Network Coding for Efficient Communication in Extreme Networks

Jörg Widmer*
DoCoMo Euro-Labs
Landsbergerstr. 312
80687 Munich, Germany
widmer@acm.org

Jean-Yves Le Boudec
Ecole Polytechnique Fédérale de Lausanne
(EPFL)
CH-1015 Lausanne, Switzerland
jean-yves.leboudec@epfl.ch

ABSTRACT

Some forms of ad-hoc networks need to operate in extremely performance-challenged environments where end-to-end connectivity is rare. Such environments can be found for example in very sparse mobile networks where nodes "meet" only occasionally and are able to exchange information, or in wireless sensor networks where nodes sleep most of the time to conserve energy. Forwarding mechanisms in such networks usually resort to some form of intelligent flooding, as for example in probabilistic routing.

We propose a communication algorithm that significantly reduces the overhead of probabilistic routing algorithms, making it a suitable building block for a delay-tolerant network architecture. Our forwarding scheme is based on network coding. Nodes do not simply forward packets they overhear but may send out information that is coded over the contents of several packets they received. We show by simulation that this algorithm achieves the reliability and robustness of flooding at a small fraction of the overhead.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Performance

Keywords

Network Coding, Delay-Tolerant Network

1. INTRODUCTION

Traditional network architectures as used in the Internet are not well suited for communication in environments, where the construction of a continuous end-to-end path between source and destination is difficult or impossible. Such intermittent connectivity is common, for example, in very sparse mobile networks [12] or in sensor networks where nodes sleep most of the time to conserve energy.

Delay-tolerant networking (DTN) architectures [4] are designed to cope with the adverse conditions found in such environments. Routing protocols in DTNs often have to make use of so-called knowledge oracles that provide full or partial information about past or future node encounters (and possibly also buffer occupancies and traffic demand). As mentioned in [11], algorithms that do not have access to such information usually perform poorly compared to algorithms that make use of knowledge oracles. Nevertheless, there exist many scenarios where such information is not available or can only be obtained at very high cost.

In this paper, we are interested in stateless algorithms which neither have information about future encounters nor are able to derive such information from past encounters. Existing algorithms in this area are usually based on some form of flooding. A node forwards a packet with a fixed probability p. If the packet is forwarded, the neighbors that are reached do the same, unless they already sent out the packet. The higher p, the larger the fraction of nodes that are reached by the packet and for $p = 1$, we obtain network-wide flooding. This principle of rebroadcasting a given message with a certain probability has been called epidemic routing, probabilistic routing, or gossiping. Throughout this paper, we will use the term probabilistic routing.

We propose a communication algorithm that is similar to probabilistic routing but is based on network coding. Network coding [1] is a relatively recent field in information theory. In contrast to simply forwarding the information contained in the packets, nodes may send out packets with linear combinations of previously received information. Perhaps the simplest example in our setting is a three node topology where nodes A and C want to exchange packets via an intermediate node B. A [resp. C] sends a packet a [resp. c] to B, which then broadcasts a `xor` c instead of a and c in sequence. Both A and C can recover the packet of interest, while the number of transmissions is reduced.

Linear network coding, in general, is similar to this example: an intermediate node sends a linear combination $\sum_i g_i x_i$ of the packets x_i it has received or wants to send, where all coefficients g_i and the data x_i are interpreted as numbers in a finite field (in this example, the field is $\mathbb{F}_2 = \{0, 1\}$ and the linear combination sent by B after receiving $x_1 = a$ and $x_3 = c$ is simply $x_1 \oplus x_3 = a \oplus c$). When a node receives m combinations of n packets, it can decode them provided that the set of combinations has rank n; for $m = n$ this occurs with a probability close to 1 if the coefficients g_i are chosen randomly and independently at every node[10]. This appealing property of network coding makes it interesting for decen-

*The research for this work was done while J. Widmer was with EPFL.

SIGCOMM'05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.

tralized wireless ad-hoc networks [9, 2]. We propose to code over the field $\mathbb{F}_{2^8}$, which can efficiently be implemented for 8-bit processors, using combinations of `xor` and a small lookup table. This is feasible even for the limited resources of sensor nodes.

We say that a packet received by a node is innovative if it increases the rank of the set of received packets at this node. As with probabilistic routing, our network coding-based algorithm has a parameter that controls with which probability the reception of an innovative packets causes the node to send a packet.

The memory available at a node constrains the number of information units a node can code over. We discuss how efficient network coding can be achieved even with little available memory. Also for a high probability of successful decoding, it helps to not code over very old information units. We present a scheme for information aging that allocates very little memory to old information while maintaining a relatively high probability that nodes that still require this information (because they just woke up or entered the network) can still decode as long as some of the information is still available.

We compare the performance of our network coding-based forwarding to alternative protocols and find that it significantly outperforms probabilistic routing, particularly in challenging scenarios where connectivity is rare. An overview of related work is given in Section 2. Our main contribution is a network coding-based protocol for delay-tolerant networks and is described in Section 3. It consists of methods for deciding when to send a packet, for limiting the size of a generation, and for aging of information. Simulation results to analyze its behavior in performance-challenged environments are presented in Section 4.

2. RELATED WORK

As mentioned in the introduction, we are interested in stateless routing protocols for delay-tolerant networks [4], where no information about node encounters or traffic demand is available. Existing stateless routing protocols can be divided into wait-and-forward and flooding-based algorithms. Since wait-and-forward algorithms usually only work under certain assumption on the mobility model, we will primarily concentrate on flooding-based algorithms.

In case of highly dynamic and scarcely connected networks, probabilistic routing is one of the few routing algorithms that can achieve a certain degree of service. In [5], Fall mentions that using message replication or more sophisticated coding techniques such as erasure coding [17] together with probabilistic routing are possible candidates. Such algorithms are also of high interest in the context of survivability of mobile networks [21], where multipath or spray routing are possible ways to cope with adverse network conditions.

Randomized broadcasting and gossiping have been studied as communication paradigms for networking for quite a while [7]. A good introduction to epidemic information dissemination is given in [3]. The paper discusses basic mathematical properties of epidemic algorithms where nodes "infect" neighbor nodes with a certain probability (which corresponds to a forwarding probability in networking). In [20], the properties of such algorithms were analyzed in the context of percolation theory.

ZebraNet [12] is one of the prominent examples where epidemic algorithms have successfully been used for information dissemination in ad-hoc sensor networks.

Vahdat et. al. look at a slightly different aspect of epidemic routing in ad-hoc networks [22]. Instead of adapting the forwarding probability, they investigate protocol performance when nodes can buffer only a limited number of packets. During an encounter, nodes check available packets and will always exchange innovative packets if such packets exist and the buffer space is not exhausted.

The PROPHET protocol [16] improves upon plain epidemic routing by trying to predict the probability of future node encounters. However, the authors assume that these probabilities can be derived from past encounters and that node mobility is not entirely random. In particular, for this protocol to work well, the nodes' encounter probabilities should not be homogeneous (as would be the case for the random-waypoint or random-walk mobility models).

Haas et. al. [6] propose a gossiping protocol for wireless ad-hoc networks. They compare their protocol to basic flooding and conclude that it achieves the same delivery ratio with 35% fewer messages, while the routes tend to be 10%-15% longer than with flooding. This work is primarily interested in using gossiping for route discovery in reactive ad-hoc routing protocols such as AODV. Gossiping has also been used for reliable multicast in ad-hoc networks [18].

The concept of **network coding** allows interior nodes of a network to not only forward but also to process information they receive. In their seminal work on network coding, Ahlswede et al. [1] showed that in general, the broadcast capacity of a network can only be reached with network coding. This still holds true if encoding functions at the nodes are restricted to linear coding [14]. In addition to throughput gains, a number of problems such as minimum-energy multicast have a polynomial-time solution when using network coding [24], while they are NP-complete for normal routing.

Much work has been published on optimal centralized network coding algorithms [19, 13]. However, it is also possible to do network coding in a completely decentralized manner. Algorithms where the coefficients of the encoding functions are chosen randomly [9, 2] have properties that make them very suitable for wireless ad-hoc networks [8, 23].

3. A NETWORK CODING PROTOCOL FOR INFORMATION DISSEMINATION

We now describe our protocol. In addition to the basic operation of network coding, we specify when a node is allowed to send a packet, with which other packets it can be combined (i.e., which packets constitute a generation), and how to age out old packet generations.

3.1 Network Coding Model

We use the following discrete time model to describe our architecture, loosely following the terminology used in [2].

Let V be the set of nodes in the network and $S \subseteq V$ be the set of source nodes (with $m = |S|$ elements). Each node $v \in V$ has a set of neighbors $N(v)$ it can reach through physical layer broadcast. Let x_i be the information vector that originates at source $s_i, i = 1, ..., m$. A vector contains symbols that lie in the finite field $\mathbb{F}_{2^k}$ for some k. To allow an efficient implementation of the coding, we use operations over the finite field $\mathbb{F}_{2^8}$, so that each symbol of the finite field can be stored in a byte. Addition and multiplication operations over this finite field can be implemented using `xor` and two lookup tables of size 255 bytes [15].

Each information vector is associated with an encoding vector g. For x_i, this is simply the ith unit vector e_i. Let the $M = |x_i|$ be the number of symbols contained in vector x_i (i.e., the size of the data packet payload). Information vectors are always sent together with the corresponding encoding vector [2]. In general, a node will send out a linear combination of all the information it has (at the given time).

A node v stores the tuples of encoding vectors and information vectors it receives, as well as its original information vector in case

it is a source, in a so-called decoding matrix G_v. The matrix has a variable size (depending on the generation size, see Section 3.3) upper bounded by $m(m+M)$. The matrix of a source s_i that has not yet received information from any other node contains only a single row (e_i, x_i).

By $(g_v(t), y_v(t))$ we denote the tuple of encoding vector and information vector that is sent out by node $v \in V$ at time t and received by the neighbors $v' \in N(v)$.

$$(g_v(t), y_v(t)) \quad = \quad r_v(t)\, G_v(t) \tag{1}$$

where $G_v(t)$ is the matrix at node v at time t, and $r_v(t)$ is a random vector of non-zero symbols of length m. Each node $v' \in N(v)$ that receives $(g_v(t), y_v(t))$ will insert it into its matrix $G_{v'}$ (as the last row). A received packet is said to be innovative if its vector increases the rank of the matrix. Reception of non-innovative packets is simply ignored. (For probabilistic routing, a received packet is innovative if it is the first time the node receives it, and non-innovative otherwise.)

To decode the original source vectors, the matrix $G_v(t)$ is kept in reduced row echelon form using Gaussian elimination. As soon as the matrix contains a row (e_i, x_i), the node can decode source packet x_i. At the very latest, this will be the case after m innovative packets have been inserted into a matrix coded over m original vectors. However, a node will usually be able to retrieve some of the original source vectors beforehand.

This model can easily be extended to support generation of source vectors at a given time rather at the beginning as well as multiple source vectors per source.

3.2 When to Send a Packet

Our network coding protocol works in analogy to probabilistic routing algorithms that forward packets with a certain probability. We generalize this concept by introducing a so-called forwarding factor $d > 0$. For probabilistic routing, when $d \leq 1$, d is simply the probability to rebroadcast an innovative packet; when $d > 1$, $\lfloor d \rfloor$ copies of the packet will be rebroadcasted, and a further copy will be rebroadcasted with probability $d - \lfloor d \rfloor$ (i.e., the fractional part of d).

Similarly, with network coding, when an innovative packet is received for a given generation, $\lfloor d \rfloor$ vectors will be generated from the corresponding matrix and broadcasted to the neighbors. A further vector is generated and sent with probability $d - \lfloor d \rfloor$.

Packets that originate at a source are handled differently. Obviously, if the source decides not to send out the packet, none of the other nodes will ever receive it. Such packets need additional protection and are therefore sent out (or, in case of network coding, vectors are generated) $\max(1, \lfloor d \rfloor)$ times, and an additional packet is sent out (generated) with probability $d - \lfloor d \rfloor$ if $d > 1$. In other words, a new packet is sent out by the source at least once.

With the network coding model described above, a node sends out on average $d\,G + 1$ packets, where G is the maximum generation size ($G = m$ in the simplest case, see next section). Since it can decode all original vectors when it receives a number of innovative packets equal to this generation size, a good value for d strongly depends on the number of neighbors (i.e., the node density). Similar to probabilistic routing, a high forwarding factor results in a high decoding probability at the expense of a high network load.

3.3 Managing Generations

In a simplistic model, every node would have only one packet to transmit and the packets of all sources would be combined. However, this is not realistic as nodes will usually have more than one packet to send. Further, the maximum matrix size that would have to be stored (i.e. $m(m+M)$) might be too large to hold in memory. For these reasons, doing the coding and decoding using only a single large matrix is impractical. To limit the size of the matrix, source vectors have to be grouped into generations. There exists one matrix per generation and only vectors of the same generation can be combined.

Let $x_{i,j}$ be the jth information vector that originates at source s_i. The function $f(x_{i,j})$ determines which generation the packet belongs to and

$$\Gamma_\gamma = \{x_{i,j} | f(x_{i,j}) = \gamma\}$$

is the set of all source vectors of a generation γ.

How to manage generations is an important design decision. As mentioned above, using one single generation only works for very constrained scenarios. We have investigated several generation management methods. We found that simply incrementing the generation index from time to time as suggested in [2] does not work well for wireless ad-hoc networks. We simulated generation membership based on local scope (only packets that were originally emitted few hops away from the node that created the generation are allowed to be in the same generation), but it also did not perform as well as the last alternative, generation hashing, which we presented next.

Generation Hashing: Generation membership is determined through hashing over sender address and packet identifier. For a good protocol performance, it is necessary to adapt the hash function to the number of sources in the network. This can be done locally by analyzing the average size of the decoding matrices. Whenever the average size of the decoding matrices becomes too large, a hash function that produces fewer collisions is chosen. It is not necessary to use the same hash function for all nodes. The hash function is only used at a node to determine which generation to insert a given packet into, provided the size of this generation does not exceed a certain maximum threshold.

To determine which specific generation to use, the node computes a hash value between 1 and the number of generations whose size does not exceed this threshold. The node then checks, if the generation corresponding to the hash value already contains a packet from the node. If this is the case, the node starts a new generation and the given vector is included in this generation. Otherwise, the vector is inserted into the generation the hash value points to.

The choice of the hash function determines the number of packets in a generation and thus has a significant impact on network coding performance. New generations should only be created when necessary since the fewer the number of packets in a generation, the smaller the benefits of network coding. Therefore, a node may additionally wait for some time before starting an own generation, during which a suitable generation might be created by a neighbor.

3.4 Information Aging

When a node can decode a packet it is interested in, it passes the packet to the application layer. Once the node has decoded the whole generation, the corresponding matrix is no longer of use to the node. However, its information might be required by neighbors in order to be able to decode.

A node should therefore keep information from a matrix, as long as the node is still eligible to generate packets from it. However, it is not necessary to keep the whole matrix. Instead, to save memory, a node may over time reduce the rank of a matrix it has already decoded. This can be done for example by replacing the last two rows r_{n-1}, r_n of the matrix by a single row r'_{n-1}, that is a random linear combination of r_{n-1} and r_n. This operation does not reduce the probability that a packet generated from this matrix is innova-

tive for a neighbor, it merely reduces the number of such packets that can be generated. As long as a sufficient number of nodes still store at least a single vector from a given generation, a new node will be able to decode all of the packets of that generation.

4. SIMULATIONS

To analyze the performance of network coding and probabilistic routing, we implemented both algorithms in a custom time-based network simulator. The time between sending a packet until it is fully received at the destination is exactly one time unit. A node can either send or receive and it can only send or receive one packet at a time. Transmissions use physical layer broadcast and have a nominal range of 250m. The MAC layer is an idealized version of CSMA/CA. 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 at a given time step. In most of the sparse scenarios that we are interested in, packet collisions do not play any role. It was further shown in [23], that network coding is much more robust against packet loss than alternative algorithms. We believe that, if anything, a more accurate physical layer model will result in a larger performance gain for network coding.

We investigate the performance of these protocols for different levels of knowledge about the local neighborhood, which can be obtained with the help of beacon messages. In the most basic case *without beacons*, no information about the neighborhood is available, and a node sends its packets without knowing if anyone will receive them. For *normal beacons*, a node sends out periodic beacon messages which allows to detect neighbors. The node will not send out a packet if it has no neighbors. With *intelligent beacons*, beacons not only announce neighbor presence but also the information the neighbor already has. A node will not send out a packet if *all* of its neighbors already have the information contained in it.[1]

For a fair comparison, probabilistic routing has a random access interface queue so that together with intelligent beacons, a node can send out an innovative packet if any packet in its interface queue is innovative for any of the neighbors. Since the overhead of the beaconing is the same for network coding and probabilistic routing, we omit the beacon packets in the simulations and simply assume that the corresponding information is available at the nodes.

We first analyze simulations where the network coding algorithm uses a single generation as described in Section 3.1 and later investigate the influence of different generation sizes on protocol performance. For all simulation results we also show 95% confidence intervals.

4.1 Different Node Densities

First we compare the performance of network coding against probabilistic routing for three different node densities. The network always contains 100 nodes. Nodes do not sleep.

To avoid edge effects, we let the network area wrap around at the edges (i.e., it envelopes the surface of a torus). The dense network has a size of 1250m $\times$ 1250m, resulting in an average number of neighbors of around 12. The medium sized network has four times the area with an average number of neighbors of 3. The sparse network is again a factor of 4 larger than the medium network and nodes have on average less than 1 neighbor. In most cases, the dense network provides an end-to-end path, while the medium network has several disconnected clusters. The sparse network has almost no connectivity and only node mobility will allow nodes to communicate.

For network traffic, one packet originates at a random node per time unit for the first 100 time units. 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.

Dense Networks: As shown in the left graph of Figure 1, 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 overhead to achieve the same. The difference in performance is most 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%.

Network coding also benefits much more from node mobility than probabilistic routing (Figure 1, right graph). In these simulations, nodes move according to the random waypoint mobility model with 0 pause time, a minimum speed of 2 m/s (to avoid non-stationary simulations) and a maximum speed of 10 m/s. With probabilistic routing, the PDR increases by roughly 10%, but the forwarding factor required for a 100% delivery ratio does not change. In contrast, for network coding, the phase transition where the PDR increases rapidly occurs earlier. A PDR of 100% is achieved for a forwarding factor of 0.125, resulting in a forwarding overhead 6 times lower than that of probabilistic routing.

The forwarding factor directly determines the forwarding overhead, i.e., the total number of MAC layer transmissions per successfully delivered packet that is unique (and that can be decoded, in the case of network coding). From Figure 2 (left graph) we can see that both protocols have roughly the same forwarding overhead. However, between $d = 0.05$ and $d = 0.1$, the overhead of network coding is slightly higher. In this region, the network coding algorithm already delivers a high number of innovative packets, but this number is still not high enough to allow decoding. In contrast, between $d = 0.1$ and $d = 0.2$ where the Packet Delivery Ratio (PDR) of network coding increases rapidly, its overhead is slightly lower than that of probabilistic routing. As a reference we also plot the curve $f(x) = x$. For low values of d, overhead for both protocols is slightly higher than this curve since sources send out new packets at least once.

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 Figure 2. Interestingly, 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, many 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 early packets tend to be 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. Overhead and delay results for the mobile case are very similar and are therefore omitted.

Sparse Networks: We now analyze PDRs for networks with lower node densities. In contrast to the dense scenarios, knowledge about the local neighborhood can significantly improve protocol performance when connectivity is much more random. We therefore compare network coding and probabilistic routing for static networks, mobile networks without beacons, mobile networks with normal beacons, and mobile networks with intelligent beacons. As

[1]It would be easy to further improve the performance of network coding by sending out information that provides the most benefit to *all* of the neighbors. However, in this paper we restrict ourself to very simple low-complexity algorithms.

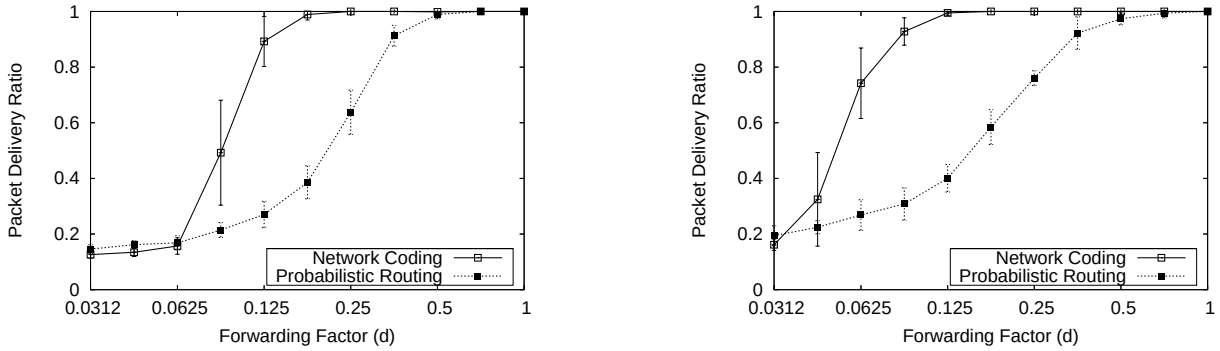


Figure 1: Packet delivery ratio for different forwarding factors in dense static (left graph) and mobile (right graph) networks. With network coding, the transition where all of the nodes receive all packets occurs for a much lower forwarding factor.

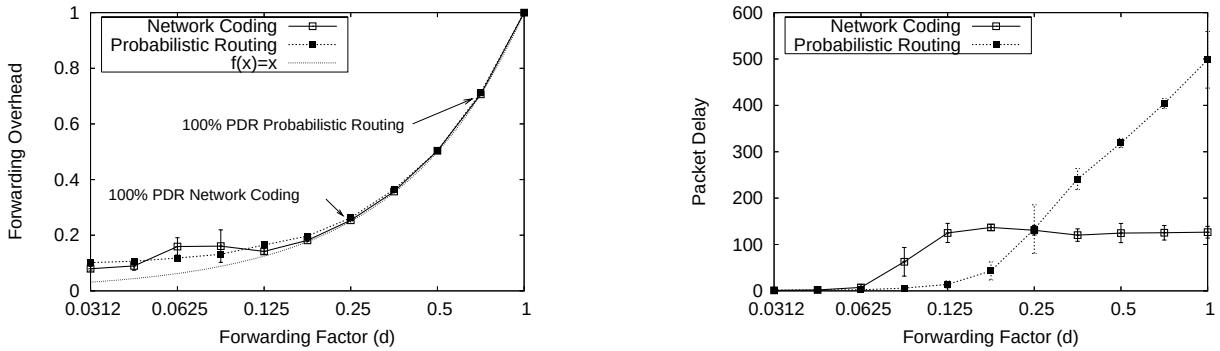


Figure 2: Forwarding overhead (left graph) and end-to-end delay (right graph) for different forwarding factors for static networks. While overhead is similar for both protocols, network coding shows no delay increase for high forwarding factors.

mobility model we again use random-waypoint mobility with 0 pause time, a minimum and maximum speed of 2 m/s and 10 m/s, respectively.

In general, we observe that confidence intervals are larger since communication opportunities depend much more on the current topology. Due to the low probability of node encounters and forwarding possibilities, we also consider forwarding factors above 1.

Already for medium network densities shown in Figure 3, node mobility is essential. Without mobility, network coding as well as probabilistic routing perform the same and for forwarding factors $d \geq 1$ simply reflect the degree of connectivity in the network. When the nodes are mobile and no beacons are used, network coding achieves a 100% PDR for $d = 1$. In contrast, even for a four times higher forwarding factor, probabilistic routing does not able to deliver more than 90% of the packets. The probability of having at least one neighbor is still sufficiently large, which is why the performance without beacons and with normal beacons is about the same. Only with intelligent beacons is probabilistic routing able to achieve a 100% delivery ratio.

For very sparse networks with an average number of neighbors below 1, probabilistic routing fails for any reasonable forwarding overhead Figure 4. Only with intelligent beacons and $d \geq 1$ does it achieve 100% PDR. However, this simply amounts to passing a packet from one node to another whenever two nodes meet and one has a packet the other has not yet seen. This results in a very large average end-to-end delay of more than 1600 time units.

Network coding fares much better. With normal beacons and even without beacons it gets to or close to a 100% PDR for forwarding factors between 2 and 4. This suggests that a random operation of networks without any coordination is possible with network coding even in extremely sparse networks. Average decoding delay is around 800 for the worst parameter settings and between 500 and 600 otherwise.

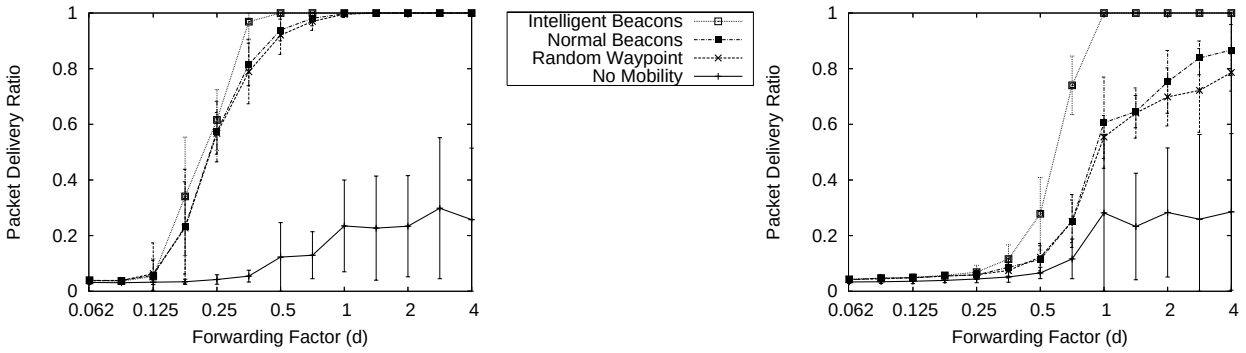
4.2 Impact of Generation Size

Well-known examples of existing sensor nodes are the Berkeley MICA2 Motes and the nodes used for ZebraNet. From a network coding point of view, both architectures provide similar features with a 64 to 128KB on-chip program flash memory and a 512KB serial flash memory for data. A calculation of network coding memory requirements yields the following results.

Each entry in a generation requires space for the corresponding sender identifier as well as the encoding coefficient in the vectors that belong to this generation. Recall that we use a symbol size of 1 byte and let us further assume that a node identifier is 1 byte long. Table 1 shows the packet header overhead of network coding, as well as the size of the matrices required to decode a generation for two different packet sizes.

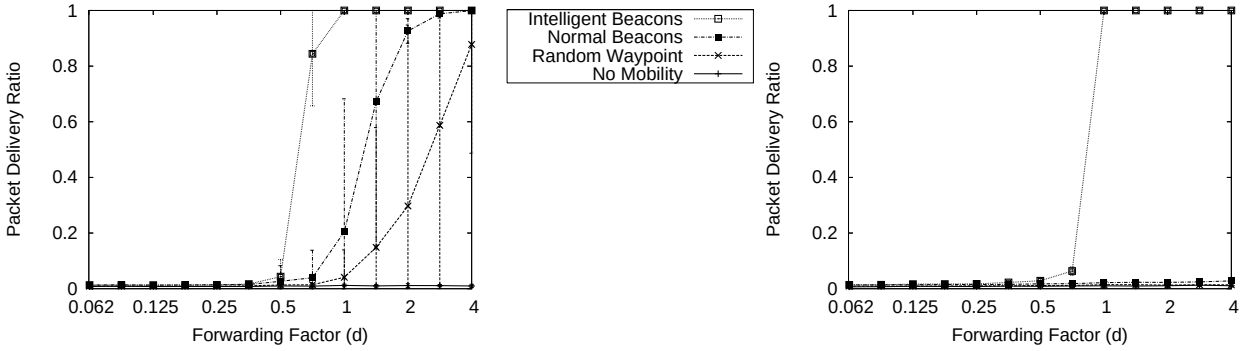
Table 1: Overhead of network coding

Generation size $\|\Gamma\|$	4	8	16	32
Relative overhead	6.25%	12.5%	25%	50%
Matrix size (bytes)	516	1032	2064	4128



(a) Network coding (b) Probabilistic routing

Figure 3: Packet delivery ratio for different forwarding factors in networks with medium density.



(a) Network coding (b) Probabilistic routing

Figure 4: Packet delivery ratio for different forwarding factors in sparse networks.

For network coding to be efficient, it has to reduce the required number of transmissions by at least the relative overhead. For example, network coding with a generation size of 16 and a packet size of 128 bytes needs to achieve at least the same delivery ratio as probabilistic routing using only 75% of the packets. For a packet size of 64 bytes, the overhead doubles and the size of the matrix is slightly more than half as large as before. Clearly, a generation size of 32 does not make sense with packets of 64 bytes, since the whole packet payload would be used up by the sender IDs and encoding coefficients. However, small generation sizes may still be useful. The size of the decoding matrix including the source ID labels is $(ps+1)|\Gamma|$ bytes for a packet size of ps. For reasonable generation sizes, current sensor nodes can easily hold several generations in memory.

Figure 5 shows the impact of generation management on PDRs. Except for the generation size, we use the same number of nodes, network size, and traffic pattern as in the previous scenario of a dense network. The curve for a generation size of 100 (G=100) corresponds to the one shown in Figure 1. With a generation size of 1 (G=1), network coding performance degenerates to that of plain probabilistic forwarding.

As we have seen in Figure 2, for a given forwarding factor the number of transmissions for network coding and probabilistic routing is almost the same. We can therefore concentrate solely on the PDR. When using generation hashing, already small generation sizes provide a significant improvement. For a generation size of G=4, a PDR above 99% is reached for $d = 0.35$, whereas G=1

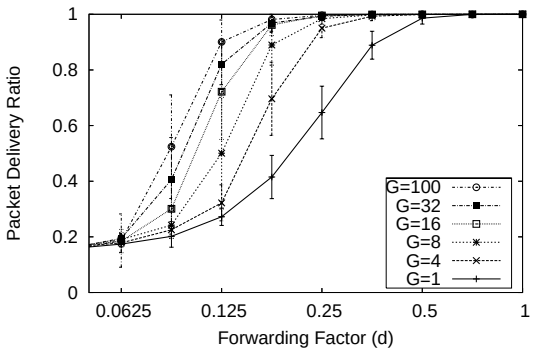


Figure 5: PDRs for different generation sizes with generation hashing. Probabilistic routing corresponds to G=1

or probabilistic routing require $d = 0.7$ to achieve the same. The 100% performance improvement is offset by a 6.25% increase in overhead for a packet size of 128 bytes and 12.5% for a packet size of 64 bytes. This shows that network coding does provide benefits even in the most constrained scenarios. The performance gains become much more significant for moderately larger generations and less stringent delivery requirements. For PDRs above 95% and a generation size of G=16, network coding offers a gain of a factor of 2.5 to 3.

4.3 Extreme Network Conditions

Finally, we present an example of truly extreme network conditions. We use the same simulation parameters as in the simulation with medium density in Section 4.1 (i.e., 100 nodes with roughly 3 neighbors per node). Nodes move with speeds between 2 m/s and 10 m/s and normal beacons are used. In addition, there is a 20% packet loss probability and nodes sleep for a certain fraction of the time. The forwarding factor is fixed at $d = 1$.

Nodes are awake for one time slot and then sleep for an amount of time that is uniformly distributed between 0 and a maximum sleep time. The maximum sleep time is chosen such that a given average sleep ratio (time spent sleeping over total time) is achieved.

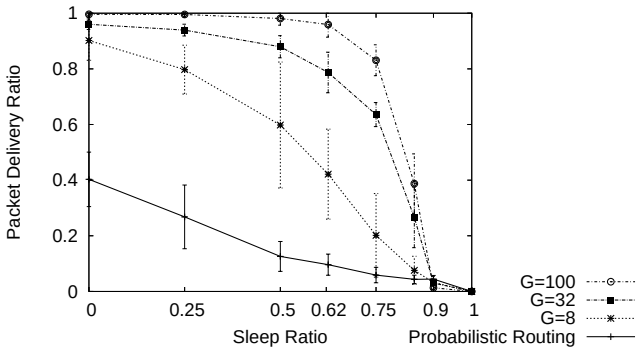


Figure 6: PDRs for different ratios of nodes' sleep time for network coding with different generation sizes and probabilistic forwarding.

We observe, that even under such extreme conditions, the network coding protocol continues to work well. Particularly for larger generation sizes, packet delivery ratios are still in a regime where one can consider the network to be "usable". Only when nodes sleep for 90% or more of the time, performance drops considerably (and a higher forwarding factor would be necessary).

5. CONCLUSIONS

With this paper, we have shown that a network coding based forwarding algorithm compares very favorably to probabilistic routing. A network coding-based solution allows to disseminate information to all nodes in a network with a high probability at a significantly lower overhead. This is specifically the case for performance-challenged network environments. Even for very small sizes of the decoding matrix, a significant performance gain is possible. Such encoding and decoding is entirely feasible in terms of processing power and memory requirements even on the very limited hardware found for example in sensor nodes.

One interesting characteristic of our network coding protocol is, that it benefits more from node mobility than probabilistic routing. Furthermore, even for extreme conditions such as a sparse mobile network with a high packet drop rate and with nodes that sleep for a considerable amount of time, the network coding protocol performs reasonably well, whereas a protocol based on probabilistic routing is simply unusable.

This work is only a first step towards a full communication architecture. In the future, we intend to investigate a number of issues. We have mainly investigated network coding performance in terms of packet delivery ratio. Particularly for sensor networks, energy consumption is a prime concern and would be interesting to analyze for our protocol.

Finally, a data collection architecture like ZebraNet, where mobile sensors exchange data to deliver it to a common sink node, has different requirements than the model used in this paper. With such an architecture, it is not necessary for non-sink nodes (i.e., the actual sensors) to be able to decode any data. Such nodes can therefore combine information vectors almost arbitrarily with little concern for matrix size. The challenge is then to find a coding scheme that results in the highest probability for the sink to be able to decode, given certain constraints on node memory (and possibly sink processing capabilities). We intend to investigate the performance of such an architecture in future work.

6. REFERENCES

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

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

[3] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–6, May 2004.

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

[5] K. Fall. Messaging in difficult environments. Technical Report IRB-TR-04-019, Intel Research Berkeley, Dec. 2004.

[6] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *Proc. IEEE Infocom*, June 2002.

[7] S. M. Hedetniemi, S. T. Hedetniemi, and A. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(129-134), 1988.

[8] T. Ho, B. Leong, M. Medard, R. Koetter, Y. Chang, and M. Effros. On the utility of network coding in dynamic enviornments. In *Proc. International Workshop on Wireless Ad-hoc Networks*, June 2004.

[9] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *Proc. Allerton*, Oct. 2003.

[10] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. submitted to IEEE Transactions on Information Theory, July 2003.

[11] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. ACM Sigcomm*, Aug. 2004.

[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. ASPLOS-X*, Oct. 2002.

[13] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–796, Oct. 2003.

[14] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, Feb. 2003.

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

[16] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proc. SAPIR Workshop*, Aug. 2004.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

[18] J. Luo, P. T. Eugster, and J.-P. Hubaux. Route driven gossip:

Probabilistic reliable multicast in ad hoc networks. In *IEEE Infocom*, pages 2229–2239, San Francisco, CA, Mar. 2003.

[19] P. Sanders, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, June 2003.

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

[21] J. Sterbenz, R. Krishnan, R. R. Hain, A. Jackson, D. Levin, R. Ramanathan, and J. Zao. Survivable mobile wireless networks: Issues, challenges, and research directions. In *Proc. WiSe*, Sept. 2002.

[22] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, Apr. 2000.

[23] J. Widmer, C. Fragouli, and J.-Y. Le Boudec. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. In *Proc. Workshop on Network Coding, Theory, and Applications*, Apr. 2005.

[24] 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*, Oct. 2004.