International Journal of Computer Applications (0975 8887)
Volume 85 - No. 10, January 2014

Analysis the Performance of Network Coding for Ad Hoc Networks in Realistic Simulation Scenarios

Saed Tarapiah
Telecommunication Engineering Dept.
An-Najah National University
Nablus,Palestine

Shadi Atalla
Lavoro Autonomo (LA)
Torino,Italy

Ahmed Masri
Telecommunication Engineering Dept.
An-Najah National University
Nablus,Palestine

ABSTRACT

Network coding has recently emerged as an effective solution for multicast and broadcast communications in ad hoc networks. We focus on broadcast traffic and design a network coding-based scheme that we compare against simpler solutions, through extensive simulations in the ns-2 network simulator. Indeed, while often the benefits of network coding have been shown via theoretical analysis or in simplified simulation scenarios, our aim is to assess the performance of network coding in ad hoc networks when realistic MAC and physical layers are considered. The performance of network coding for traffic broadcasting strongly depends on the network node density and on the generation size. In particular, network coding fails to work in sparse networks, where connectivity is low, and leads to significant gains in terms of end-to-end packet loss probability in dense networks, where congestion is likely. It requires neither a global nor a partial view of the network, nor does it require information about neighboring nodes. Moreover, achieved results show that the protocol delivers broadcast data reliably with minimal network overhead, by eliminating redundant data transmissions, even under adverse network conditions.

Keywords:

Ad-hoc networks, Broadcast, Network Coding, ns2

1. INTRODUCTION

Mobile ad hoc networks (MANETs) are self-organizing mobile wireless networks that do not rely on any preexisting infrastructure to communicate. Broadcasting is a communication operation in which one node sends a message to all other nodes in the network. It is widely used in MANETs with several different aims such as finding a route to a particular node, sending a warning signal, performing service discovery or for topology update.
The simplest method to implement broadcasting is flooding, where every node in the network retransmits a message to its neighbors upon receiving it for the first time. Flooding, however, can lead to undesired effects such as the well-known *broadcast storm problem*: redundant packet retransmissions resulting in repeated contention, collisions, and extra-power consumption [4]. More sophisticated solutions, probabilistic-based counter-based, distance-based, location-based, neighbor knowledge-based, and cluster-based, have been proposed to solve the above shortcomings [4].

In this paper we address the problem of efficiently supporting broadcast traffic in mobile ad hoc networks by using network coding. This technique, firstly introduced in [2], can indeed reduce the overhead due to multiple copies of broadcast transmissions, by letting intermediate nodes encode multiple packets into a single output packet before forwarding.
Several works, e.g., [16], [8], have shown that network coding can provide very high reliability, bandwidth gains, reduced delays and better traffic distribution, thus leading to both *load balancing* and *energy consumption reduction*. However, previous works have shown these benefits either via theoretical analysis or in simplified simulation scenarios. Here, we evaluate the impact of network coding in a realistic ad hoc network environment with Constant Bit Rate (CBR) traffic, using the network simulator ns-2.
In our scheme, the source node as well as intermediate relay nodes encode native/incoming packets by using random linear network coding and broadcast them to all downstream nodes. We compare the performance of broadcasting based on network coding with that of two simpler schemes, namely flooding-based and deferred broadcasting.
Preliminary results of our study are reported in [3]. In this paper, in order to account for more realistic scenarios, we modeled the channel propagation through the Rayleigh distribution [15] which accounts for multipath effects and time-varying channel conditions. Using realistic MAC and physical layers, we can observe the effects of packets loss and propagation delay on network coding.
Indeed, firstly, a single packet loss has the potential to cause multiple packet losses in terms of decoded packets at the receiver, by invalidating the encoded information. Secondly, different propagation delays, typical of multihop communication in ad hoc networks, force the destination to wait for all packets that were encoded together to be received before decoding can be performed. Finally, packetization delay at the source is exacerbated by the need to wait for receiving enough packets from the application layer before encoding can be performed.

2. RELATED WORK

Several solutions have been proposed with the aim to improve the efficiency of broadcasting in ad hoc networks. A widely applied approach is based on probabilistic message transmissions [4], in which the messages generated by a traffic source are rebroadcast by other nodes with a certain probability. This solution reduces the overhead with respect to simple flooding, however it may lead to

a low delivery ratio, especially in the case of scarcely connected nodes.
In counter-based schemes [4], instead, a node determines whether to rebroadcast a packet by counting how many identical packets it receives over a given time interval. In [17] the authors enhance the previous work in [4], by adaptively adjusting the probability of transmission or delay timer by taking local connectivity information into account. As for broadcasting based on network coding, a practical scheme is proposed in [10], where network coding is applied only by a subset of nodes which are selected as forwarders. Also, through promiscuous mode, a forwarder can decide not to send a packet if it hears that all of its neighbors have received it. Even though the scheme in [10] gives good performance, it relies on the exchange of neighborhood information and on the partial dominant pruning algorithm presented in [11], which may cause significant overhead and add complexity relatively to probabilistic approaches.
A randomized broadcast scheme based on network coding is proposed in [12]. There, the network coding parameters are finely tuned for the case of a grid topology so as to reduce the packet loss probability with respect to simple flooding. However, such benefits can be perceived only in dense networks where the packet loss probability due to collisions can be higher than the probability of decoding failure [9]. In [7] the impact of several IEEE 802.11-based MAC protocols on the performance of network coding for broadcast communications is analyzed. We highlight that to the best of our knowledge only a few works, e.g., [7], [6], exist on network coding for ad hoc networks, referring to practical scenarios or using realistic network simulators. Thus, one of the main goals of our work is to assess the performance of network coding for message broadcasting by using the network simulator ns-2 as an evaluation tool [1]. Furthermore, we implement an efficient network coding scheme that does not require nodes to have knowledge on their two-hop neighbors; we compare its performance against both simple flooding and deferred broadcast, the latter including the solution proposed in [13].

3. RANDOM LINEAR NETWORK CODING: AN OVERVIEW

The network coding scheme we use in this work is the *random linear network coding*, where the output flow at a given node is obtained as a linear combination of its input flows. In more detail, this scheme regards a block of data as a vector over a certain base field and applies a linear combination to this vector before forwarding it. It is referred to as *random* because the transformation is performed by each node independently of the others, and using random coefficients [8]. The coefficients of the combination are typically referred to as *coding vector*, and, by definition, the coefficients are selected independently and randomly from a finite field. We introduce the network coding operations and notations below.

3.1 Network Coding Operations

Three different operations are performed when network coding is applied:

—*Encoding*. Let $M_1, ..., M_n$ denote the source native packets, and s denote the generation size, i.e., the maximum number of native packets that can be encoded together in one new encoded packet. Note that the header of each encoded packet must specify the generation to which the packet belongs. An encoding node outputs a generic encoded packet:

$$Y_j = \sum_{i=1}^{s} e_{j,i} M_{s(j-1)+i} \quad j = 1, 2, ... \lceil n/s \rceil$$

with $e_{j,i} \in GF(2)$, where $\mathbf{e_j} = (e_{j,1}, e_{j,2}, ..., e_{j,s})$ denotes the coding vector independently and randomly chosen at the node to encode the packet, and j represents the generation identifier the encoded packets belong to. It is worth pointing out that in $GF(2)$ the sum operation is the bitwise `xor`. The coding vector is included in the header of the transmitted packet and it is used by receivers to decode the data or to further encode the data, as explained below.

—*Re-encoding*. The re-encoding operation is performed over encoded packets. When a node has received $u \leq s$ encoded packets belonging to the same generation, then the node may generate a new encoded packet, by picking a new random vector $\mathbf{g_k} = (g_{k,1}, g_{k,2}, ..., g_{k,u})$, with $g_{k,i} \in GF(2)$ and still chosen independently and randomly, and by computing the linear combination

$X_k = \sum_{i=1}^{u} g_{k,i} Y_{k,i}$ with $g_{k,i} \in GF(2)$ and where $Y_{k,i}$ represents the i-th received innovative encoded packet within the k-th generation identifier.

It must be noted that the coding vector with respect to the original packets $M_{s(k-1)+1}, M_{s(k-1)+2}, ..., M_{sk}$ is now $\mathbf{e'_k} = (e'_{k,1}, e'_{k,2}, ..., e'_{k,s})$, where $e'_{k,i}$ should be computed as the following linear combination:

$$e'_{k,i} = \sum_{h=1}^{u} g_{k,h} e_{h,i}$$

The new coding vector $\mathbf{e'_k}$ is included in the header of the newly encoded packet.

—*Decoding*. Decoding at any receiver can be performed by collecting packets of a given generation. These packets yield a system of linear equations that need to be solved to retrieve the original native packets. Suppose a node has received v encoded packets $X_1, X_2, ..., X_v$ belonging to a given generation, i.e., k, with $v \leq s$, while $\mathbf{e'_1}, \mathbf{e'_2}, ...\mathbf{e'_v}$
represent the coding vectors corresponding to the encoded packets. The decoding matrix G is as follows:

$$G = \begin{pmatrix} \mathbf{e'_1} \\ \mathbf{e'_2} \\ . \\ . \\ . \\ \mathbf{e'_v} \end{pmatrix} = \begin{pmatrix} \mathbf{e'_{1,1}} & \mathbf{e'_{1,2}} & \cdots & \mathbf{e'_{1,s}} \\ \mathbf{e'_{2,1}} & \mathbf{e'_{2,2}} & \cdots & \mathbf{e'_{2,s}} \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ \mathbf{e'_{v,1}} & \mathbf{e'_{v,2}} & \cdots & \mathbf{e'_{v,s}} \end{pmatrix}$$

The rank of this matrix is obtained by computing the inverse of G through the Gaussian elimination method. Let us denote the rank of G by R. When the matrix has full rank, i.e., $R = v = s$, for a given generation, then the node can solve the system of linear equations to retrieve all native packets belonging to that generation. By deriving e'_i from the packet headers and solving the linear equations with M_i as the unknowns:

$$\begin{pmatrix} \mathbf{X_1} \\ \mathbf{X_2} \\ . \\ . \\ . \\ \mathbf{X_v} \end{pmatrix} = \begin{pmatrix} \mathbf{e'_{1,1}}\ \mathbf{e'_{1,2}}\ \ldots, \mathbf{e'_{1,s}} \\ \mathbf{e'_{2,1}}\ \mathbf{e'_{2,2}}\ \ldots\ \mathbf{e'_{2,s}} \\ .\ \ .\ \ \ldots\ \ . \\ .\ \ .\ \ \ldots\ \ . \\ .\ \ .\ \ \ldots\ \ . \\ \mathbf{e'_{v,1}}\ \mathbf{e'_{v,2}}\ \ldots\ \mathbf{e'_{v,s}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{M_{s(k-1)+1}} \\ \mathbf{M_{s(k-1)+2}} \\ . \\ . \\ . \\ \mathbf{M_{s(k-1)+v}} \end{pmatrix}$$

the destination can recover the set of native packets $M_{s(k-1)+1}, M_{s(k-1)+2}, ..., M_{sk}$.

In [5] an early decoding process is suggested, so that the receiver could recover some source packets before a node receives all s innovative packets within a given generation for full decoding. However, the method to perform it is not specified there.

A node which has received $v < s$ innovative packets builds all possible submatrices as combinations of $i \leq v$ encoding vectors. When the submatrix consisting of i encoding vectors for an encoded packet, which is just a combination of i native packets, has got a full rank, then the receiver can perform an early decoding to retrieve all i native packets.

Figure 1 shows four different innovative packets that arrived in order X_1, X_2, X_3, X_4 and are cached at receiver node for further decoding. Let we assume the generation size, s, is set to 5, so upon only receiving X_1 and X_2, the receiver cannot recover any native packet. However, as soon as X_3 arrives, the receiver can recover both the native packets $M_1$and M_5. Then, when X_4 arrives, M_3 will be recovered.

In Figure 1 is is shown how the decoding matrix is built from the encoding vectors embedded within the packets header. Then the sub-matrix is built as a combination of X_2, X_3, X_4. By deleting the all zeros columns we get the sub-matrix which will lead to the linear system to be solved in order to recover the native packets, as depicted in Figure 1.

The receiver can also perform an early decoding when a G sub-matrix has full rank, i.e., when the sub-matrix rank is equal to v, with $v < s$. In this case, the receiver can recover v out of s source native packets belonging to the given generation.

We finally observe that when a node receives a packet, it must check whether it is *innovative* or not, i.e., whether it increases the rank of the decoding matrix G. If not, the packet is dropped.

4. BROADCASTING SOLUTIONS

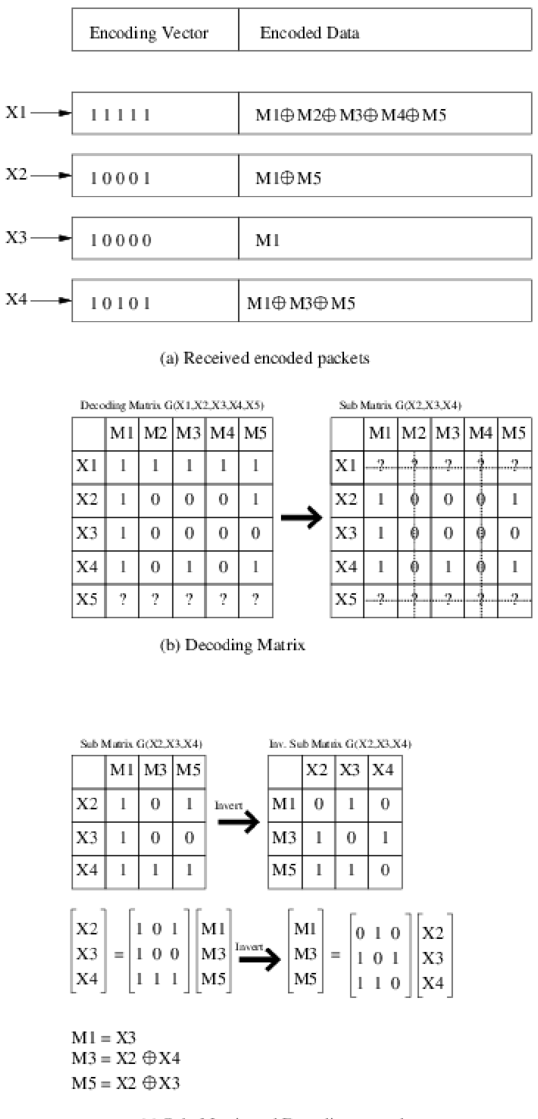
Here we introduce our network coding-based scheme and briefly describe two other broadcasting schemes that provide a term of comparison with our own.

4.1 Network Coding-based Broadcasting

Our network coding-based broadcasting is implemented on top of the network layer running over an IEEE 802.11 MAC protocol, as depicted in Figure 2(a). This choice allows us to avoid encoding routing address information carried in IP headers.

As suggested in [5], we embed the coding vector in the packet header, thus dispensing with the need for centralized knowledge of the graph topology or decoding functions. Therefore, our network coding-based scheme is a topology-independent network coding solution, where every node may decode a set of input packets that arrive from different neighboring nodes. In particular, the encoding header follows the IP header as the first piece of data in the IP payload.

The network coding header, which we refer to as *RLNC-header*, contains information about the encoded packet, as depicted in Figure 2(b).



(a) Received encoded packets

	M1	M2	M3	M4	M5
X1	1	1	1	1	1
X2	1	0	0	0	1
X3	1	0	0	0	0
X4	1	0	1	0	1
X5	?	?	?	?	?

	M1	M2	M3	M4	M5
X1	?	?	?	?	?
X2	1	0	0	0	1
X3	1	0	0	0	0
X4	1	0	1	0	1
X5	?	?	?	?	?

(b) Decoding Matrix

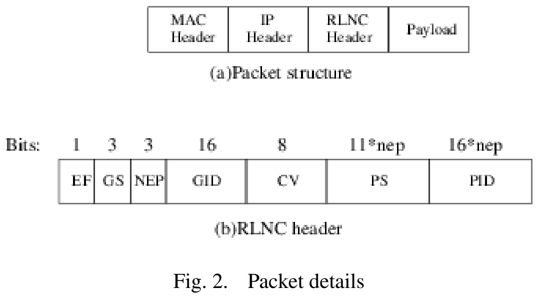
	M1	M3	M5
X2	1	0	1
X3	1	0	0
X4	1	1	1

	X2	X3	X4
M1	0	1	0
M3	1	0	1
M5	1	1	0

$$\begin{bmatrix} X2 \\ X3 \\ X4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} M1 \\ M3 \\ M5 \end{bmatrix} \xrightarrow{\text{Invert}} \begin{bmatrix} M1 \\ M3 \\ M5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} X2 \\ X3 \\ X4 \end{bmatrix}$$

(c) Sub–Matrix and Decoding procedure

Fig. 1. Early decoding



(a)Packet structure

(b)RLNC header

Fig. 2. Packet details

The description of each field in the *RLNC-header* is explained as below:

—Encoding Flag (EF): it indicates whether the packet carries an encoded piece of data or just a native one.

—Generation Size (GS): it represents the maximum number of native packets that can be encoded together into a new encoded packet. This information allows the receiver to build and configure the dimensions of the decoding matrix in order to recognize when full rank is reached. Furthermore, it indicates the maximum number of innovative encoded packets that can be re-encoded together. We set equal to 8 the maximum value of GS, thus a 3-bit long field is enough to store this value.

—Number of Encoded Packets (NEP): it shows the number of native packets that are encoded along the current packet. It is a 3-bit long field, as the GS field.

—Generation IDentifier (GID): only packets having the same generation identifier number, GID, can be encoded/re-encoded together. Moreover, at the receiver side, innovative packets within the same generation identifier are collected in their corresponding decoding matrix. The GID field is 2-byte long.

—Coding Vector (CV): it holds the corresponding coding vector used to encode the packet. The length of the encoding vector is determined by the generation size, thus, its maximum length is 8 bits.

—Packet Size (PS): it includes the packet size of every native packet that has been used to create the encoded packet (the same generation can carry native packets of different sizes). Shorter packets are padded with zeros during the encoding procedure; the receiving node uses this field to remove the trailing zeros in the padding by checking the native packet size from the PS field in the *RLNC-header*. We need 11 bits to represent the size of every native packet that is encoded, by assuming the maximum size of packets carried over the ad hoc network equal to 2000 bytes. The reader should notice that at this stage of research we consider packets of the same size, future work will consider realistic video traffic, accounting for packet of different sizes.

—Packet IDentifier (PID): it carries the identifier of every packet which is generated at the transport layer, at the source node only, and encoded before the transmission by every intermediate node. Every packet identifier requires 2 bytes.

Thus, the *RLNC-header* roughly accounts for an overall overhead of $(4 + (NEP \times 4))$ bytes, which in the worst case, namely when NEP is equal to the maximum generation size, 8, is equal to 40 bytes.
We now detail the different operations performed by the source node, the intermediate nodes, and the receiver nodes. It is worth remarking that, in broadcasting, intermediate nodes are also receiver nodes; however, for the sake of clarity, we distinguish the two operations. Also, note that each node (except for the source) needs to collect s (recall that s is the generation size) independent packets for further encoding/decoding, therefore some buffer space is needed at the receiver. In the following, we will call this buffer *NC buffer*.

—*Source node operations.* The source node's application layer generates native packets, which are then encoded at the network coding sub-layer. The source collects s native packets and encodes them to generate s different encoded packets $Y_{j,1}, Y_{j,2}, ..., Y_{j,s}$ with $j \in \{1, 2, ... \lceil n/s \rceil\}$ representing the generation identifier, as explained in Section 3.
The coding vectors are $\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_s}$: the elements of the generic vector $\mathbf{e_i}$ $(i = 1, \ldots s)$ are all zeros except for the i-th element, which is equal to one. The source neighbors that receive s encoded packets belonging to a given generation, can therefore decode the corresponding s native packets. Therefore, regarding a given generation, i.e., k, the sender can generate s independent packets of a generation size s as follows:

$$\begin{bmatrix} \mathbf{Y_{k,1}} \\ \mathbf{Y_{k,2}} \\ . \\ . \\ . \\ \mathbf{Y_{k,s}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & ... & \mathbf{0} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{M_{s(k-1)+1}} \\ \mathbf{M_{s(k-1)+2}} \\ . \\ . \\ . \\ \mathbf{M_{s(k-1)+s}} \end{bmatrix}$$

—*Intermediate node operations.* Intermediate nodes perform re-encoding operations. When an intermediate node receives the first encoded packet of a given generation, i.e., its generation identifier differs from the one seen in earlier packets, the packet is cached in the NC buffer and a timer is started (note that one timer per generation is needed). The intermediate node then has to establish whether the subsequently received encoded packets belonging to the same generation are innovative. It thus applies the Gaussian elimination method and checks whether the rank of G increases when the new encoded packet is added to the buffer. If not, the packet is dropped, otherwise it is cached in the buffer. When G has full rank, the node performs decoding to retrieve the native packets belonging to the given generation. The native packets are then encoded again into one packet with a random independent coding vector, so that only one packet is forwarded. If the timer of the NC buffer associated to that generation expires before the matrix rank has reached s, the re-encoding operation is applied to the set of encoded packets buffered at the node and the new encoded packet is forwarded. Therefore, an intermediate node only forwards one re-encoded packet following the reception of at most s encoded packets. It is worth noticing that in our scheme we consider both cases when intermediate nodes can perform encoding only once or twice over the same generation.

—*Receiver node operations.* Upon receiving encoded packets, the innovative ones are cached for decoding. After building the decoding matrix G, its rank is checked. If G has full rank, then the receiver can recover all native packets in a given generation; an early decoding can be performed when a sub-matrix of G has full rank v, with $v < s$, then v out of s native packets can be retrieved, as explained in Section 3.

4.2 Simple Flooding

The simple flooding scheme we use as term of comparison is based on the IEEE 802.11 standard. When a node sends a broadcast packet, all its one-hop neighbors will receive it. Since all the neighbors need to rebroadcast the packet in their turn, their transmissions may occur at almost the same time, thus likely resulting in collisions.
To reduce the collision probability, we introduced a simple link-layer modification to the standard: after receiving the broadcast packet, we let the receiving node defer the retransmission by a small time, randomly chosen in the range [0, 10] ms.

4.3 Deferred Broadcast

The deferred broadcast scheme we consider includes the mechanism proposed in [13], which, for the sake of clarity, we briefly recall.
Upon receiving a broadcast packet, nodes store the information about the predecessor node[1] and, before taking a decision whether

[1]The information about the predecessor node is included by each node in the packet.

to rebroadcast it or not, they wait for a *hold-off* time T_{ho}, during which they listen to possible retransmissions of the same packet by other nodes. In particular, for each packet, the node counts the number of rebroadcast events from other nodes with different predecessors. If this counter exceeds a given threshold, the packet is dropped. We set this threshold to 2, since simulation results show that a greater value does not lead to any improvement. This choice is supported by the findings in [4], where it is stated that if a node rebroadcasts a packet heard more than twice, the extra area it can cover is only about 9%.
Furthermore, in order to ensure that only the farthest nodes from the source rebroadcast packets, we added a further check as an improvement with respect to [13]. Each node receiving a broadcast packet for the first time records the received power level, P_{rx}. If this value is lower than a certain threshold, which we set equal to twice the receiver sensitivity, P_{th}, it classifies itself as a *border node* for that packet. A node which hears the same packet rebroadcast from other nodes with different predecessors exactly twice during its hold-off time, forwards the waiting packet only if it is a *border node*, otherwise it drops the packet.
The hold-off time, for which a node waits before a possible rebroadcasting, is computed according to the received signal strength, in such a way that nodes that are farther away from the sender compute a shorter delay and rebroadcast first. The hold-off time is given by:

$$T_{ho} = T_{max} - \frac{P_{rx} - P_{tx}}{P_{th} - P_{tx}} \cdot (T_{max} - T_{min}) - J \qquad (1)$$

where T_{min} and T_{max} are the minimum and the maximum hold-off time, respectively, P_{tx} is the transmission power, and $J \in [0, T_J]$ is a time interval used to prevent concurrent rebroadcasts from nodes that receive a packet with the same power level, P_{rx}.
For the sake of simplicity, in our simulations we will assume that all nodes have a common radio range and receiver sensitivity; also, we will use the following parameter settings: $T_{min} = 40$ ms, $T_{max} = 100$ ms, $T_J = 30$ ms, and $P_{tx} = 0.28$ W (i.e., the default value of transmission power in ns-2).

5. PERFORMANCE EVALUATION

5.1 Simulation Settings

To assess and compare the performance of the three broadcasting schemes discussed above, we implemented them in the network simulator ns-2. At the MAC layer we used the 802.11 protocol with a data rate of 1 Mb/s. We consider one source node whose application layer generates CBR traffic; messages have all the same size, equal to 1000 bytes, and the traffic rate is fixed to 50 kb/s. We also experimented with higher bit rates that however yielded exceedingly high values of packet loss probability for the simulated scenarios. Furthermore, since in this work we are not concerned about the delay performance of the proposed network coding-based scheme, we set the maximum buffering timeout of the NC buffer to 1 s.
To analyze the performance of network coding-based broadcast, we focus on stationary nodes by considering 100 static nodes uniformly deployed within an area of 100×100 m.
Both the well known Two-ray Ground Model, which accounts for near-ideal propagation conditions, and the Rayleigh fading model [14], which models realistic propagation conditions, by accounting for multipath effects and time-varying channel conditions, are used as propagation models.
We perform our analysis through simulations, assuming nodes with homogeneous transmission range and deployed on a bounded area. The metrics used to evaluate the broadcast schemes are the delivery delay, the packet loss rate, the transmission fairness, and the overhead. Specifically the last metric we consider is defined as the ratio of the total number of bytes transmitted at the MAC layer by all nodes (including the source node) to the total number of bytes generated at the source node application layer. It accounts for the total number of bytes transmitted to broadcast a packet; hence, it can be related to the energy consumed for the broadcast transmissions by the whole network. In particular, in the case of broadcasting based on network coding, by recording the protocol overhead we can compare the reduction in transmitted bytes due to the encoding procedure against the additional bytes, included in the *RLNC-header*, needed for the decoding procedure.
Each simulation is repeated five times with different seed numbers and instances of random network topologies. The collected data are averaged over those runs.

5.2 Simulation Results

In the following, we present the performance of our network coding-based broadcasting for different values of generation size, namely $s = 2, 3, 4$, and we compare it with the results obtained through simple flooding and deferred broadcast.
Static nodes under the two-ray ground propagation. We start by looking at the results achieved in a static scenario, when the two-ray ground propagation model is considered. The average value of the packet delivery delay, which is defined as the time elapsed from the time instant when the packet is generated at the source node to the time instant when the packet is received by a destination node, is shown in Figure 3. We can observe that the delay decreases as the number of neighbors increases (as a result of the increased radio range). The reason for this behavior is that a wider radio range yields fewer hops, hence lower delay.
Such a decreasing trend can be noticed for each broadcasting scheme under study, except for the flooding scheme, where for highly dense network, i.e., neighborhood size equal to 28, delay increases because of the higher contention experienced by nodes when accessing the channel.
Moreover, we find that the values of delay for deferred broadcast are higher than the ones achieved with the simple flooding, due to the hold off time a node (i.e., a node capable of ensuring the most forward progress of the packet) has to wait before transmitting. Deferred broadcast outperforms simple flooding in terms of delay only under high density of nodes, for the reasons explained above.
Broadcasting based on network coding records the highest delay value compared to the other two simple schemes. Such a behavior is due to the additional time an intermediate node waits to get enough number of packets to be combined and encoded together before forwarding a packet. In the worst case, the waiting time equals the buffer time out of the network coding which we recall it is set to 1 s. However, delay values strongly decrease as the neighborhood size increases. For values of neighborhood size higher than 8, increasing the generation size yields to higher delivery delay. In fact, the receiver node has to wait for sufficient number of independent and innovative packets in order to perform decoding over a certain given generation. The higher is the number of packets to be encoded together, the higher is the delay.
Next, we focus on the packet loss probability recorded at the application layer. As shown in Figure 4, the packet loss probability decreases as the neighborhood size increases.

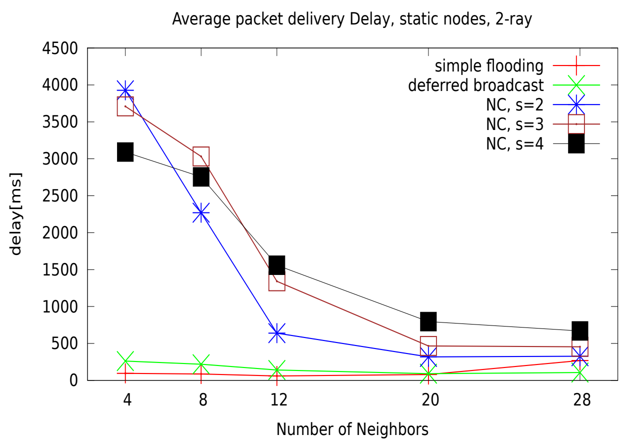


Fig. 3. Avg. packet delivery delay as a function of the neighborhood size

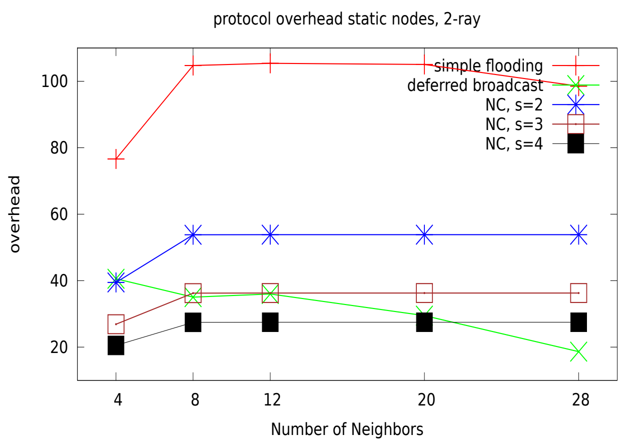


Fig. 5. Protocol overhead as a function of the neighborhood size

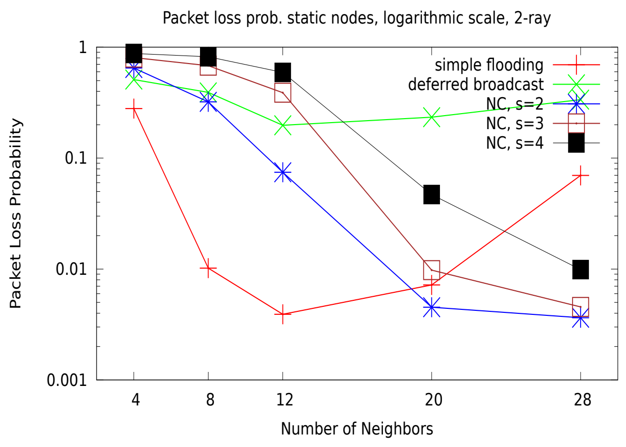


Fig. 4. Avg. packet loss probability as a function of the neighborhood size

Indeed, as the network becomes more and more connected, nodes can receive packets from different neighbors thus having more chance to receive all broadcast packets.
In the case of flooding, the minimum value of packet loss is recorded for neighborhood size equal to 12. For lower values of neighborhood size, higher values of packet loss probability are experienced because of rarely or scarcely connected network conditions. On the other hand, for neighborhood size higher than 12, achieved by increasing the radio range, the packet loss probability increases due to the higher number of collisions experienced at the MAC layer.
When compared to flooding and deferred broadcast, however, the broadcast scheme based on network coding suffers higher loss probability than the other schemes, in case of large values of generation size and poorly connected networks.
Interestingly, for large neighborhoods size the network coding-based scheme significantly outperforms simple flooding. The reason is to be found again in the higher collision probability at the MAC layer: the random node deployment may lead to very dense neighborhood where the medium sharing conditions become extremely crowded.
By focusing on the performance of the network coding-based scheme, we could notice that increasing the generation size yields the increasing of the loss probability. By increasing the generation size more packets are encoded and combined together. Hence, less packets are transmitted in the network, with less redundancy of the innovative packets at the receiver, which may not receive all the innovative packets in a given generation in order to perform the full decoding required to retrieve all native packets in that generation.
Instead, the performances of the deferred broadcast are negatively affected by the random deployment of nodes, in fact it records the higher packet loss probability.
Another metric of interest in wireless networks is the protocol overhead, which is presented in Figure 5. In the case of simple flooding, the overhead increases with the increase of the node radio range, since the larger the neighborhood size and connectivity degree of the network, the larger the number of performed transmissions. In fact, more intermediate nodes receive a broadcast packet and, in turn, act as forwarder of the received broadcast packet, this resulting in a larger number of performed transmissions.
Note that, in this case, low values of protocol overhead (i.e., small neighborhood size) correspond to high loss probabilities.
The higher packet loss probability recorded when network is highly dense (i.e. 28 neighbors/node) is directly related to the lower values of overhead. Indeed, since a multihop propagation occurs, the unsuccessful transmission of a broadcast packet at any of the forwarding nodes, because of collisions, may cause the packet loss in the whole network. Thus, if packets do not propagate any more in the network, the total number of transmissions is reduced, and as a consequence, the overhead metric decreases as well.
When, instead, the deferred broadcast technique is used, the overhead decreases as the node radio range increases. Indeed, for small values of node radio range, the network is scarcely connected and only few nodes can hear each other; it follows that several nodes must take part in the re-broadcasting of traffic packets. As the radio range increases, improving the network connectivity, many nodes refrain from re-broadcasting packets as they hear their neighbors transmitting first.
In all the deployed schemes, low values of overhead are recorded for sparsely connected network conditions (i.e., 4 neighbors/node), because only few nodes can hear from each other.
As for network coding-based broadcast, by eliminating redundant data transmissions, it provides similar results to those achieved by deferred broadcast, although the performance of the network coding mechanism significantly depends on the generation size: the larger the generation size, the more packets are encoded into one, and the smaller the broadcast overhead.

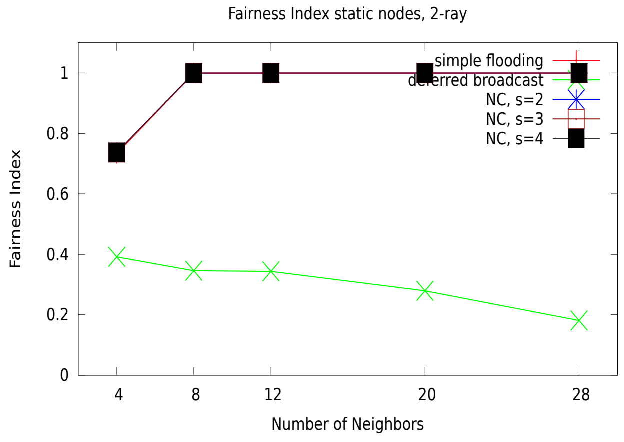


Fig. 6. Fairness index as a function of the neighborhood size

Finally, we want to evaluate the fairness among all nodes (except for the source) in terms of number of transmitted bytes, hence traffic load and energy consumption. To this end, we compute the well-known Jain's fairness index as $\left(\sum_{i=1}^{N} x_i\right)^2 / \left(N \sum_{i=1}^{N} x_i^2\right)$ where x_i represents the total number of bytes transmitted by node i for rebroadcasting the received packets, and N is the number of network nodes not including the traffic source.

By looking at the plots in Figure 6, we observe that the broadcast schemes based on simple flooding and network coding provide high index values for neighborhood size greater than 4. Worse fairness is recorded for neighborhood size equal to 4 due to lower network connectivity level (some nodes are not receiving nor retransmitting data).

Conversely, deferred broadcast yields poor performance in terms of fairness, especially for a large neighborhood size. This is due to the fact that, in deferred broadcast, the same set of nodes are selected as forwarders for all traffic packets, namely the nodes that ensure the greater spatial progress for the packets.

Rayleigh against two-ray ground propagation for static scenarios. We will now look at the results achieved by the Rayleigh propagation model, by comparing them with the ones achieved when the two-ray ground model is deployed. The two models exhibit almost the same behavior in terms of end-to-end packet delivery delay, with slightly lower values recorded for the Rayleigh propagation model, as depicted in Figure 7.

Due to the higher number of lost packets, both due to collisions and to channel-induced losses, packets may not propagate in some hops, as a consequence, packets experience lower contention, which directly translates in lower delays as compared to the two-ray ground scenario. Such a trend is noticed both in simple flooding and deferred broadcast. For network coding-based broadcasting smaller values of delay are experienced for low density scenarios, while for high neighborhood size higher delays are found with respect to the two-ray ground model. This is due to the fact that since several packets are lost due to the channel conditions, thus, more time is needed for waiting enough linearly independent encoded packets.

Figure 8 shows that the packet loss probability exhibits the same trend already discussed for two-ray ground scenarios with respect to the neighborhood size, for all the deployed schemes, and generation size, in case of network coding-based schemes.

Again the minimum value of packet loss probability is experienced by the flooding scheme under medium density conditions,

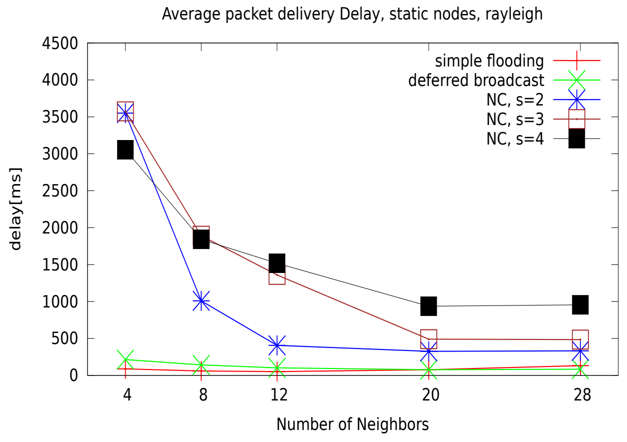


Fig. 7. Avg. packet delivery delay as a function of the neighborhood size

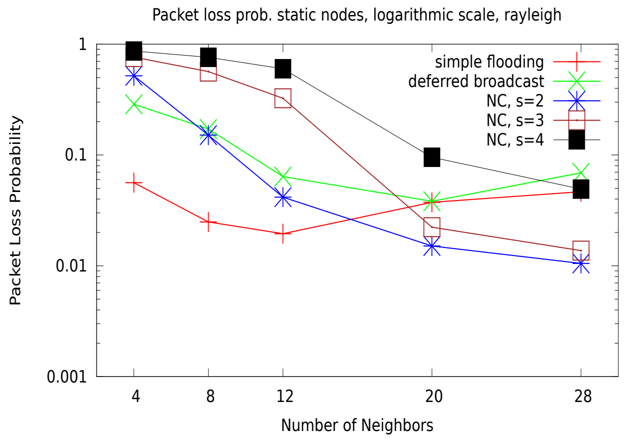


Fig. 8. Avg. packet loss probability as a function of the neighborhood size

i.e., neighborhood size equal to 12. The only remarkable difference, as we can expect, is that all the broadcasting schemes experienced worse packet loss performances with respect to the two-ray ground scenario, because of the additional channel-induced losses.

Surprisingly, even under adverse network conditions, i.e., when losses are both due to collisions and to the non-ideal propagation, the broadcasting scheme based on network coding manages to outperform simple flooding under high density network conditions, for every value of the generation size. Such a gain in terms of better packet delivery, is achieved again at the expenses of a higher delay, with respect to simple flooding. Another benefit perceived by deploying network coding is again a lower number of wasted network resources. Indeed, network coding maintains the overhead lower than in flooding, as depicted in Figure 9.

Again, simple flooding represents the worst scheme in terms of protocol overhead compared to other schemes. In addition, the same performances are experienced irrespectively of the neighborhood size for all schemes.

Furthermore, as for the two-ray ground model, network coding is able to ensure high fairness in terms of number of performed transmissions, Figure 10.

Flooding and network coding have the same behavior in terms of fairness, while deferred broadcast always shows a very bad fairness compared to previous two schemes.

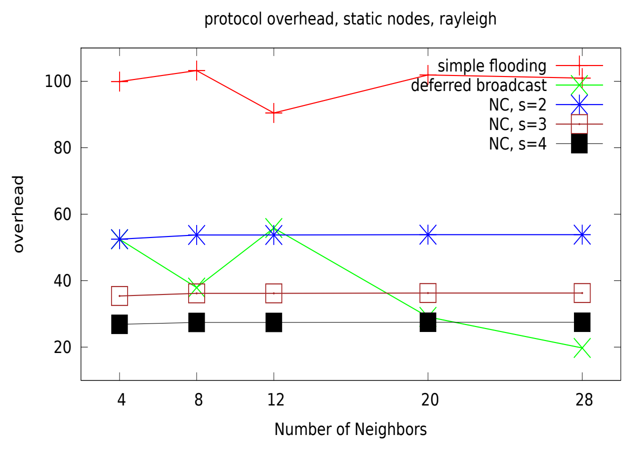


Fig. 9. Protocol overhead as a function of the neighborhood size

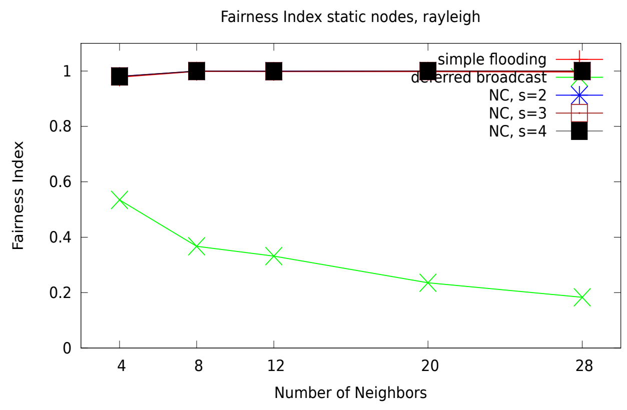


Fig. 10. Fairness index as a function of the neighborhood size

6. CONCLUSIONS

We developed a network coding-based scheme for broadcast traffic in ad hoc networks and compared its performance against simpler solutions, based on flooding and deferred broadcast. Simulation results, obtained through the network simulator ns-2, showed that the network node density and the generation size play a crucial role in the performance of network coding for broadcasting. It was also observed that network coding significantly outperforms other broadcasting schemes in terms of end-to-end packet loss probability and protocol overhead only for large neighborhood sizes (i.e., more than 12 neighbors) and generation sizes smaller than or equal to three.Finally, it was shown to achieve a better load balancing among nodes compared to deferred broadcasting. We remark that previous works on broadcasting based on network coding were carried out either through theoretical or simulative analysis in idealized settings, while our work accounted for the effects of a realistic MAC and physical layer underneath the coding layer. The protocol is able to reliably deliver broadcast packets even under adverse network conditions.

7. REFERENCES

[1] The Network Simulator, ns2.

[2] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

[3] Claudia Campolo, Claudio Casetti, C-F Chiasserini, and Saed Tarapiah. Performance of network coding for ad hoc networks in realistic simulation scenarios. In *Telecommunications, 2009. ICT'09. International Conference on*, pages 31–36. IEEE, 2009.

[4] Yuh-Shyan Chen, SY Ni, YC Tseng, and JP Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of MOBICOM*, 1999.

[5] Philip A Chou, Yunnan Wu, and Kamal Jain. Practical network coding. 2003.

[6] Xiaowen Chu and Kaiyong Zhao. Practical random linear network coding on gpus. In *GPU Solutions to Multi-scale Problems in Science and Engineering*, pages 115–130. Springer, 2013.

[7] Elena Fasolo, Michele Rossi, Jörg Widmer, and Michele Zorzi. On mac scheduling and packet combination strategies for practical random network coding. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 3582–3589. IEEE, 2007.

[8] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, 2006.

[9] Nishant Jain, Sanjeev Sharma, and Santosh Sahu. Efficient flooding for a large sensor networks using network coding. *International Journal of Computer Applications*, 30(9), 2011.

[10] Li Li, Ramachandran Ramjee, Milind Buddhikot, and Scott Miller. Network coding-based broadcast in mobile ad-hoc networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1739–1747. IEEE, 2007.

[11] Wei Lou and Jie Wu. On reducing broadcast redundancy in ad hoc wireless networks. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2003.

[12] Takahiro Matsuda, Taku Noguchi, and Tetsuya Takine. Broadcasting with randomized network coding in dense wireless ad hoc networks. *IEICE transactions on communications*, 91(10):3216–3225, 2008.

[13] Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 108–119. ACM, 2006.

[14] Ratish J Punnoose, Pavel V Nikitin, and Daniel D Stancil. Efficient simulation of ricean fading within a packet simulator. In *Vehicular Technology Conference, 2000. IEEE VTS-Fall VTC 2000. 52nd*, volume 2, pages 764–767. IEEE, 2000.

[15] Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. Prentice Hall PTR New Jersey, 1996.

[16] Parastoo Sadeghi and Mingchao Yu. Instantly decodable versus random linear network coding: A comparative framework for throughput and decoding delay performance. *arXiv preprint arXiv:1208.2387*, 2012.

[17] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *Computers, IEEE Transactions on*, 52(5):545–557, 2003.