Combination of Intra-Flow Network Coding and Opportunistic Routing: Reliable Communications over Wireless Mesh Networks

Pablo Garrido, David Gómez, Ramón Agüero
University of Cantabria
Avda. De Los Castros s/n
Santander (Spain)
{pgarrido, dgomez, ramon}@tlmat.unican.es

Joan Serrat
Universitat Politècnica de Catalunya
C/ Jordi Girona, 1-3
Barcelona (Spain)
serrat@tsc.upc.edu

ABSTRACT

Opportunistic routing has recently appeared as a technique aimed to increase the performance of wireless mesh networks, by taking advantage of the broadcast nature of the wireless medium. Despite the remarkable attention the research community has paid to it, there are still some issues that need to be addressed; one of the most relevant ones is the unnecessary forwarding of the same packet by a number of nodes. Since *Random Linear Coding (RLC)* mechanisms randomly mix packets before forwarding them, they can be exploited to avoid (or at least to minimize) the aforementioned problem. In a previous work we introduced a flexible Network Coding (NC) entity that we integrated within the `ns-3` framework. We extend herewith its functionalities, by integrating an opportunistic routing module that enables it to be used over random topologies. In addition, we assess the performance of using different external algebraical libraries to carry out the coding/recoding/decoding operations (i.e. matrix inverse and rank calculation).

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Simulation, Network Coding, Mesh Networks, Opportunistic Routing

Keywords

Network Coding; Reliable Communications; Lossy Wireless Channels

1. INTRODUCTION

In recent years, a rekindled interest has come back for the design and performance analysis of wireless mesh networks, since they might broaden the coverage area of more traditional accesses, without incurring in a high cost. As a consequence, they are undergoing a constant and quick evolution, and they stand as one of the access alternatives to be explored in forthcoming wireless networking architectures. However, they need to deal with poor link quality, scarce resources and multiple interferences. Furthermore, while it goes without saying that future communications will be strongly dominated by wireless networks, the performance of the mainstream transport protocol at the time of writing, *TCP* (used for countless applications), is severely jeopardized when used over this type of networks.

Opportunistic routing [1] is a novel approach, recently proposed to be used over wireless networks, which has recently received remarkable attention from the research community. The corresponding techniques are able to offer high throughputs, even when links are lossy and exhibit a low quality. While traditional routing schemes select a single path to send the data through the network, opportunistic routing establishes that any node that overhears a transmission could participate in the forwarding process, even if it did not belong to an already established route. On the other hand, opportunistic routing also leads to several challenges. In particular, when multiple forwarders overhear the same packets, they might unnecessarily retransmit the same information. In a nutshell, there is a trade-off between the objectives of masking errors by means of redundant transmissions and maximizing the overall performance. One interesting alternative to overcome such issue is the use of NC techniques, which randomly mix the received packets before forwarding them. In this sense, forwarders that overhear the same packets do not necessarily retransmit the same information.

In a previous work [2] we presented a fully fledged implementation of a NC module that was integrated within the **ns-3** framework. It includes two complementary approaches: a solution based on the combination (coding) of segments belonging to different TCP connections (*Inter-flow NC*) as well as an *Intra-flow* scheme that mixes packets belonging to the same flow. Both of these two solutions offer a reliable transport service. In such work we identified a

SIMUTOOLS 2015, August 24-26, Athens, Greece

DOI 10.4108/eai.24-8-2015.2261115

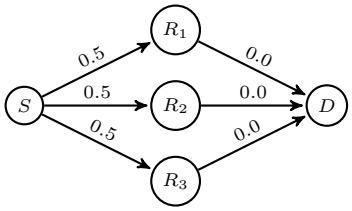


Figure 1: Simple Network Example

number of aspects that we wanted to tackle in our future research and in this paper we introduce some of the new features that have been integrated within the module. First, we have reduced the computation time of the matrix rank and inverse calculations, by using a new external library (*M4RI-M4RIE*) for all the coding, recoding and decoding tasks, which are based on Galois-Field ($GF(2^q)$) algebra. In addition, we have implemented the opportunistic routing scheme originally proposed by Chachulsky *et al.* [3]; they introduced *MAC-independent Opportunistic Routing and Encoding (MORE)*, a holistic solution that combines the NC and the opportunistic routing paradigms. The main goal is therefore to complement and extend these previous works, by systematically assessing the performance of *intra-flow* NC techniques over random topologies, exploiting the advantages offered by opportunistic routing, by means of extensive simulation campaigns.

This document is structured as follows: Section 2 discusses some of the approaches that have been made in order to enhance the performance of traditional transport protocols (in particular, TCP) when used over wireless mesh networks, focusing on the use of NC or opportunistic routing solutions. Section 3 briefly describes the NC functionality that is included in our module, while Section 4 presents the potential benefits that the use of opportunistic routing might bring about. Section 5 depicts the changes that have been made in the NC module within the **ns-3** framework, discussing the additional functions that we developed so as to integrate MORE. Section 6 introduces the simulation setup that was used to assess the performance of the proposed scheme, discussing the most interesting results. Finally, Section 7 concludes the paper and advocates some aspects that will be tackled in our future work.

2. RELATED WORK

Although the research on routing protocols for Wireless Mesh Network (WMN) started twenty years ago, the relevance of such topologies has recently increased. During the latest years, WMN are proposed as a means to extend the coverage of more traditional access topologies and also appear with *Device-to-Device (D2D)* communications. However, there exist several issues that need to be overcome. For instance, the most relevant transport protocol, TCP, exhibits a poor performance over wireless networks [4], [5], mainly due to its misinterpretation of random packet losses.

There have been many proposals to enhance the performance of communications over WMNs. Some of the most relevant approaches are NC [6] and opportunistic routing [1, 3], which exploit the inherent broadcast nature of the wireless medium. This allows a packet transmission to be heard by other neighbouring nodes, besides the one the packet was originally addressed to.

In opportunistic routing, a set of intermediate nodes, the so-called **Candidate Set (CS)**, are selected as potential forwarders. They will forward packets according to some established criteria, as opposed to traditional uni-path routing schemes, which select the next-hop (a single node) before actually transmitting the packet. A different priority value is assigned to each forwarder belonging to the **CS**, according to a certain cost. This could consider a number of different aspects, such as the distance (number of hops) to the destination, the power consumption or the Expected Number of Transmissions (ExNT), among others. Based on that priority, the node is able to establish a probability of transmitting a packet (or likewise, an average number of transmissions); the lower the cost, the higher the probability. All in all, this mechanism allows a packet to dynamically decide the path it flows through.

Figure 1 shows a simple network topology where the use of opportunistic routing might increase the network performance; the source S sends a flow to the destination node D. As can be seen, each of the links is characterized by a certain loss probability. A traditional uni-path routing scheme would select a single forwarder node, for instance R_2, and the communication path would therefore be $S \rightarrow R_2 \rightarrow D$. The overall loss probability of this traditional path would be 0.5%, and it would be thus necessary transmitting two packets to guarantee, on average, a single reception. On the contrary, opportunistic routing might select the three nodes as simultaneous potential forwarding entities, and, assuming perfect coordination between nodes (i.e. there are no collisions), the end-to-end loss probability would be reduced to $(0.5)^3 = 0.125$.

Opportunistic routing needs to deal with three main problems, as discussed in [7]: (1) choice of an appropriate opportunistic metric, (2) identification of an algorithm to select the best candidates, and (3) selection of a method to enable the required coordination between them. There are various works that have addressed these questions from different perspectives.

Biswas and Moris [1] proposed *ExOR* in 2004. It uses the Expected Transmissions (ETX) metric [8] and requires strict scheduling, since each forwarding node uses a reserved time slot to send an acknowledgment upon a new packet is received, sorted by the corresponding ETX metric. When the forwarder with the highest priority does not correctly receive the packet, an alternative relaying entity realizes (absence of the ACK) that the packet was not forwarded and will therefore retransmit the packet itself. It is worth highlighting that this solution requires modifying the legacy *MAC* level implementation.

Other proposals use different metrics; for example, Zorzi *et al.* propose in [9] a local metric, the so-called Distance Progress (DP), which considers just the local information provided by each of the nodes, in particular, the distance towards the destination. However, it does not take into account the quality of the links. More recently, a more advanced metric, Expected Distance Progress (EDP) [10], was proposed as an enhanced version of the DP, overcoming such limitation.

The aforementioned opportunistic routing techniques need a strict scheduling. Network Coding, originally proposed by Ahlswede *et al.* [11], offers an interesting solution to avoid such constraint. If NC was integrated with opportunistic routing it might help to avoid transmitting duplicated packets by mixing/coding them.

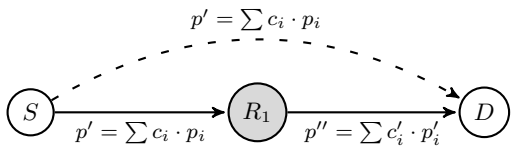


Figure 2: Intra-flow NC cannonical scenario

Last, but not least, it is worth mentioning that, despite the growing interest that opportunistic routing techniques have recently gathered, there are not many works that have used simulation frameworks to systematically analyze their performance. For instance, Yunfeng *et al.* [12] use a proprietary simulator to study the behavior of an enhanced MORE scheme. Furthermore, some other works [13] use **ns-2** to assess the performance of *COPE* [6]. To our best knowledge, there is not any study using the **ns-3** simulator.

3. INTRA-FLOW NETWORK CODING

The functionality of the intra-flow NC was presented in our previous work [2]. We introduced a intermediate new layer which lies between the network and transport levels. By means of the combination of a Random Linear Coding (RLC) scheme and UDP, we provide a reliable (*TCP-like*) service, providing an additional resilience to packet erasures over hostile network conditions. For a better understanding of the combined scheme, we describe below its main functionalities.

The NC entity at a source node receives the information from the upper layers and stores the datagrams at its *transmission buffer*. This container is different for each active flow, identified by the source and destination IP address / port tuples. Upon receiving K native packets of the same flow, the transmitter builds a random linear combination of them, $p' = \sum_{i=0}^{K-1} c_i \cdot p_i$. The corresponding random coefficients, c_i, are generated from a finite field $GF(2^q)$, and p_i are the native packets. These random coefficients are represented as a *coded vector*, $\vec{c} = (c_0, c_1, \cdots, c_{K-1})$. If this entity does not receive the required K packets before a *transmission buffer timeout* expires, the combination would be built anyway, using all the stored datagrams ($< K$). The source periodically sends coded packets, until the destination node confirms the successful reception and decoding of the corresponding block, i.e the K native packets.

When the receiver entity receives an "innovative" packet, the corresponding coefficient vector will be kept within a matrix, $\mathcal{C}$, while the corresponding coded packets are stored in another buffer. After receiving K linearly independent vectors, the coefficient matrix rank equals K and then we can calculate its inverse, $\mathcal{C}^{-1}$. This is used, together with the stored coded packets, to decode the native information, $P = \mathcal{C}^{-1} * P^{-1}$, recovering the K original packets, which are afterwards delivered upwards. Finally, an ACK message is sent to the source node, confirming the successful decoding of the whole block.

The intermediate nodes can play two different roles: in a (Random Linear Source Coding (RLSC)) solution, they use the traditional store-and-forward scheme, and all the coding tasks are carried out just within the source nodes, and the intermediate entities only forward the coded packets. In the enhanced Random Linear Network Coding (RLNC) solution, the intermediate nodes participate in the coding tasks by *re-coding* the previously stored packets before transmitting them; in this sense, the information is modified as it traverses the network. In addition, the intermediate node checks (as was done by the receiver) whether the coded packet is linearly independent from those already stored to silently discard it otherwise. When the intermediate node gets a transmission opportunity, a new coded packet, p'', is built, before delivering it to the lower layers. It can be seen that p'' is actually a linear combination of the original native packets. In addition, when a relaying node receives an ACK, or a coded packet with a greater block (sequence) number, it discards all the stored packets belonging to an already received block, starting the process again for the new block.

As was mentioned earlier, when either the source or a relaying node receives an ACK, the NC entity sends a signal (cross layer scheme) to the MAC layer, instructing it to remove all the frames belonging to such flow that are waiting to be transmitted from its transmission buffer. In this way, we avoid transmitting useless frames.

Figure 2 shows an illustrative example of both operations. The source S sends packets to the destination node D. If the RLSC scheme was used, then the relay node, R_1, would only forward the coded packets. In any case, we lower the constraints that would be imposed to a traditional end-to-end transport protocol, since the destination does not need to acknowledge every single reception, but just the whole block, provided that all the linearly independent packets carry the same amount of information. We can therefore conclude that there is a approximate gain of $K - 1$ transmissions.

On the other hand, using the RLNC scheme, the forwarder node will build new linear combinations of the already coded packets it had previously stored. Although S is using R_1 to reach the destination, it might happen that, due to the intrinsic broadcast nature of the wireless medium, D could actually *overhear* packets p'_i directly transmitted from S. Under these circumstances, when the link between $R_1 \rightarrow D$ is not error-free, the fact that R_1 sends coded packets that are different from those transmitted by the source, would actually increase the probability of receiving innovative packets at D, compared to the original RLSC approach.

4. OPPORTUNISTIC ROUTING

The main objective of this work is to broaden our previous research [2] to apply the proposed NC techniques over random network deployments. In order to do so, we have implemented a set of new functionalities, carrying out some additional modifications on the existing code, following the MORE scheme proposed by Chachulski *et al.* [3].

MORE is a routing protocol for *stationary* wireless mesh networks, being particularly suitable for transmitting large files. MORE addresses some of the most relevant challenges of opportunistic routing techniques, since it establishes the number of packets each forwarder needs to send and when they should do it. The authors propose a heuristic approach to route a packet from the source, S, to the destination, D, being the main principle that, among those nodes that receive a new packet, the closest one to the destination should forward it onwards.

The proposal is a distributed solution where each node should know the loss probability for the links $\epsilon_{i,j}$, towards each of its neighbours. This information could be disseminated afterwards throughout the network by means of signalling messages (for instance, those used by link-state rout-

ing protocols). When a source wants to establish a route towards a particular destination, it uses the underlying graph, with all the loss probabilities, to calculate the ETX metric for all the potential path candidates.

4.1 Candidate Selection

As was mentioned earlier, the opportunistic routing approach used in this work is based on the initial contribution of Chachulski *et al.* [3]. We provide some preliminary concepts that are required to follow the description of the proposed scheme. Let z_i be the expected number of transmissions that node (*forwarding entity*) i should perform to route one single packet from the source, S, to the destination, D. We assume that nodes are sorted according to their distance towards the destination; i.e. if $i < j$ then i is closer to the destination than j and it thus has a lower ETX.

We can now calculate the average expected number of packets that j receives from nodes with a higher ETX using Eq. 1, where $\epsilon_{i,j}$ represents the frame loss probability over the $i \rightarrow j$ link. Node j only retransmits a packet received from a node with a higher ETX if it has not been received by other node with a lower ETX. Hence, the number of packets that j would eventually forward can be calculated as shown in Eq. 2; note that $L_s = 1$, since the source node originally generates all packets.

$$r_j = \sum_{i>j} z_i(1-\epsilon_{i,j}) \tag{1}$$

$$L_j = \sum_{i>j}(z_i(1-\epsilon_{i,j}) \prod_{k<j} \epsilon_{i,k}) \tag{2}$$

We can as well calculate the expected number of transmissions that j would perform (Eq. 3), considering that it continuously forwards each packet until a node with a lower ETX receives it.

$$z_j = \frac{L_j}{(1-\prod_{k<j}\epsilon_{j,k})} \tag{3}$$

A forwarding node might not be aware of new packets transmitted by the source, if it is not within its coverage area. In order to deal with this, the $TXcredit$ parameter was proposed; it is defined as the number of transmissions that a node should make for every packet it receives from a node with a higher ETX (i.e. its distance towards the destination is larger). Given that for each transmitted packet j receives $\sum_{j>i}(1-\epsilon_{j,i})z_j$, the $TXcredit$ of node i can be obtained using Eq. 4.

$$TXcredit = \frac{z_i}{\sum_{j>i} z_j(1-\epsilon_{j,i})} \tag{4}$$

To compute the $TXcredit$ for every node, the source uses Algorithm 1, originally proposed in [3], with a complexity of $\mathbb{O}(N^2)$.

4.2 Protocol

The source node divides the file into chunks of K packets and delivers information to the lower layers as was explained in Section 3. Afterwards, the sender executes Algorithm 1 and attaches the corresponding MORE header, which is shown in Figure 3, to every data packet. The header

for $i = n$ *... 1* **do**
  $L_i \leftarrow 0$;
end
$L_n \leftarrow 1$;
for $i = n$ *... 2* **do**
  $z_i \leftarrow L_i/(1-\prod_{j<i}\epsilon_{i,j})$;
  $P \leftarrow 1$;
  for j *= 2 ... i-1* **do**
    $P \leftarrow P \cdot \epsilon_{i,(j-1)}$;
    $L_j \leftarrow L_j + z_i \cdot P \cdot (1-\epsilon_{i,j})$;
  end
end

Algorithm 1: Algorithm to compute the $TXcredit$

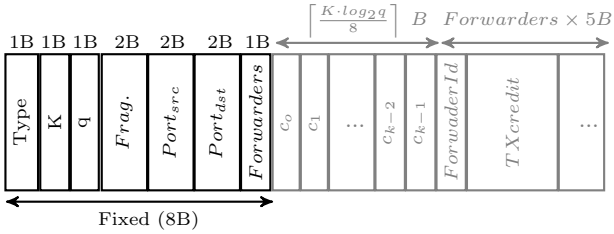


Figure 3: Simple Network Example

starts with a number of compulsory fields, which are used in every packet; the first elements are devoted to NC related information: `Type` distinguishes data packets from ACKs, K is the block size (number of packets), q indicates the size of the current Galois Field $GF(2^q)$, while $Port_{src}$ and $Port_{dst}$ fields facilitate the flow identification. The next field, `Forwarders`, indicates the number of nodes that could potentially relay the packet. This fixed-length part is followed by a number of optional fields (with variable length). The code vector is only used for data packets and indicate the coefficients that were used to build the coded packet; afterwards, the list of forwarders includes all candidates, sorted according to their distances to the source node. While the header is initialized by the source, the coded vector can be afterwards changed by each of the forwarding nodes.

Forwarding nodes keep a *credit counter*; whenever a node i receives a packet from a node with a higher ETX, it increases such counter with the $TXcredit$ of the corresponding header. When a forwarder needs to decide whether to transmit a packet, the node checks if the aforementioned counter is positive. In that case, the node creates a coded packet, broadcasts it and decreases the counter. We avoid infinite loops by establishing that a node could only forward a packet if it had received it from an upstream node. In addition, we also prune those nodes with a negligible contribution within the routing scheme, $z_i < 0.01 \cdot \sum_{j \in N} Z_j$[1].

When the destination node receives K linearly independent packets, it follows the procedure that was described in Section 3, sending an ACK to the source. This confirmation will use the best path to reach the source, as it would happen using a traditional routing scheme.

In the next section we discuss the changes that were required within the NC entity to integrate the opportunistic routing scheme.

[1]In dense networks, the number of these nodes might be actually rather large

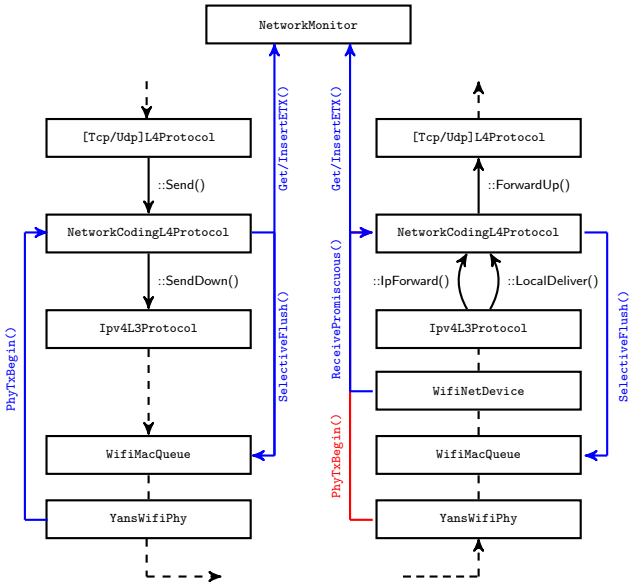


Figure 4: Transmission and reception/forwarding flow charts

5. UPDATES ON NS3 MODULE

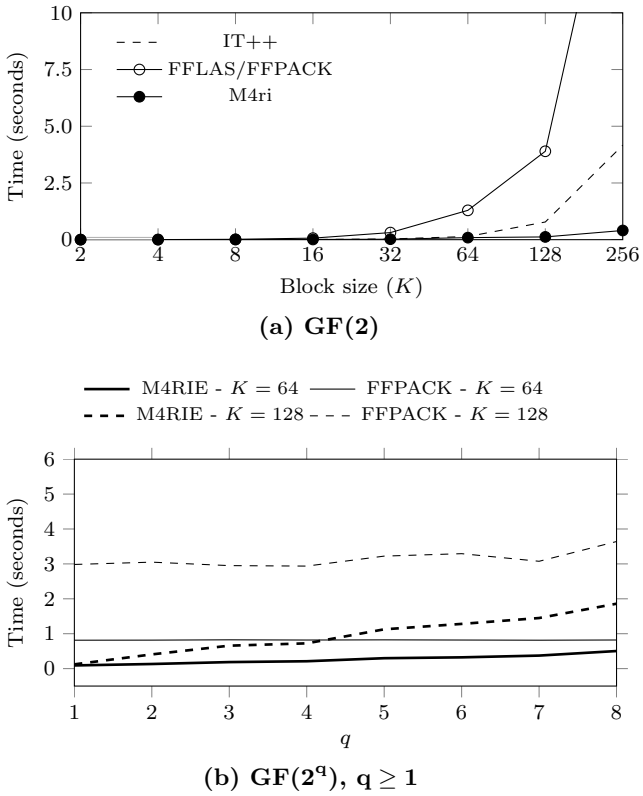
As previously mentioned, in [2] we described a flexible NC module, which we integrated within the **ns-3** framework. We developed an abstract class, **NetworkCodingProtocol**, which holds all the methods belonging to the two derived classes (**IntraFlowNetworkCoding** and **InterFlowNetworkCoding**). In order to integrate the opportunistic routing scheme, we modified the first one; in addition, we developed a new class, the so-called **NetworkMonitor**, implementing all the methods required for the opportunistic routing solution. In order to better understand the overall scheme, Figure 4 shows both the transmission and reception flows over the **ns-3** simulator. The diagram depicts the relationship between the protocol stack and the **NetworkMonitor** class, and also reflects a number of *cross-layer* connections that allow the system optimizing the performance [2].

NetworkMonitor is a singleton object, and there is therefore a unique instance that is seen by the NC entities at every node. When the scenario is initially generated, each node (i) stores, within a *ChannelMatrix* container, the loss probability of the link to each of the remaining nodes ($j \neq i$).

The connections between the NC layer and the **NetworkMonitor** entity are described below.

1. **InsertETX (IpAddress)**: when a transmitter node receives the initial packet for a particular flow from the upper layers, it uses this method to call the **NetworkMonitor** to create a vector with the ETX metric to the corresponding destination, comprising all nodes. This function uses the Dijkstra algorithm [14], using, as the cost for all links, the corresponding ETX metrics.

2. **GetETX (IpAddress)**: both the source and relaying nodes can afterwards call the **NetworkMonitor** instance to obtain the vector of ETX metrics to a specific destination address.



(a) GF(2)

(b) GF(2^q), q $\geq$ 1

Figure 5: Required time for the inverse matrix calculation

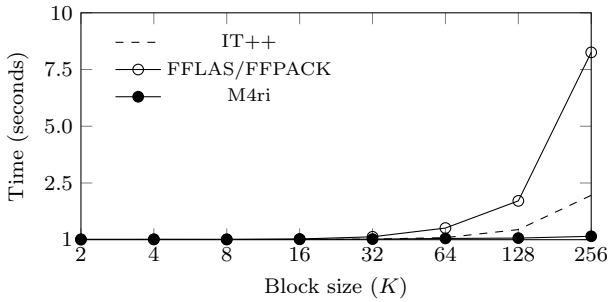
Besides, we use a static routing configuration to ensure that each node has a valid route towards the destination, by means of randomly selected neighboring nodes. It is worth highlighting that all these nodes overhearing the packet will decide whether to forward it based on the information carried in the header. On the other hand, the acknowledgment uses the shortest path from the destination to the receiver, which is calculated using the Dijkstra Algorithm and the ETX as the corresponding metric. The **NetworkMonitor** class collects all the information, as was mentioned before, and establishes the appropriate routes.

The most relevant changes that were made to the **IntraFlowNetworkCoding** class are briefly described below.

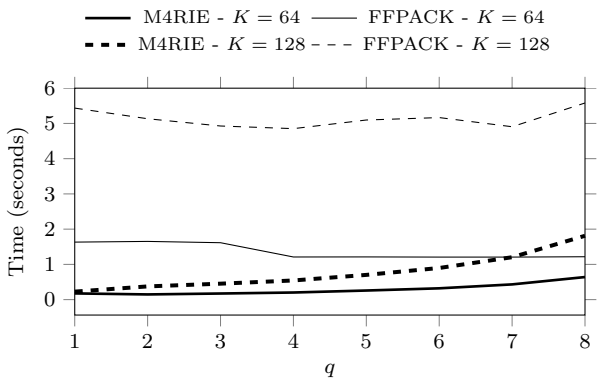
1. **Source operation**: the transmitter needs to calculate the number of forwarding nodes when a new data flow is created. The source gets the ETX metric vector from the **NetworkMonitor** instance and, using Algorithm 1, assigns a *TXcredit* to each forwarder.

2. **TXcredit counter**: each node has a new parameter. Upon the reception of a coded packet, the node increases the *TXcredit* counter, decreasing it each time the node transmits. As can be inferred, the source does not make use of this value.

3. **Forwarders**: in a traditional scheme [2], a packet would not reach the transport layer unless its IP header destination address matches that particular node's. This behavior is not appropriate for the NC operation, since the intermediate nodes play a key role. We needed to tamper the legacy **Ipv4L3Protocol::IpForward()** func-



(a) GF(2)



(b) GF(2^q), q ≥ 1

Figure 6: Required time for the rank matrix calculation

tion source code, to allow the NC layer to decide what to do with every incoming packet. Furthermore, an intermediate node can act as a forwarder, even when it does not directly receive a packet (several nodes can overhear the packet in one single hop, as was shown in Figure 1). Hence, intermediate nodes shall enable their *promiscuous* reception mode, allowing the NC layer to get these packets from the **WifiNetDevice** entity.

In the previous version, when a forwarder entity overheard a packet, it would *always* retransmit a (coded or recoded) packet. In the enhanced implementation, and when the medium is idle, a forwarder transmits a packet only if its *TXcredit* is positive. Afterwards, the node correspondingly decreases its *TXcredit* (one unit).

4. **Coding and decoding tasks**: in [2], all the *GF-related tasks*, i.e. vector multiplication, matrix rank and inverse calculation, were carried out by means of two external libraries: *FFLAS-FFPACK* [15], used when the calculations were over a generic Galois field, $GF(2^q)$ ($q > 1$); and *IT++* [16], which only works over a binary Galois Field $GF(2)$. Due to their high computational cost of coding and decoding operations, we have integrated a quicker external library. The two aforementioned alternatives have been replaced with *M4RI-M4RIE* [17]. It provides fast arithmetic with dense matrices over $GF(2^q)$ for $1 \leq q \leq 10$.

   In order to assess the efficiency of the different libraries, we generated 1000 random matrices, studying the average time consumed by both rank and inverse operations. We have used different configurations, $K = \{2, 4, 8, 16, 32, 64, 128, 256\}$ and $q \in [1, 8]$. Figure 5a compares the time required for the matrix inverse calculation by each external library. For the $GF(2)$ configuration, the three libraries can be used and the corresponding times are shown in Figure 5a. In addition, Figure 5b compares the results for generic Galois Fields, with $K = 32$ and $K = 64$ for the sake of clarity. In both cases, *M4RI-M4RIE* exhibits the best performance, regardless of K and q values. The same conclusion is yielded from Figure 6, which shows the rank matrix calculation time.

Table 1: Simulation parameters

Feature	**Value**
Physical link	IEEE 802.11b (11 Mbps)
Error model	RateErrorModel (modified)
Coverage area	20m disk radius
FER values	[0: 0.1 : 0.6]
RTS/CTS	Disabled
IEEE 802.11 RTX	1
Transport level	UDP / TCP "New Reno"
Application	OnOffApplication (20 MB)
App. data rate	CBR (11 Mbps)
Packet length	Max size allowed (MTU 1500B)
Traffic	Unicast
Simulations	1000 independent runs/point
Scenario	32 Nodes (Random)
K	64

The last update worth mentioning is the one that was required in the **NetworkCodingHeader**, adding those elements required by the opportunistic routing scheme: `Forwarders`, to indicate the number of forwarders; as well as the list of nodes that could potentially take that role.

6. SIMULATION AND RESULTS

Once we have described the opportunistic routing scheme and the changes that were required in the NC module, we discuss in this section some results that shed light on the performance enhancements that are brought about by the combination of these two techniques, after carrying out an extensive simulation campaign.

The most relevant parameters of the simulation setup are summarized in Table 1. In a nutshell, the physical and MAC layers are specified by the IEEE 802.11b specification (using the **ns3::YansWifiPhy** model provided by the simulator). We disable both the **RTS/CTS** exchange and 802.11 retransmission schemes. Besides, we use a modified **ns3::RateErrorModel** to arbitrarily fix the Frame Error Rate (FER) over the different links, randomly establishing a value between 0.0 and 0.6. Regarding the application at the source nodes, a Constant Bit Rate (CBR) flow is used, ensuring that there is always at least one packet waiting to be transmitted at the corresponding buffers, thus establishing a saturated scenario, where the bottleneck is at the wireless channel.

Network scenarios are generated by randomly deploying 32 nodes over a squared area, following a *Poisson Point Process*. Furthermore, we discard all deployments whose subjacent graphs are not fully connected, thus ensuring that there is at least one path between any pair of nodes. The coverage

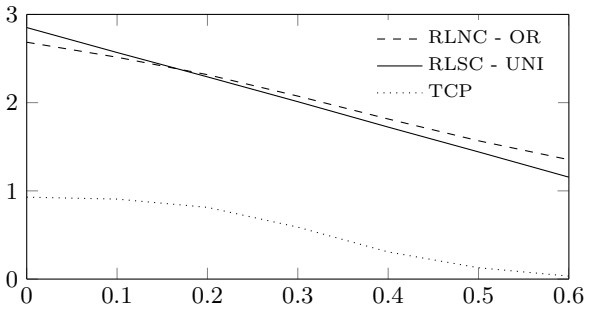


Figure 7: Performance comparison over a canonical scenario

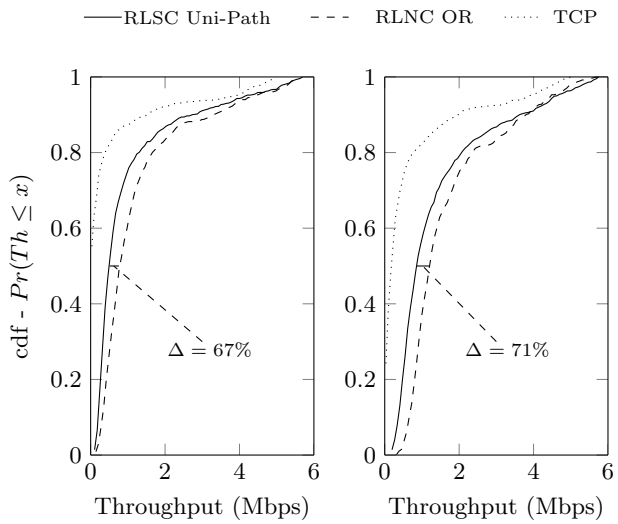
area of the nodes is modelled by a 20 meter disk radius, by correspondingly setting the **MaxRange** parameter of the **RangePropagationLossModel**.

Regarding the operational parameters of the *Intra-flow* coding scheme, we have selected the following configuration: $K = 64$, $q = 1$ and a *buffer timeout of* 100 *ms*. Although this does not guarantee the optimum performance, it shows an appropriate trade-off between throughput (it is only 3% lower than the best one) and computational complexity [2]. It is worth noting that UDP is used as the transport protocol layer, although the RLC scheme ensures a reliable communication, and all data packets reach their destination.

We will study the throughput measured at the receiver's application layer, defined as the total number of information bytes that were correctly received, divided by the required transmission time. The performances of three different alternatives are compared: a traditional TCP implementation (we are implementing a reliable service), an RLSC scheme without the opportunistic routing extension (we refer to this solution as *Uni-path*) and the opportunistic routing RLNC. In the first two alternatives, the relaying nodes are selected with the minimum-cost path between the source and the destination, established by the Dijkstra Algorithm [18]. It is worth mentioning that in the *Uni-path* solution the relaying nodes do not recode the information stored in their *transmission buffer*, and they simply forward the received packets.

First of all we assess the performance over a canonical scenario, as the one presented in Figure 1. In this case, the two links between the source and the two forwarding nodes are configured to have the same FER, 0.1, while the links between those and the destination exhibit a FER that is increased from 0.0 to 0.6. Although the *Uni-path* scheme provides a higher throughput when the links between the relaying nodes and the destination are ideal, the opportunistic routing scheme leads to a higher throughput than that achieved by the two other solutions, when the conditions of the links between the forwarding nodes and the destination get worse. The combination of opportunistic routing and network coding leads to a gain of almost three times as compared to the throughput offered by the traditional TCP.

In order to study the performance of the opportunistic routing scheme over random scenarios, Figure 8 shows the cumulative distribution function (cdf) of the throughput for two different node densities: the first one (see Figure 8a), which uses the parameters that were presented earlier, corresponds to a density of $3.2e-3$ nodes/m^2, while the second one (Figure 8b) was obtained over a network in which



(a) Scenario 10000 m^2 **(b) Scenario 5000 m^2**

Figure 8: cdf throughput

the density was twice as big as the previous one ($6.4e-3$ nodes/m^2), since we kept the same number of nodes, but we reduced the area to 70.71×70.71 m^2. We can see that the use of opportunistic routing is favored by higher node densities, since it increases the probability of finding advantageous forwarding nodes, leading to situations similar to those shown in Figures 1 and 2. The median of the throughput increases from 0.76 *Mbps* to 1.2 *Mbps*, yielding a gain of $\approx 63\%$.

In both figures we compare the performance achieved by opportunistic routing with the one that would have been obtained by means of a more traditional *Uni-path* scheme. The results show that opportunistic routing significantly increases the throughput. In particular, this gain is $\approx 67\%$ for the lower density deployment (see Figure 8a) and $\approx 71\%$ for the second network deployment (Figure 8b). We can see that, although the performance of the legacy approach is higher for higher densities (shorter routes can be used) the gain of the opportunistic routing scheme remains almost constant.

This relevant improvement is a consequence of the combination of intra-flow NC and the capability of taking advantage of the broadcast nature of wireless channels. As can be seen in Figure 9, the combined use of opportunistic routing and RLNC, where intermediate nodes recode the packets, brings about a performance that surpasses the one observed with both RLSC and RLNC over a single-path configuration. The combination of the RLSC scheme and opportunistic routing does not seem to bring additional advantages, since the forwarders would be transmitting the same information. Compared to other opportunistic routing solutions, the recombination at intermediate routers helps to reduce the complexity of the corresponding scheduler, since nodes are continuously generating new information from the previously stored packets; on the other hand, traditional solutions would need to handle the possibility of transmitting redundant messages.

Finally, we study the impact of using a greater Galois Field, $GF(2^q)$. Figure 10 shows the throughput as a function of the number of hops of the shortest path from the source to the destination. Although there exists a small gain when $q = 2$, it is almost negligible, and due to the higher

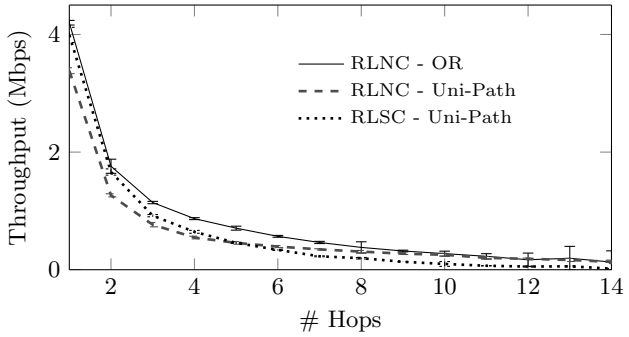


Figure 9: Throughput Vs. Number of hops

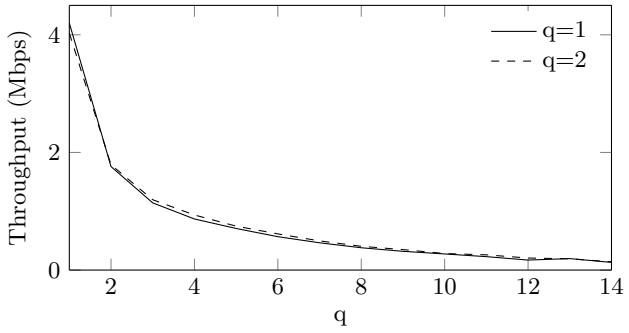


Figure 10: Throughput Vs. Number of hops for different q values

computational cost (as was discussed in Section 5), the use of larger q does not pay off.

7. CONCLUSIONS AND FUTURE WORK

In this work we have presented an updated version of a NC module which we implemented within the **ns-3** framework. We have extended its original scope, so that it can also deal with random network deployments. In order to do so, we have integrated the opportunistic routing scheme that was proposed by Chachulski *et al.*in [3], and we have exploited the advantages of combining the two approaches. Furthermore we have also enhanced the earlier implementation, by integrating a new library, $M4RI - M4RIE$, which clearly outperforms (in computational time) the previous alternatives regarding all the GF-related calculations (rank and inverse).

By means of an extensive simulation campaign, we have assessed the performance of the opportunistic routing scheme; on a first stage, we focused on canonical topologies, evaluating the throughput of these techniques, comparing it to that offered by more traditional routing mechanisms. Afterwards, we have extended our analysis to random topologies, where we have seen that the benefits of this alternative become more relevant when node density gets higher.

There are still a number of aspects that we will tackle in our future research. First of all, we will implement the signalling protocol (based on *Hello messages* or probes), which is required to enable a distributed operation of the proposed routing scheme. Another aspect that might be of interest would be to increase the complexity of the studied scenarios, for instance, starting more than one single flow at the same time.

Finally, it is worth highlighting that all the NC protocol's code, together with the opportunistic routing implementation, follow an open-source approach and can be found in [19]. We strongly encourage other interested researchers to download the code and use it, since this would certainly help us to incorporate additional enhancements. The repository also contains the files that are required to mimic the scenarios that were used, and thus allow studying the same results. We would also welcome people interested in joining this effort.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the Spanish government for its funding in the project "Connectivity as a Service: Access for the Internet of the Future - 1", COSAIF (TEC2012-38574-C02-01).

8. REFERENCES

[1] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 69–74, Jan. 2004. [Online]. Available: http://doi.acm.org/10.1145/972374.972387

[2] D. Gómez, R. Rodríguez, E. Agüero, and L. Muñoz, "Reliable Communications over Wireless Mesh Networks with Inter and Intra-Flow Network Coding," in *Workshop on NS3 (WNS3)*, Atlanta, USA, 2014.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 169–180, Aug. 2007. [Online]. Available: http://doi.acm.org/10.1145/1282427.1282400

[4] F. Lefevre and G. Vivier, "Understanding tcp's behavior over wireless links," in *Communications and Vehicular Technology, 2000. SCVT-200. Symposium on.* IEEE, 2000, pp. 123–130.

[5] M. Zorzi, A. Chockalingam, and R. R. Rao, "Throughput analysis of tcp on channels with memory," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 7, pp. 1289–1300, 2000.

[6] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 243–254, Aug. 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159942

[7] A. Boukerche and A. Darehshoorzadeh, "Opportunistic routing in wireless networks: Models, algorithms, and classifications," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 22:1–22:36, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2635675

[8] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wirel. Netw.*, vol. 11, no. 4, pp. 419–434, Jul. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-1766-z

[9] M. Zorzi and R. Rao, "Geographic random forwarding (geraf) for ad hoc and sensor networks: multihop performance," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 4, pp. 337–348, Oct 2003.

[10] A. Darehshoorzadeh and L. Cerda-Alabern, "Distance progress based opportunistic routing for wireless mesh

networks," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, Aug 2012, pp. 179–184.

[11] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[12] Y. Lin, B. Li, and B. Liang, "Codeor: Opportunistic routing in wireless mesh networks with segmented network coding," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, Oct 2008, pp. 13–22.

[13] Y. Yan, B. Zhang, J. Zheng, and J. Ma, "Core: a coding-aware opportunistic routing mechanism for wireless mesh networks [accepted from open call]," *Wireless Communications, IEEE*, vol. 17, no. 3, pp. 96–103, June 2010.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *NUMERISCHE MATHEMATIK*, vol. 1, no. 1, pp. 269–271, 1959.

[15] "FFLAS-FFPACK. Finite Field Linear Algebra subroutines package," http://www-ljk.imag.fr/membres/Jean-Guillaume.Dumas/FFLAS/index.html.

[16] "IT++ Mathematical library," http://itpp.sourceforge.net/.

[17] M. Albrecht, *The M4RIE Library – Version 20120613*, The M4RIE Team, 2012. [Online]. Available: http://m4ri.sagemath.org

[18] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pp. 225–227, 1990.

[19] D. Gómez, P. Garrido, and R. Agüero, "Network coding architecture source code and documentation (ns-3)," https://github.com/pgarridounican/ns-3.20-OpportunisticRouting-.git.