Improving wireless multicast communications with NC: performance assessment over a COTS platform

Pablo Garrido[†], Ramón Agüero[†]

[†] Dept. of Communications Engineering
University of Cantabria, Santander 39005, Spain
{pgarrido,ramon}@tlmat.unican.es

***Abstract*—Multicast services are believed to play a relevant role in next wireless networking scenarios. In this paper we exploit Tunable Sparse Network Coding techniques to increase reliability of multicast communications. We show that the proposed network coding scheme yields a better performance than state-of-the-art solutions, which are traditionally based on retransmissions. We first use a model to analytically compare the two approaches. Then, we validate and broaden this analysis by means of an experimental campaign over a testbed deployed with Commercial Of-The-Shelf devices. This platform, comprising low cost devices (Raspberry-PI), allows us to assess the feasibility of the proposed solution, which offers a relevant gain in terms of performance.**

***Index Terms*—Multicast Communications; Random Linear Network Coding; Sparse Network Coding; Measurements.**

I. Introduction

It is well known that multicast is an efficient mechanism for packet delivery in one-to-many applications. Their popularity has been significantly growing during the last years, and they are believed to play a fundamental role in next wireless networking scenarios, where services as video streaming will be prominent. However, most of the existing protocols are designed seeking reliability, and in-order delivery for unicast communications and they are thus not suited for multicast.

Hence, the research community has put more effort in providing new multicast solutions, as the NACK-oriented Reliable Multicast (NORM) protocol, which was designed to offer reliable multicast communications based on Negative ACKnowledgement (NACK)s. Besides, several protocols based on coding schemes have been also proposed, such as fountain codes (for instance, LT [1] or Raptor Codes [2]) or Network Coding (NC), initially proposed by Ahlswede *et al.* in [3]. Unlike fountain codes, NC brings greater versatility, where intermediate nodes can discard, or recode the information, and has been shown to offer a better performance over packet-erasure channels.

Random Linear Network Coding (RLNC) stands out, among other NC solutions, as the most widespread scheme, mostly due to its simplicity and good performance. Indeed, it hides losses form the upper layers [4], [5], reduces signaling overhead over opportunistic networks [6], and leverages efficient transmissions over wireless mesh networks [7]. On the other hand, RLNC was in fact designed to offer optimal multicast performance [8]. However, there are some voices questioning the use of RLNC, mostly due to its complexity. Tunable Sparse Network Coding (TSNC) techniques, introduced by Feizi *et al.* [9] propose a dynamic increase of the coding density as long as the transmission evolves, which can help to reduce the coding and decoding complexity.

In this work we propose using a coding scheme based on TSNC techniques to improve the performance of multicast communications. The proposed solution exploits a semi-analytical model to find an almost optimum trade-off between performance and decoding complexity. Another distinguishing aspect of this work is that it answers one of the most demanding issues in NC research, which has been mostly based on analytical models and simulation-based analysis. In this paper, we use a Commercial of-the-shelf (COTS) platform to assess the performance of the different solutions. The testbed is based on low computational power devices (Raspberry-PIs), which offer a perfect benchmark to analyze the potential of TSNC schemes. The results are also compared with legacy protocols, which are based on retransmissions.

The rest of the paper is structured as follows: Section II summarizes some preliminary concepts of RLNC. It also introduces the proposed solution, based on a sparse coding scheme, which alleviates the decoding complexity. We also model the behavior of a more traditional solution, based on retransmissions. Section III introduces the platform that was deployed to run the corresponding experiments, discussing the results that were obtained. Finally, Section IV concludes the paper, introducing some aspects that will be addressed in our future work.

II. Preliminaries and Motivation

In this Section we depict the initial RLNC approach, and a model than can be used to assess its performance. We then study the gain that might be yielded by a coding scheme, compared to a more traditional solution, based on retransmission requests, as the NORM protocol. Last, since the goal is implementing the coding scheme into devices with low computational power, Raspberry-PIs, we introduce a coding approach based on TSNC, which offers a performance alike the RLNC, but with a much lower computational cost.

A. Random Linear Network Coding - (RLNC)

RLNC scheme was originally proposed in [8], where nodes independently and randomly select linear mappings from input packets onto output links over some field, $GF(2^q)$. Ho *et al.* showed in [10] that this approach can yield the maximum multicast capacity over random networks with high probability. Besides, Trullols *et al.* computed the exact probability that a receiver can decode the information after receiving N coded packets [11].

In our previous work [5] we characterized the performance of RLNC over a single wireless link. The information is divided at the source node into generations, each of them containing k symbols of L bytes. Then, it transmits coded packets, which are built by the linear random combination of symbols belonging to the same generation, $pkt' = \sum_{i=0}^{k-1} c_i \cdot pkt_i$, where c_i are the random coefficients, selected from a Galois Field, $GF(2^q)$. These coefficients can be represented as a coding vector, $\bar{c} = \{c_0, \cdots, c_{k-1}\}$, which is included within a coding header in every coded packet, pkt'. On the other side, the destination constructs a decoding matrix, D, of size $k \times k$. Every coding vector, after being extracted from each coded packet, is added to the i^{th} row of such decoding matrix, provided it was linearly independent.

We can establish the probability of receiving a linearly independent packet, from the previously received ones, or similarly that such reception brings new information. This probability depends on the degree of freedom at the receiver, i.e the rank of the decoding matrix (r), and can be obtained by: $\text{Prob}_{r^+}^{\text{RLNC}} = 1 - \frac{(2^q)^r - 1}{(2^q)^k - 1}$. Hence, the average number of packets that need to be received to decode a generation is given by: $\overline{\#TX} = \sum_{i=0}^{k-1} \frac{1}{\text{Prob}_{r^+}^{\text{RLNC}}} = k + \alpha$, where α is a constant value, which does not depend on the generation size, when $k \gg 1$, but it varies with the Galois Field, as it is derived in [12]. For the binary case, $GF(2)$, $\alpha \approx 1.6$, being almost negligible as the Galois Field length increases. We can indeed see that the resulting overhead is rather small, and its relative impact can be reduced by increasing the generation size, k. It is also worth highlighting that for the RLNC coding scheme it is not important the individual packets that are received at the decoder, but getting enough packets to decode a generation. This is precisely one of the main advantages of the RLNC scheme, since it avoids the need for the feedback channel, and it yields optimum multicast performance.

In our implementation,the source node transmits $N = k(1+\rho)$ packets, where ρ is a configurable parameter that defines the system redundancy. Hence, ρ should be tuned according to the particular link conditions. The probability that a receiver can decode a generation after receiving N' coded packets, which was calculated by Trullols *et al.* [11], is given by:

$$\xi_q(k, N') = \xi_q^0 \left(\begin{bmatrix} N' \\ N' - k \end{bmatrix}_{2^q} + \right. \\ \left. + \sum_{i=1}^{N'-k} (-1)^i \binom{N'}{i} \begin{bmatrix} N' - i \\ N' - k - i \end{bmatrix}_{2^q} \right) \quad (1)$$

where $\begin{bmatrix} m \\ n \end{bmatrix}_q$ are the Gauss Coefficients and ξ_q^0 is the probability of decoding a generation after receiving k coded packets[1]

Lets assume that the quality of the link between the source and the i^{th} destination (packet erasure channel) is FER_i and that losses follow a random uniform distribution. Hence, we can calculate the probability of decoding a generation after transmitting N packets by the encoder:

$$\text{Prob}_{\text{dec}} = \sum_{j=k}^{N} \binom{N}{j} \text{FER}_i^j \cdot (1 - \text{FER}_i)^{N-j} \cdot \xi_q(k, j) \quad (2)$$

Finally, assuming that the whole information to be transmitted is divided into M generations, the probability of successfully complete a transmission is given by:

$$\text{Prob}_{\text{succes}} = (\text{Prob}_{\text{dec}})^M \quad (3)$$

Note that if the link with a receiver has a better quality than FER_i, then the probability of successfully receiving the information will be, at least, equal to $\text{Prob}_{\text{succes}}$.

B. Decoding Complexity

RLNC techniques have been shown to offer a good robustness against packet erasure wireless channels. However, the main arguments questioning the use of RLNC is their decoding complexity, which strongly depends on the Gaussian elimination algorithm, $\mathbb{O}(k^3)$. The decoding complexity is considerably high compared to other approaches (for instance, LT [1] or Raptor Codes [2]). One of the most interesting alternatives to reduce such decoding complexity is TSNC, originally proposed by Feizi *et al.* [9].

Sparse Coding Techniques, as opposed to RLNC, work by only combining a few symbols from each generation to build a coded packet. The coding coefficients are selected according to the following probabilistic scheme, where p is the probability for a coefficient to be zero in the coded packet. There is a clear trade-off, since increasing the sparsity level indeed reduces the decoding complexity, but at the cost of augmenting the likelihood of receiving linearly dependent packets, which do not provide new information. When $p = 0.5$ this probabilistic scheme is equivalent to the legacy RLNC.

Figure 1 shows the impact of the sparsity level and the generation size over the decoding probability. In the left Figure we fixed a sparse coding scheme, $p = 0.9$, and we changed the generation size. β is defined as the additional transmissions that are required to decoded the generation, $\beta = N - k$. We can see that the generation is decoded with greater probability and fewer extra transmissions, when it gets higher. Then, in Figure 1b, we keep the generation size ($k = 100$), and the results show that the higher the sparsity, the larger the overhead, since the number of extra transmissions that are required to decode the generation grows with p.

On the other hand, higher sparsity levels yield a decoding complexity reduction, as can be seen in Figure 2, which

[1] $\xi_q^0 = \xi_q(k,k) = \frac{(2^q)^{k^2}}{\left(2^{q^k} - 1\right)^k} \prod_{j=1}^{k} \left(1 - \frac{1}{(2^q)^j}\right)$

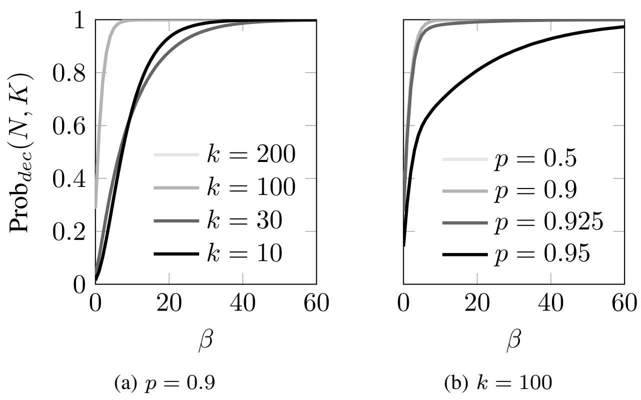


Fig. 1: Overhead, in terms of the extra transmissions (β) that are required to successfully decode a generation, for different number of generation sizes and sparsity levels

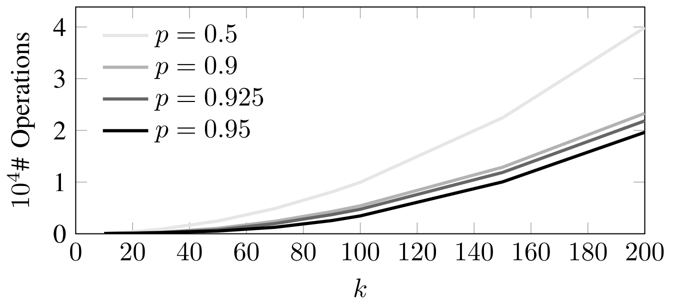


Fig. 2: Complexity of the decoding process (# of operations) Vs. the generation size for various sparsity levels

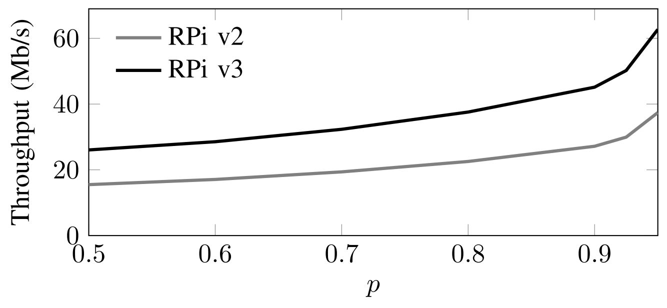


Fig. 3: Decoding throughput over different devices (RPi v2 and RPi v3) and for different values of sparsity level, p

shows the average number of operations that are performed by the receiver to decode a generation. Coding and decoding operations are performed exploiting the KODO library [15], which also offers a benchmark that counts the number of finite field operations. When using the binary Galois Field, $GF(2)$, the operations correspond with permutation and subtraction of rows from the decoding matrix. When the field size is higher, $2^q > 2$, then they involve constant multiplication, subtraction and permutation.

Note that there is a clear trade-off between decoding complexity and network overhead. Reducing the decoding complexity yields an increase of the decoding throughput. This might be quite relevant, since in computational low power devices, as the Raspberry-PIs, or sensors, it might become the performance bottleneck. In this sense, Figure 3 shows the decoding throughput, for two different versions of Raspberry PIs, where all decoding operations are performed within the devices themselves. To analyze the decoding throughput of the Raspberry PI, several coded packets are first built. Afterwards we measure the time required to copy those coded packets and decode the corresponding generation. As can be seen, the decoding throughput achieved by the newer version is higher. However, it is interesting to note that for the two devices, when RLNC is used ($p = 0.5$), the decoding throughput might be lower than network throughput, thus becoming the system bottleneck. On the other hand, if we exploit sparse coding techniques, this performance indicator could increase, $\approx \times 2.4$ for $p = 0.95$, in both devices.

In [12] we introduced a semi-analytical model that accurately captures the performance of Sparse Network Coding (SNC). It is based on an Absorbing Markov Process, $\mathcal{S}$, where the states are defined by the decoding matrix rank, i.e number of useful packets received, and the non-zero columns at such matrix (i.e. coefficients that have been already received). The model is characterized by the corresponding fundamental matrix $\mathcal{N} = (I - Q)^{-1}$, where Q is a matrix that includes the transition probabilities between the transient states. Since they are used in this work, we recall some of the results that were derived in [12], in particular Corollary 7:

Corollary 1: Probability of receiving a linearly independent packet. We define the set of states $\mathbf{s}(r)$ as all the states from the chain where the rank equals r, $\mathbf{s}(r) = \{(i,j) \in \mathcal{S} \mid i = r\}$. Hence, the probability of increasing the rank of the matrix when r independent packets have been already received can be calculated as follows:

$$\delta(r) = \sum_{\forall j \ |(r,j) \in \mathcal{S}} H(1, n_j) \cdot (1 - p_{r,j}(0,0)) \tag{4}$$

where n_j is the index corresponding to the (r, j) state and H is the transient probability matrix, which is defined by $H = (\mathcal{N} - I)\mathcal{N}_d^{-1}$, where $\mathcal{N}_d$ is a diagonal matrix with the same diagonal of $\mathcal{N}$.

In order to mimic the performance that would have been obtained for RLNC, we tune the sparsity as the transmissions evolves. We select the corresponding sparsity so as to guarantee that the probability of receiving an innovative packet is slightly smaller than the one that we would characterize the legacy RLNC: Prob$_{r^+}^{TSNC}(r, w) <= Prob_{r^+}^{RLNC}(r)(1 - \epsilon)$. The ϵ parameter establishes a trade-off between decoding complexity and network overhead. In this sense, higher ϵ yields higher sparse coding, but at the cost of larger probability of generating linearly dependent packets.

We include Algorithm 1 at the source node to implement such functionality. To ease the implementation we use a fixed sparse coding scheme, where every coded packet is built by combining w randomly selected packets, $W = \{pkt_{j_1}, pkt_{j_2}, \ldots, pkt_{j_w} | pkt_{j_k} \neq pkt_{j'_k}, \forall j_k \neq j'_k\}$.

Algorithm 1 Sparse Network Coding Transmission process

```
Input: $k$, $q$, $\rho$, $\epsilon$
 1: w = 1
 2: while TX < k(1+ $\rho$) do
 3:     TransmitRandomVector(k,w,q)
 4:     if rand() $\leq FER$ then
 5:         if rand() $\leq \text{Prob}^{\text{TSNC}}_{r+}(r, w)$ then
 6:             $rank_t = rank_t + 1$
 7:         end if
 8:     end if
 9:     while $\text{Prob}^{\text{TSNC}}_{r+}(r, c) \geq \text{Prob}^{\text{RLNC}}_{r+}(r) \cdot (1 - \epsilon)$ do
10:         if $w' < \frac{k}{2}$ then
11:             $w + 1$
12:         end if
13:     end while
14:     TX = TX + 1
15: end while
```

C. Protocol bases on retransmissions

NORM protocol is currently being developed within the Internet Engineering Task Force (IETF) Reliable Multicast Transport working group, [13]. It is conceived to provide a reliable end-to-end communication over generic IP multicast services. NORM uses a selective, negative acknowledgment (NACK), mechanism to yield the required reliability.

The main limitation for NORM scalability is the high overhead caused by the feedback traffic sent by the receiver set to ensure reliability. NORM uses probabilistic suppression of redundant feedback, based on exponentially distributed random back-off timers. The authors of [14] analyze the performance of this type of schemes. This allows NORM to scale well, yet keeping a reliable data delivery transport, and without jeopardizing the overall latency.

However, the performance of retransmission-based protocols is clearly impacted by the number of devices and the correlation between losses over the different links. We assume that the amount of information to transmit is $T(\text{bytes}) = M(\text{generations}) \times k(\text{generation size}) \times L(\text{symbols size})$. We further consider that all links are characterized by the same loss probability, $\overline{FER}$, and that they are uncorrelated. Under these circumstances the probability that all destination nodes successfully receive a packet is $(1 - \overline{FER})^R$. Hence, the probability of losing X packets is given by:

$$\text{Prob}_{\text{lost}}(X, T) = \binom{L}{X} \left[1 - \left(1 - \overline{FER}\right)^R\right]^X \cdot \\ \cdot \left[\left(1 - \overline{FER}\right)^R\right]^{(T-X)} \quad (5)$$

Since every receiver needs to get all packets, and losses are assumed to be uncorrelated between all receivers, the average number of lost packets after a complete transmission can be calculated as follows:

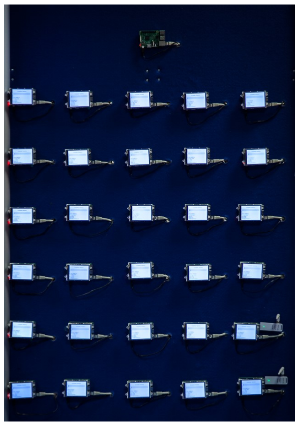


Fig. 4: COTS platform, with a source node, at the top, and 30 receivers (matrix of 6×5)

$$\overline{\text{TX}_{\text{NORM}}} = \frac{T}{(1 - \overline{FER})^R} - T \quad (6)$$

In order to ensure that all receivers successfully get all the information, the sender would need to retransmit every lost packet. In order to obtain a simple bound we would assume that retransmissions are not lost. In addition, it is worth noting that if the losses were correlated, the average number of lost packets would notably decrease, as will be seen in Section III.

III. Results

This section describes the experimental platform that has been deployed to carry out the corresponding measurement campaign. Then it shows some of the results that were observed, comparing them with the values that would have been obtained with the analytical models that were previously introduced.

A. Experimental Platform

As was briefly discussed before, one of the most relevant limitations of the research on Network Coding so far is the lack of results over real platforms. In order to address this, we deployed a testbed comprising low computational power devices (Raspberry-PI), which is used to assess the performance of Sparse Network Coding techniques. We have built a panel, which can be seen in Figure 4, comprising 31 Raspberry-PIs, configuring a multicast topology: one transmitter, placed at the top of the platform, and 30 receivers, deployed as an array of 6×5 Raspberry PIs v3. Every device has a wireless network interface (802.11), which is used to conduct the experiments, and a Ethernet connection, which is used to manage the platform (configuration, results retrieval, etc).

The Wireless LAN is configured in Infrastructure mode, where the node at the top of the platform takes the access point role, besides being the transmitter, as was already said. In order to limit the complexity of the scenario, we fix the devices to operate only in one wireless channel, and we also reduce the frequency at which beacons are sent. Furthermore, we limit the number of clients that can be associated to the access point, so that it ignores searching processes by other devices within the coverage area of the panel.

The coding scheme is implemented as depicted in Section II, exploiting the KODO library [15]. The transmitter broadcasts $N = k \cdot (1 + \rho)$ coded packets per generation. The receivers collect as many packets as possible, before trying to decode the generation. They keep track of the generations that they are able to decode. When this is not possible, the receiver registers this information, and will try to decode subsequent generations.

Since we are interested on measuring the performance of the proposed scheme over different channel conditions, and the Raspberry PI devices are at fixed locations, we have added a function that emulates packet erasures. Loss events are uniformly generated, with probability *FER*.

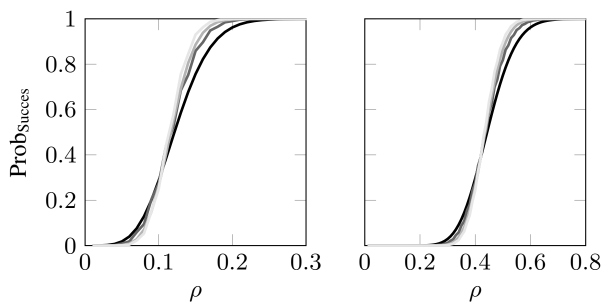
B. Results and Discussion

First of all, we analyze the probability of successfully receiving all the information, when using different redundancy values, as can be seen in Figure 5. We can see that the redundancy level that is required to achieve a high successful probability clearly depends on the link quality. On the other hand, the figures also show that such probability gets lower as we increase the number of generations, as was already seen in Eq. 3. We can thus decrease the redundancy by increasing the generation size and, consequently, transmitting the same amount of information with a fewer number of generations. Nevertheless, it is important to highlight that increasing the generation size would also yield larger coding vectors[2], and a bigger overhead. This trade-off was studied in [5], which concluded that if the network Maximum Transfer Unit (MTU) is 1500 bytes, the generation size should not be larger than 255.

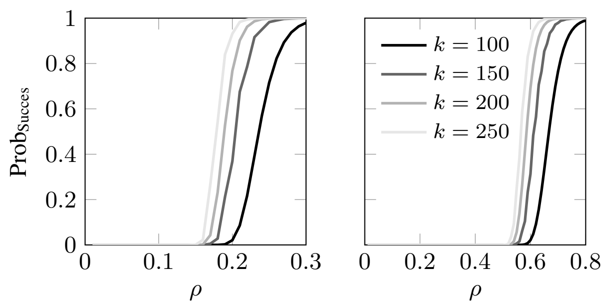
A comparison in terms of network overhead, between the RLNC scheme and a retransmission-based approach, NORM, is shown in Figure 6. For the RLNC scheme, we plot the redundancy level that should be used by the transmitter to yield a successful decoding event with probability 99.99%, using Eq. 3. This, as was previously discussed, does not depend on the number of receivers. For the retransmission-based schemes, we just consider the average number of lost packets by the different receivers during the first transmission, assuming that retransmitted packets are never lost.

So far, we have assumed that losses are uncorrelated. In order to assess the impact of correlated loss events, Figure 7, shows the evolution of lost packets, as the number of nodes is increased. In this case, the Raspberry PI does not longer discard any packet, but the transmitter is separated from the receivers, leading to an average FER ≈ 0.1. The results are obtained after the transmission of 100×100 packets by the source node. Every receiver keeps track of the lost packets' identifier, and the experiment is repeated 50 times. We also plot the bound that would correspond to the situation that has been considered so far, with uncorrelated losses, Eq. 6. We can

[2]The coding vector length, $\overline{c}$, equals $\frac{kq}{2}$



(a) $FER_i = 0.1$, $M = 1$ (b) $FER_i = 0.3$, $M = 1$



(c) $FER_i = 0.1$, $M = 100$ (d) $FER_i = 0.3$, $M = 100$

Fig. 5: Probability of successfully receiving the information Vs. system redundancy. The results are compared for different loss probabilities and generation sizes

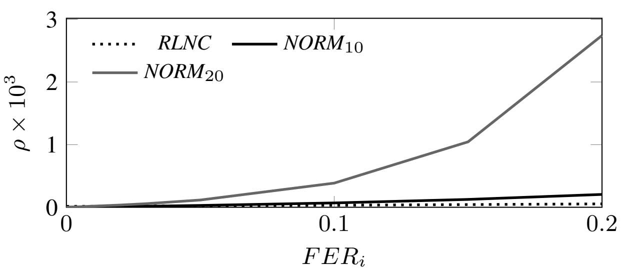


Fig. 6: Network overhead for RLNC and NORM Vs. link qualities

indeed see that, although receivers are very close to each other, the amount of lost packets gets higher as long as we increase the number of receivers, but the amount of retransmissions is quite lower than the bound.

Finally, we compare the RLNC approach with the sparse network coding proposal. We fix ϵ to 10^{-4}, parameter associated with the Algorithm 1, and we carry out 50 independent experiments. Figure 8 shows the probability of successfully receiving the information given a redundancy value, ρ. The overhead caused by the proposed sparse scheme is similar to the one exhibited by the legacy RLNC but, as shown in Table I, the decoding complexity is heavily reduced. Hence, we could assume a higher decoding throughput, or reduce the energy consumption of the devices. Table I also compares the overhead caused by the three solutions. Throughput is measured over a Raspberry-PI v3, and the average number

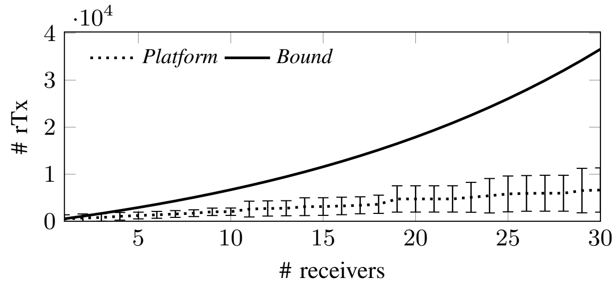


Fig. 7: Retransmissions observed over the platform and using bound (6) Vs. # of receivers

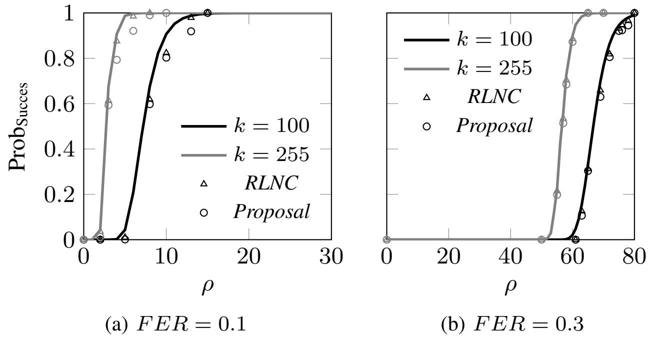


Fig. 8: Probability of successfully receiving the information, given a redundancy, ρ. The solid lines correspond to the theoretical values, while the experimental results are represented by markers

of transmissions are those needed to ensure a decoding probability of 99.99%, according to Eq. 3.

IV. Conclusions and future work

Despite the attention that has been recently paid to network coding, there are still very few works that have actually assessed its feasibility over real platforms. In this paper we have exploited a COTS tesbed, comprising 30 Raspberry-PIs, to analyze the behavior of a combination of multicast communications and network coding. We have compared the proposed scheme with a more traditional retransmission-based solution, showing that NC can indeed yield a large performance gain. Since the platform was based on low computational power devices, we have studied whether the use of NC schemes could be actually limited. In this sense, we have seen that the legacy RLNC imposes a great complexity. We showed that the use of a Tunable Sparse NC solution could actually overcome this drawback, without a relevant impact of the corresponding performance.

TABLE I: Performance comparison between the three different approaches

Scheme	**Thput**	**# Operations**	$\#\overline{\mathbf{Tx}}_{0.1}$	$\#\overline{\mathbf{Tx}}_{0.3}$
RLNC	26 Mb/s	$9.96 \cdot 10^3$	$4.1 \cdot 10^3$	$9.4 \cdot 10^3$
TSNC	91.24 Mb/s	$2.09 \cdot 10^3$	$4.1 \cdot 10^3$	$9.4 \cdot 10^3$
NORM_{30}	-	-	$2.26 \cdot 10^5$	$4.43 \cdot 10^8$

The work that has been presented in this paper will be broadened in our future research. First, the testbed that was introduced could be exploited to carry out more measurements, which could help us to better understand the full potential (and limitations) of NC when applied over real devices. On the other hand, we would like to use real traffic, for instance video streaming, to assess whether it can actually benefit from the use of NC. Last, we will also study different schemes to select the sparsity level of the NC solution.

Acknowledgements

This work has been supported by the Spanish Government (Ministerio de Economía y Competitividad, Fondo Europeo de Desarrollo Regional, FEDER) by means of the project **ADVICE** (TEC2015-71329-C2-1-R).

References

[1] M. Luby, "LT codes," in *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–280.

[2] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

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

[4] J. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," in *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, April 2009, pp. 280–288.

[5] D. Gómez, E. Rodríguez, R. Agüero, and L. Muñoz, "Reliable communications over lossy wireless channels by means of the combination of UDP and random linear coding," in *Proc. of the IEEE Symposium on Computers and Communications (ISCC)*, June 2014, pp. 1–6.

[6] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of the ACM conference on Applications, techonologies, architectures and protocols for computer communications*, vol. 37, no. 4. ACM, Aug. 2007, pp. 169–180.

[7] P. Pahlevani, D. E. Lucani, M. V. Pedersen, and F. H. P. Fitzek, "PlayNCool: Opportunistic network coding for local optimization of routing in wireless mesh networks," in *Proc. of the IEEE Globecom Workshops (GC Wkshps)*, Dec 2013, pp. 812–817.

[8] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc of the IEEE International Symposium on Information Theory*, 2003, p. 442.

[9] S. Feizi, D. E. Lucani, and M. Médard, "Tunable sparse network coding," in *Proc. of the Int. Zurich Seminar on Comm*, 2012, pp. 107–110.

[10] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[11] O. Trullols-Cruces, J. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *IEEE Communications Letters*, vol. 15, no. 1, pp. 67–69, January 2011.

[12] P. Garrido, D. E. Lucani, and R. Aguero, "A markov chain model for the decoding probability of sparse network coding," *IEEE Transactions on Communications*, vol. PP, no. 99, pp. 1–1, 2017.

[13] B. Adamson, C. Bormann, M. Handley, and J. Macker, "Rfc 5740: Nack-oriented reliable multicast (norm) transport protocol," *IETF*, 2009.

[14] J. Nonnenmacher and E. W. Biersack, "Optimal multicast feedback," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Mar 1998, pp. 964–971 vol.3.

[15] M. V. Pedersen, J. Heide, and F. Fitzek, *Kodo: An Open and Research Oriented Network Coding Library*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–152.