n$ information units on the average after $\frac{\Theta(n)}{(1-p)^2}$ iterations. $\square$

Note that the performance of network coding is not affected by node mobility. In contrast, mobility has a significant effect on forwarding. Initially, as nodes randomly move, the information is disseminated faster than in the case of a static network. However, because of the assumption that *nodes do not know what information their neighbors have*, as approximately half the nodes collect the information, more and more often transmissions do not bring new information to the receiving nodes. This point has been observed in rumor spreading algorithms over networks represented as graphs, and is known as “the last coupon problem”.

It should be noted that these results do not hold, if we assume that nodes have some information about other nodes in their transmission range. For example, if we assume that a node knows *how many* active nodes are in its transmission range, it can wait (a possibly infinite time) until *all* nodes are simultaneously in its transmission range and are active, and then broadcast its message using just one transmission.

Although we performed this analysis in the context of ad-hoc wireless networks, similar benefits are also possible in environments where we need to broadcast information to a set of receivers in a distributed manner and without knowledge of the network topology. Many of these problems reduce to simple variations of the coupons collectors problem. The following example illustrates one such case.

*Example 1 (Broadcasting in Cellular Networks):* We consider a cellular network model with $m$ base-stations and $n$ mobile phone receivers. The base-stations have $K$ information units that they want to transmit to all mobiles. We assume that the transmission range is the same for all base-stations, each transmission conveys one unit of information, and that the coverage areas of the base-stations do not overlap.

In this model base-stations are always active, while nodes are mobile and may turn on and off. A node is active and successfully receives information approximately $M(1-p)$ out of $M$ iterations. Thus, if base-stations broadcast using an erasure correcting code of rate $(1-p)$, then each transmission brings useful information at each node. For a node to receive $K$ messages we need $\frac{K}{(1-p)}$ iterations.

In the case of forwarding, assume that base-stations randomly select and transmit one of the $K$ messages. Thus each node at each iteration observes one of the messages uniformly at random. We can think of this problem as a balls-in-bins experiment, where the bins are the $K$ messages the node wants to collect, and the balls correspond to the iterations. Using standard results [6] we again need on the average $\frac{n \log n}{(1-p)}$ iterations. Thus, network coding offers a $\log n$ benefit. □

## B. Simulation Results

We use the random topology simulation environment described in Section VI-A. The network size is scaled with the number of nodes such that each node has on average four neighbors. For simplicity, we use Algorithm 4 for the network coding, since the more sophisticated algorithms mainly help with inhomogeneous topologies, while in highly mobile scenarios we tend to observe more canonical topologies. The forwarding factor is set to the lowest value that achieves a 100% PDR.

We first discuss simulation results of the case where at each iteration each node is placed uniformly at random in the rectangular simulation area. We measure the total number of packet transmissions required such that all nodes receive all packets for different network sizes. Fig. 8 shows the ratio of the required number of transmissions of flooding and network coding for different mobility models. Uniformly random mobility corresponds to the mobility model used in the theoretical analysis in Section VII-A. The corresponding curve in Fig. 8 confirms the $\log n$ factor in the ratio of overhead of flooding and network coding from the theoretical analysis. A similar overhead ratio can be observed in simulations with a more dense or more sparse network (not shown here).

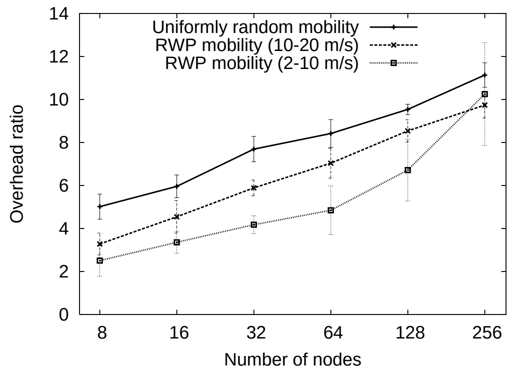


Fig. 8. Ratio of flooding overhead to network coding overhead for random waypoint mobility.

The uniformly random mobility model implies that the composition of the neighborhood of a node is completely uncorrelated from iteration to iteration. In practice, this is true only when the node speed is very high or the packet transmission rate is very low. A less generous mobility implies that less data is transported through the network by node mobility and has instead to be forwarded via intermediate nodes. In Fig. 8 we present simulation results for a more realistic mobility model. We use the same simulation parameters as before, but also show the overhead ratio for mobility according to the random-waypoint mobility model with no pause time and movement speeds uniformly distributed between 2 m/s and 10 m/s as well as 10 m/s and 20 m/s, respectively. With the random-waypoint mobility model, nodes pick a random destination whose location is uniformly distributed in the simulation area as well as a movement speed with which they travel until the destination is reached.

We can see that in this case, although network coding still offers significant benefits, the performance gap with routing is smaller. This agrees with our intuition that when mobility is more restricted, network coding performance deteriorates, because how well the data is “mixed” plays a crucial role for the network coding analysis.

## VIII. Conclusions

We have investigated benefits in terms of energy efficiency that use of network coding can offer for the problem of broadcasting over ad-hoc wireless networks. We proved that network coding can offer a constant factor of benefits over a fixed network, and a $\log n$ factor over a network where the topology dynamically changes. We developed simple distributed algorithms that allow to approach the optimal performance in practice as we demonstrated through simulation results. Our work indicates that there is a potential for significant benefits, when deploying network coding over a practical wireless ad-hoc network environment, especially when we are restricted to use low complexity decentralized algorithms.

## References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, July 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, pp. 371–381, Feb. 2003.

[3] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE Infocom*, Miami, FL, Mar. 2005.

[4] C. Diot, J. Scott, E. Upton, and M. Liberatore, "The Haggle architecture," Intel Research Cambridge, Tech. Rep. IRC-TR-04-016, 2004.

[5] K. Fall, "A delay-tolerant network architecture for challengend internets," in *Proc. ACM Sigcomm*, Aug. 2003.

[6] M. Mitzenmacher and E. Upfal, "Probability and computing : Randomized algorithms and probabilistic analysis," *Cambridge Press*, 2005.

[7] S. Deb and M. Medard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," in *Proc. Allerton*, Oct. 2004.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *ACM SIGOPS Operating Systems Review*, January 1988.

[9] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," *SODA*, 2000.

[10] D. Mosk-Aoyama and D. Shah, "Information dissemination via network coding," *ISIT*, pp. 1748–1752, July 2006.

[11] M. Elkin and G. Kortsarzk, "Logarithmic inapproximability of the radio broadcast problem," *Journal of Algorithms*, vol. 52, pp. 8–25, July 2004.

[12] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, "On the complexity of radion commnication," *SODA*, 1989.

[13] R. Prakash, A. Schiper, M. Mohsin, D. Cavin, and Y. Sasson, "A lower bound for broadcasting in mobile ad hoc networks," *Technical Report (IC/2004/37)*, June 2004.

[14] K. Jain and K. Talwar, "On the power saving of network coding," *Allerton*, Oct 2005.

[15] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. ACM/IEEE Mobicom*, Aug. 1999.

[16] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *IEEE WCNC*, Mar. 2003.

[17] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," in *IEEE Infocom*, June 2002.

[18] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks Journal*, Sept. 2002.

[19] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, Jan. 2002.

[20] J. Wu and F. Dai, "A generic distributed broadcast scheme in ad hoc wireless networks," *IEEE Transactions on Computers*, Oct. 2004.

[21] Y. Wu, P. A. Chou, and K. Jain, "A comparison of network coding and tree packing," *ISIT 2004*, 2004.

[22] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. IEEE Infocom*, Mar. 2005.

[23] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," in *Proc. IEEE Information Theory Workshop*, San Antonio, Oct. 2004.

[24] D. Lun, M. Medard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *ISITA*, Oct. 2004.

[25] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Allerton*, Oct. 2003.

[26] R. Koetter and M. Medard, "Beyond routing: an algebraic approach to network coding," *INFOCOM*, vol. 1, pp. 122–130, June 2002.

[27] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM*, vol. 27, pp. 701–717, 1980.

[28] S. Jaggi, P. Chou, and K. Jain, "Low complexity algebraic multicast network codes," *ISIT*, p. 368, 2003.

[29] T. Ho, M. Mèdard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," *Allerton*, Monticello, IL, Oct. 2003.

[30] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," in *IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[31] C. H. Lim and P. J. Lee, "More flexible exponentiation with precomputation," in *Proc. Advances in Cryptology: 14th Annual International Cryptology Conference*, Aug. 1994.

[32] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," *ISIT*, 2005.

[33] J. Blömer, R. Karp, and E. Welzl, "The rank of sparse random matrices over finite fields," *Random Structures Algorithms 10*, 1997.

[34] A. Shokrollahi, "Raptor codes," *Submitted to IEEE Trans. Information Theory*, 2004.

*Fragouli Christina:* Fragouli Christina is currently an FNS Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, in 1996, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California at Los Angeles in 1998 and 2000, respectively. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, and DIMACS, Rutgers University. She currently serves as an editor for IEEE Communications Letters. Her interests are in network coding, network information flow theory, communication and coding theory, ad-hoc wireless and sensor networks, and connections between communications and computer science.

*Widmer Joerg:* Widmer Joerg is project manager at DoCoMo Euro-Labs in Munich, Germany. He leads the Self-Organized Ambient Networking research group, working on several projects in the area of wireless communication, specifically mesh and sensor networks. Before joining DoCoMo, he spent two years as postdoctoral researcher at EPFL, Switzerland, doing research on ultra-wide band networks. Joerg Widmer obtained his M.S. degree and Ph.D. degree in computer science from the University of Mannheim, Germany in 2000 and 2003, respectively. In 1999/2000, he was at the ICSI Center for Internet Research, Berkeley, CA working on equation-based congestion control. His research interests include efficient algorithms for wireless mesh and sensor networks, network coding, and transport protocols.

*Le Boudec Jean-Yves:* Le Boudec Jean-Yves is full professor at EPFL, fellow of the IEEE and director of the Institute of Communication Systems. He graduated from Ecole Normale Superieure de Saint-Cloud, Paris, where he obtained the Agregation in Mathematics in 1980 (rank 4) and received his doctorate in 1984 from the University of Rennes, France. From 1984 to 1987 he was with INSA/IRISA, Rennes. In 1987 he joined Bell Northern Research, Ottawa, Canada, as a member of scientific staff in the Network and Product Traffic Design Department. In 1988, he joined the IBM Zurich Research Laboratory where he was manager of the Customer Premises Network Department. In 1994 he joined EPFL as associate professor. His interests are in the performance and architecture of communication systems. His work includes the "MAC emulation" which later became the ATM forum LAN emulation project, contributions on network calculus, and on simulation and stationarity of mobility models (Infocom 2005 Best Paper award). He was on the program committee of many conferences, including Sigcomm, Sigmetrics and Infocom, was managing editor of the journal Performance Evaluation from 1990 to 1994, and is on the editorial board of ACM/IEEE Transactions on Networking.