Mobile Netw Appl (2017) 22:1113–1123
DOI 10.1007/s11036-016-0731-7

Providing Reliable Services over Wireless Networks Using a Low Overhead Random Linear Coding Scheme

Pablo Garrido[1] · David Gómez[1] · Jorge Lanza[1] · Joan Serrat[2] · Ramón Aguero[1]

Published online: 30 April 2016

Abstract In this work, we propose a novel *intra-flow* network coding solution, which is based on the combination of a low overhead Random Linear Coding (RLC) scheme and UDP, to offer a reliable communication service. In the initial protocol specification, the required overhead could be rather large and this had an impact over the observed performance. We therefore include an improvement to reduce such overhead, by decreasing the header length. We describe an analytical model that can be used to assess the performance of the proposed scheme. We also use an implementation within the ns-3 framework to assess the correctness of this model and to broaden the analysis, considering different performance indicators and more complex network topologies. In all cases, the proposed solution clearly outperforms a more traditional approach, in which the TCP protocol is used as a means to offer a reliable communication service.

Keywords Random Linear Coding · Wireless networks · IEEE 802.11 · Protocol overhead · Packets erasure channels · Load balance control

1 Introduction

Wireless networks have undergone a continuous evolution in a number of aspects: users, devices, traffic demands, among many other elements. As a result of this, they have become the most widespread communication alternative, clearly surpassing last mile communications based on wired technologies. The fast roll-out of new technologies, such as Long Term Evolution (LTE), and the strong consolidation of other alternatives, as those based on the IEEE 802.11 standard, are some of the most relevant examples behind this trend.

Despite the essential role that wireless networks have been playing in the last years, outstanding as the most used access alternative, the most popular transport layer protocol, TCP, exhibits a poor performance when used over them. It is known that TCP was originally designed with the assumption that losses were (mostly) caused by the congestion at intermediate network routers, (almost) disregarding the presence of errors in the transmission between the source and the destination. This assumption is sensible for wired communications, which were the dominating technologies when TCP was originally conceived. However, in the wireless realm, packet losses are mainly caused by other circumstances, such as interference, propagation over hostile links, collisions, etc. This brings a remarkable loss of performance when TCP is used over this type of networks. With the aim to overcome this limitation, several works [19, 29] have studied the impact of these error-prone channels over the TCP performance. As a consequence, new variants of the legacy protocol have been proposed in the latest years, trying to eliminate, or at least mitigate as much as possible, the harmful effect observed over these links.

Among the various solutions that have been proposed to alleviate this problem, Network Coding (NC) out stands as one of the most promising ones. Starting from its definition itself, which questions the classical *store-and-forward* paradigm, the NC approach proposes adding a certain level of intelligence to intermediate nodes, allowing them to process and even transform the information as it crosses the

✉ Pablo Garrido
pgarrido@tlmat.unican.es

1 Universidad de Cantabria, Santander, Spain

2 Universitat Politècnica de Catalunya, Barcelona, Spain

network. In particular, this work focuses on one of the NC approaches, known as intra-flow NC, in which only packets belonging to the same flow might be coded together. We study the behavior of a Random Linear Coding (RLC) scheme that works together with the UDP protocol; their combined operation provides a reliable communication service. The main contribution of this paper is the proposal of a low overhead RLC scheme. We also broaden its earlier performance analysis, studying load balancing between various flows as well as the corresponding fairness.

The rest of this paper is structured as follows: Section 2 summarizes the most relevant studies that have focused on the improvement of the TCP performance over wireless networks using NC. In Section 3 we describe the scheme proposed in this work and introduce the analytical model that can be used to characterize its performance. Afterwards, Section 4 assesses the validity of such model, by comparing it with the results of a thorough simulation campaign that, in addition, is exploited to broaden the analysis. Finally, Section 5 concludes the paper, providing an outlook of those aspects that will be tackled in our future research.

2 Related work

Due to the growing importance of wireless networks and the massive use of TCP as the mainstream transport layer protocol, the scientific community has made a great effort to overcome the limitations exhibited by TCP over such scenarios. There exist as well other alternatives, ranging from modifications of the legacy TCP operation [16], making it more appropriate for wireless networks, to novel transport protocols, which address the problem from different angles, for instance the Stream Control Transmission Protocol (SCTP) [26]. Amongst them, one of the most promising techniques is the so-called NC.

The term Network Coding was originally coined by Ahlswede et al. in 2000 [2]. They proposed a novel routing approach that questioned the traditional *store and forward* paradigm in IP networks, with the integration of additional functionalities within different nodes, which can process and code the packets traversing them. Afterwards, several works have proposed the use of this technique, to get either performance enhancements or more reliable communications. Some initial works by Koetter [17] and Li [20] showed that the use of linear codes can bring the multicast maximum capacity, while Ho et al. [13] proved that random generation of linear codes brings the optimum performance with high probability, proposing the RLC scheme.

Several works have advocated the use of NC techniques to enhance the performance over wireless networks. One of the first proposals was the COPE protocol [15], where nodes code packets belonging to different information flows, combining them with a simple XOR operation. COPE exploits the broadcast nature of the wireless medium, since the neighbouring nodes are able to overhear packets not directly addressed to them. COPE was shown to reduce the number of transmissions, yielding a significant performance gain, but some other works [7], proved that its gain is much lower when the conditions of the wireless links get worse.

Following a different approach, Chachulski et al. proposed the MAC-independent Opportunistic Routing & Encoding (MORE) protocol [5]; rather than combining packets from different information flows, MORE codes packets belonging to the same data flow, resembling Digital Fountain solutions, such as LT [21] or RAPTOR [24]. MORE included a number of additional mechanisms to avoid unnecessary retransmissions at the relaying nodes. In order to do so, nodes estimate the quality of each link by means of echo messages (these are used by various routing protocols) and then decide whether or not to forward a packet based on such information. However, the authors did not consider the interplay with any transport protocol, and their analysis is mostly focused on the lower layers.

Other interesting solution, proposed by Sundarajan et al. [27], keeps TCP as the transport protocol, and improves its performance over wireless networks by integrating a coding module above the network layer. Besides, they proposed a proprietary acknowledgment mechanism, which is used by the destination to confirm the reception of meaningful information. The performance over lossy wireless links is clearly enhanced, compared with the traditional TCP, although the number of signaling messages is increased and this could bring significant overhead under more complex scenarios.

In previous works [8, 9] we have showed the trade-off between the different RLC operational parameters. Intra-flow network coding schemes have two main issues: *(1)* overhead due to the transmission of the random coefficients used by the protocol to combine the different packets; *(2)* the computational cost of the coding/decoding tasks. On the one hand, some works have proposed to use coding patterns, [11, 22], in order to reduce complexity in the decoding process or, on the other hand, to codify a fewer number of packets on each transmission, for instance [6, 10]. In the latter case, the authors also discussed the reduction on the required overhead.

In this work we introduce the combination of a RLC scheme and the UDP protocol using two different schemes. In a first approach the source node sends all the coefficients used to combine the packets; after assessing its performance, we introduce an enhancement to reduce the corresponding overhead, by only sending the seed used by the Pseudo-Random Number Generator (PRNG). We analytically study the additional gain brought by this reduced header, following an approach similar to that used by Trullols et al. [28]. It is expected that the improvement would be more relevant for larger finite fields and this is quite significant, since it is known that larger finite fields reduce the number of meaningless transmissions (linearly dependent combinations) [10]. In any case, it is quite important to study the impact of the finite field size, since larger values would also lead to longer decoding operations and a trade-off between these performance indicators would be therefore required.

3 Intra-flow protocol

This section depicts the NC scheme we have designed to offer a reliable communication service between two nodes (i.e. unicast transmission). As was already introduced, the proposed solution is based on the joint operation of UDP in the transport layer and a novel RLC entity, lying between the UDP and IP levels. Its implementation and assessment have been carried out over the `ns-3` [1] framework.

3.1 Protocol description

The protocol we are discussing in this work addresses the end-to-end communication between a pair of endpoints. In a nutshell, a transmitter sends random linear combinations of the K original packets of a single block until the destination has enough information to obtain the packets sent by the source's application layer. Afterwards, the transmitter deletes the already recovered block and starts the process with the next one. Below, we detail the operation of both the source and the destination nodes. In addition, we discuss two different approaches to carry the information concerning the coding process: whilst the former one is based on the transmission of all the coefficients that were used to code the outgoing packets, the second alternative aims at reducing the protocol overhead, by just sending a fixed-size random seed.

In an initial stage, a source node waits until it has received K packets belonging to the same data flow, which are temporarily stored in the transmission buffer; afterwards, it starts generating the corresponding coded packets, following the process illustrated in Fig. 1. As can be observed, a coded packet, p', is built from the K original (or *native*) packets that come from the application layer, $p_i, i \in [1, K]$, using a set of K random coefficients chosen from a Galois Field, $GF(2^q)$. These coefficients can be represented as a *coding vector*, $\overrightarrow{c_i}$, where $\overrightarrow{c_i} = \{c_1, c_2, ..., c_K\}$, which will have a key role in the decoding operation, since it carries the information needed at the destination to recover the original block of K packets.

One interesting characteristic of this approach is its the *rateless* property, since a source can generate an endless number of coded packets p' until the destination is able

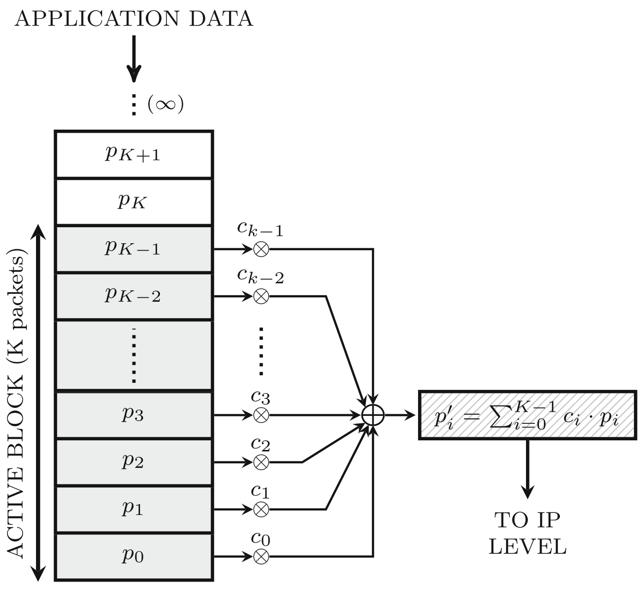

Fig. 1 Coding process (transmitter's RLC layer)

to retrieve the original information. Hence, the transmitter keeps sending coded packets until it receives an acknowledgment message from the destination, confirming that it has successfully decoded the whole block. Then, the source deletes the block from its buffer (at the RLC layer) and starts again with the following K packets. It can be thus said that a coded packet does not mean anything by itself, but each one holds a fraction of information that equals $\frac{1}{K}$ of the whole block.

At the other side, the RLC entity at the destination node needs two different containers: first, a *reception buffer* stores up to K coded packets (i.e. one complete block); second, a matrix $\mathcal{C}$, of dimension $K \times K$, which is used to keep the received coding vectors that come within the protocol header. Upon arrival of an arbitrary coded packet, p'_i, its associated coding vector, $\overrightarrow{c_i}$, is appended at the $(j+1)$th row of the $\mathcal{C}$ matrix, being j its rank at that time (note that j can also be seen as the number of linearly independent vectors/packets received so far). If the received coding vector ($\overrightarrow{c_{j+1}}$) is linearly independent from the previous ones, the corresponding coded packet can be considered as *innovative* and $\overrightarrow{c_{j+1}}$ will be therefore kept at $\mathcal{C}$; besides, the coded packet will be stored at the reception buffer, at the $(j+1)$th position. Otherwise, if the coding vector was linearly dependent, it would be automatically removed from $\mathcal{C}$, and the packet would be silently discarded. As can be easily inferred, the worthiness of every innovative packet is alike $\left(\frac{1}{K}\right)$, but, the packets are meaningless by themselves, since the receiver needs to store K innovative packets to successfully decode the corresponding whole block. This means that if a coded packet was lost, its share of information would be eventually replaced by the next innovative reception. This process is repeated until the matrix is full (i.e. its rank equals K). Then, the destination can obtain the inverse of $\mathcal{C}$ and retrieve the original information. Finally, the destination: *(1)* notifies the correct decoding process of the corresponding block, sending an acknowledgment back to the source node; *(2)* clears both the reception buffer and $\mathcal{C}$.

As an illustrative example, Fig. 2 depicts the events that take place at the source (upper figure) and destination (lower figure) nodes during the transmission of two consecutive blocks. The first one corresponds to an ideal situation where all the packets that arrive at the destination are linearly independent and are received without errors. In this case, the matrix rank (the aforementioned j variable) increases after each reception. Once the destination has finished the decoding process, it sends the corresponding acknowledgment back; the source, after receiving it, starts with the transmission of the next block. In the second example, which illustrates a more realistic situation, a number of negative events might happen: *(1)* reception of *non-innovative packets*; *(2)* loss of coded packets, due to the hostile conditions of wireless links; *(3)* loss of acknowledgments. In this latter case, if the transmitter does not receive the corresponding acknowledgment, it will keep on sending coded packets from the same block (meaningless transmissions); when the destination receives a packet belonging to an already decoded block, it will automatically send a new acknowledgment back to the transmitter.

In order to transport the data that contains the information of the coding process, i.e. the coefficient vector $\overrightarrow{c_i}$, we study two different approaches: on the one hand, the most straightforward alternative, where all the K coefficients are sent "in clear" as part of the RLC protocol header, whose length will thus depend on the block size, K, and the order of the Galois field, $Q = 2^q$. In the second approach, and with the aim to reduce the required overhead, we transmit the random seed that was used in a common PRNG; in this case, the header length is fixed, no matter the order of the Galois Field is. Hence, the RLC protocol implements a proprietary header with the following fields:

- Type of message (1B): This field indicates the packet type: data packet ('0') or acknowledgment ('1').
- Block size K (1B): Number of packets per block. The maximum block size is 256, since the latency for larger blocks (due to the computational time needed to carry out the algebraic operations) might be probably too high.
- Galois Field size q (1B): The linear combination coefficients are randomly obtained from the Galois Field $GF(2^q)$. In order to execute the required operations, we have integrated the M4RIE [3] library into the `ns-3` platform, which establishes a limit of $q = 8$.

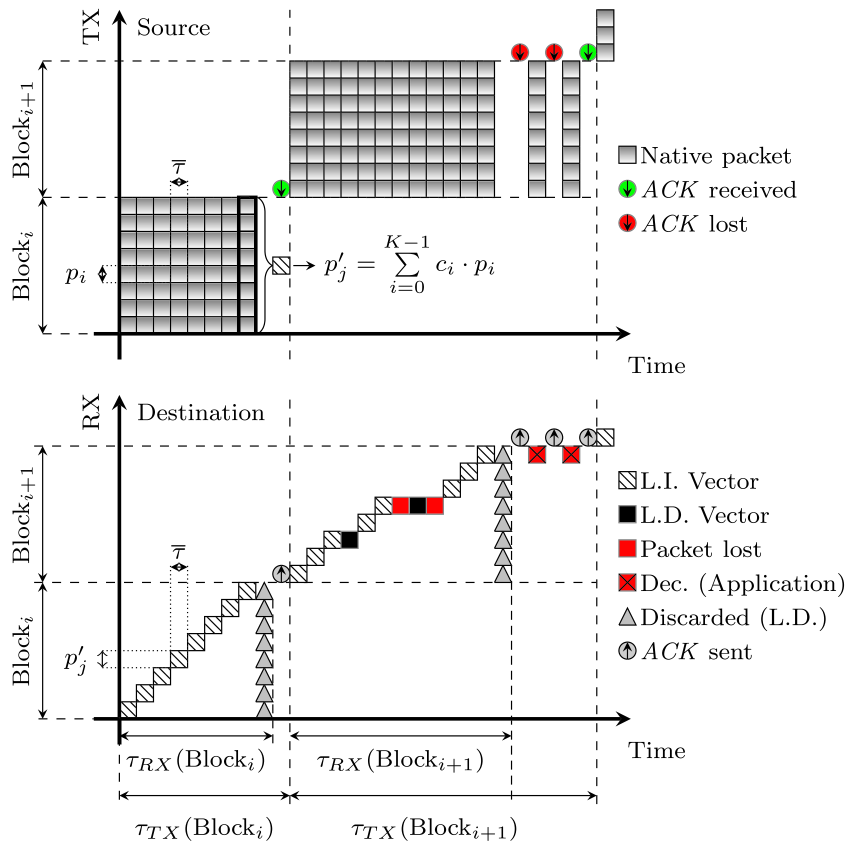

Fig. 2 Illustrative example of the RLC scheme in the source and destination

- Block Number (2B): This field identifies the block that is being sent by the source. It allows identifying spurious transmissions of already decoded blocks.
- UDP source and destination ports (4B): A flow is identified by means of the tuple source-destination IP address/UDP port. Since the UDP header goes within the payload at the RLC level, it will be encoded, and we therefore need to include them "in clear" in this header.
- Coding vector, $\overrightarrow{c_i}$, (1-256B): Contains each of the coefficients c_j used to generate the coded packet p_i'. Each coefficient has a length of q bits and the header must include the K coefficients (in case of using the straightforward option).
- Seed (4B): When using the short header, it would only include the random seed feeds the PRNG.

We can infer that, depending on the values of K and q, there would be a point from which the overhead associated to the initial option would be higher than the one of the short header alternative. The header of the former case is 9 bytes long for the fixed elements (i.e. type, block size, field size, block number and UDP ports) and it also needs an unknown number of bytes for the coefficients vector $\overrightarrow{c_i}$, that can be calculated as $H(K,q) = \left\lceil \frac{K \cdot q}{8} \right\rceil$ bytes. On the other hand, the short header would have always a static length of 9 (fixed elements) plus 4 (random seed) bytes. Taking this into account, Fig. 3 illustrates the area where each of the alternatives requires a lower overhead (if K is greater than 32 the short header approach always yields a lower overhead).

Finally, it is worth highlighting some cross-layer techniques implemented that integrate new features to the presented approach, by connecting the RLC module with the lower layers. First, the transmission rate from the RLC layer downwards is controlled by signals that are transmitted each time a packet is sent to the physical channel. Once the RLC entity receives this signal, it generates a new coded packet, delivering it to the lower layers. Another technique is applied when the source node receives an acknowledgment; besides deleting the K packets of the block at the RLC buffer, all the coded packets that might be waiting to be sent at the lower layer buffers are also cleared, upon a cross-layer signal sent by the RLC module.

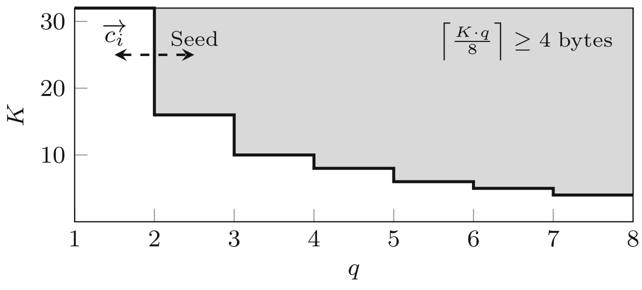

Fig. 3 Region where the use of the new RLC header is worthy

3.2 Analytical model

In order to characterize the performance that might be obtained with our proposed solution, we discuss below an analytical model of the throughput that can be achieved over an IEEE 802.11b link. We start by quantifying the performance loss that can be attributed to the RLC scheme operation. In a nutshell, there are two penalization factors to consider: *(1)* spurious packets received with linearly dependent vectors and *(2)* the overhead caused by the backwards transmissions of the RLC level ACKs, which might have a non-negligible impact on the communication performance.

The spurious transmissions are a direct consequence of the random generation of coefficient vectors, which might lead to the appearance of linear combinations. As described in Section 3.1, the immediate effect would be a packet drop, and the time consumed for its transmission and processing could be thus considered as not useful. We can obtain the corresponding throughput degradation using the model proposed by Trullols-Cruces et al. [28], where they derived the probability of successfully decoding a block when the destination has received N packets ($N \geq K$). This probability is a function of the Galois Field and the block sizes, as can be seen in Eq. 1, where $Q = 2^q$ and $\xi_Q(K,K)$ is the probability of an ideal block transmission, in which the destination node got K *innovative packets* out of K receptions, without any spurious transmissions. The probability for this ideal circumstance is that shown by Eq. 2.

$$\xi_Q(N,K)|_{N \geq K} = \xi_Q(K,K) \cdot \left(\begin{bmatrix} N \\ N-K \end{bmatrix}_Q + \sum_{t=1}^{N-K} (-1)^t \binom{N}{t} \begin{bmatrix} N-t \\ N-K-t \end{bmatrix}_Q \right) \tag{1}$$

$$\xi_Q(K,K) = \frac{Q^{K^2}}{\left(Q^K - 1\right)^K} \cdot \prod_{j=1}^{K} \left(1 - \frac{1}{Q^j}\right) \tag{2}$$

Then, we can obtain the average number of transmissions that are required so as to decode a block of K packets, as shown in Eq. 3, where $p_{dc}(N,K)$ is the probability density function, which can be computed as $p_{dc}(N,K) = \xi_Q(N,K) - \xi_Q(N-1,K)$. The ratio between the excess packets and the overall number of transmissions, ϵ, is shown in Eq. 4, which can be seen as a performance penalization factor.

$$E[N]|_{N \geq K} = \sum_{i=K}^{\infty} i \cdot p_{dc}(i,K) \tag{3}$$

$$\epsilon = \frac{E[N] - K}{E[N]} \tag{4}$$

Table 1 Simulation parameters

Feature	Value
Physical link	IEEE 802.11b (11 Mbps)
Error model	Fixed FER (memoryless)
Frame Error Rate (FER) values	[0:0.1:0.6]
RTX IEEE 802.11	1(RLC), 4(TCP)
Transport layer	UDP (NC)/TCP
Application data rate	Fixed (saturation)
Packet size	1500 Bytes at IP layer

The second aspect to be taken into account is a consequence of the acknowledgments sent by the receiver (recall that we are assuming the use of a semi-duplex shared channel, IEEE 802.11). The corresponding penalization factor, ϵ_{ack}, can be defined as the ratio between the time required to send such confirmation packet, τ_{ack}, and the average transmission time of a data packet, τ_{data}, as can be seen in Eq. 5.

$$\epsilon_{ACK} = \frac{\overline{\tau_{ack}}}{(K+\epsilon)\cdot\overline{\tau_{data}} + \overline{\tau_{ack}}} \tag{5}$$

Considering the previous factors, we can model the expected *goodput*, defined as the throughput perceived by the application layer, $\overline{S_{RLC}}$, as shown in Eq. 6, where $S_{\max}$ is the throughput of a UDP transmission, under saturation conditions, over an error-free link, value that can be obtained with the well-known Bianchi's model [4].

$$\overline{S_{RLC}} = S_{\max}\cdot(1-\epsilon)\cdot(1-\epsilon_{ack}) \tag{6}$$

Under not ideal situations, some packets might be eventually get lost over the wireless link; in this case, we can differentiate two different events; first, we can establish the probability of receiving K' packets after sending N ($K' \leq N$), and then, calculate the probability of successfully decoding a block of K packets when the overall number of received packets is K'. Hence, as can be seen in Eq. 7, we can calculate the new probability of successfully decoding a block after sending N packets, $\xi'_Q(N,K)$. $\mathcal{P}_{rx}(i,N)$ is the probability of receiving i packets after having sent N, and it depends on the particular error model that was used during the experiments.

$$\xi'_Q(N,K) = \sum_{i=K}^{N} \xi_Q(i,K)\cdot\mathcal{P}_{rx}(i,N) \tag{7}$$

4 Results

In this section we discuss the most interesting results resulting from the simulation campaign that was carried out to compare the behavior of the two different versions of the proposed RLC scheme to the one exhibited by a legacy TCP communication (in particular, TCP NewReno) as well as to that shown by more recent TCP variants (in particular TCP Westwood, which has been shown to outperform more legacy solutions, particularly over wireless links). We have structured this analysis along three different scenarios: in the first one, we evaluate the performance over a single IEEE 802.11b link; this will allow us to verify the correctness of the analytical model introduced earlier and will be also used to study the impact that the coding parameters (q and K) might have over the performance of the proposed scheme. In a second scenario, we incorporate an intermediate forwarding node and various flows, focusing on the study of additional statistics, dealing with load balancing and the corresponding fairness. Afterwards, we challenge the proposed solution over a more complex topology, where a greater number of nodes are randomly deployed.

Despite using different scenarios, all of them share several configuration parameters for the `ns-3` simulator, which are detailed in Table 1. As can be seen, we use the parameters of the IEEE 802.11b specification; in addition, we ensure saturation conditions, and the application rate is

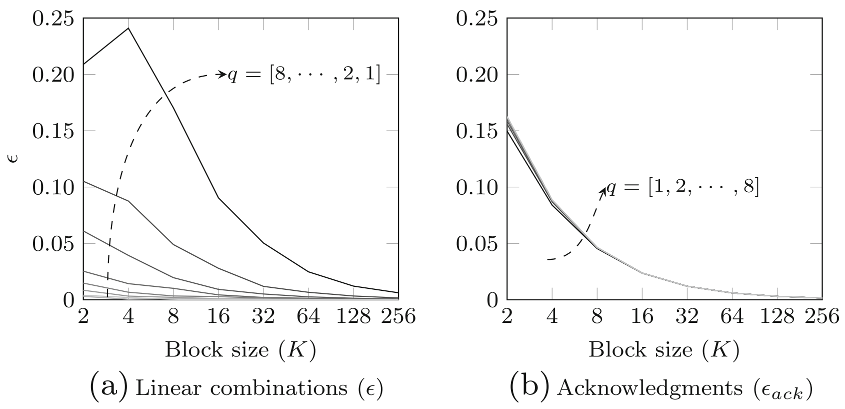

Fig. 4 *RLC* performance penalization factors

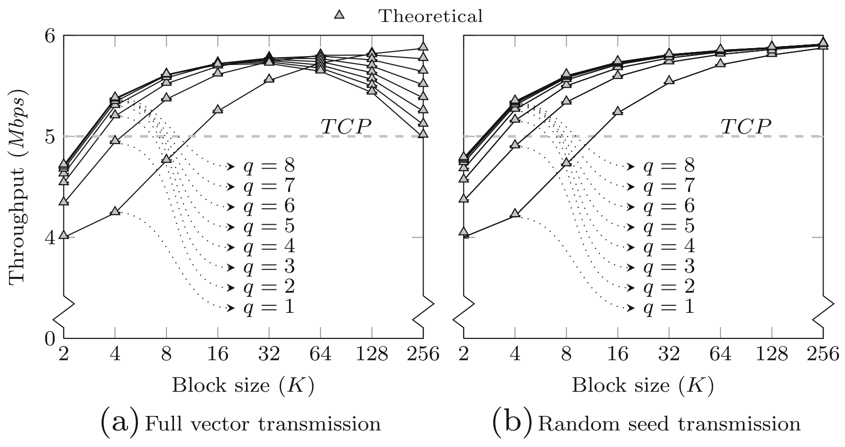

Fig. 5 *Throughput* over an ideal single hop link

thus higher than the maximum capacity of the wireless link, which will actually appear as the system bottleneck.

4.1 Scenario 1. Performance and coding parameters evaluation

As was already derived from Eqs. 4 and 5, the higher the K, the lower the performance penalization; on the other hand, if q gets lower, the probability of having linearly dependent coding vectors increases (leading to more spurious transmissions), bringing a throughput reduction. Figure 4a and b show the evolution of these two penalization factors as a function of both parameters. First, Fig. 4 shows that the use of a higher q greatly increases the observed performance, reducing the number of linear dependencies; in addition, we can also conclude that using higher K (i.e. larger blocks) also reduces the number of spurious transmissions. On the other hand, we can see (Fig. 4b) that the acknowledgment penalization only depends on the block size. From these results, we can anticipate the great relevance of the short header approach, since it would allow using higher Galois Field and block sizes without increasing the overhead.

In order to complement the previous results and to assess the correctness of the model discussed earlier, we integrated the NC protocol within the `ns-3` simulator and we carried out an extensive simulation campaign, in which we compare the throughput measured at the receiver application that was obtained with the proposed solutions to that exhibited by a the traditional TCP (using the New Reno [12] version). All the results are obtained after 50 iterations and we represent the average as well as the 95 % confidence interval.

Figure 5a shows the performance over a single link assuming an ideal situation, when no packets get lost, for different block and field sizes. As was already seen in the obtained model, the larger the block size the higher the performance, since this brings a reduction of both factors' penalization (linear dependencies and number of acknowledgments). However, we can see that there is a point upon which the overhead induced by the transmission of the coding vector jeopardizes the performance and the throughput starts decreasing. This point depends on the size of the Galois Field and the impact is higher for larger values of q, since the overhead imposed by the coding vector would be larger as well.

However, when the short header scheme is used, the overhead does not depend on the block and field sizes, since the header always has the same length (thanks to the the random seed); hence, the throughput is not jeopardized by the increase of the block and field sizes, as can be seen in Fig. 5b. The performance does not decrease, not matter the block size is, and we can therefore exploit the configuration that reduces the impact of the two penalization factors, without increasing the required overhead. In this case the throughput gain is ≈18 %, for $K > 128$. The figure also includes (with triangular markers) the values that were obtained by applying the model that was discussed in Section 3; as can be seen there is an almost perfect match between these results and the ones obtained during the simulation campaign.

In order to assess the impact of faulty links over the performance, we configure the wireless model with a Frame Error Rate (FER) that was increased between $[0.0 \cdots 0.6]$

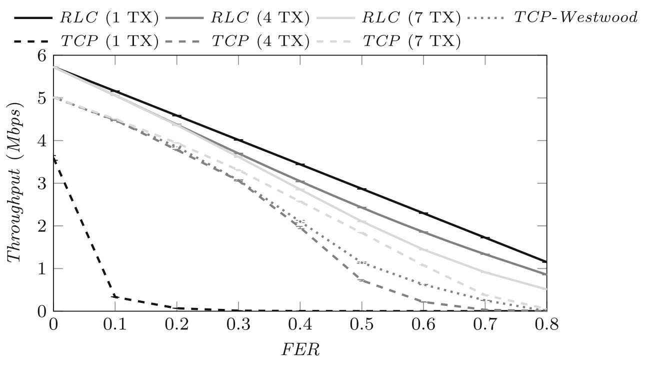

Fig. 6 Throughput Vs link quality for different configurations

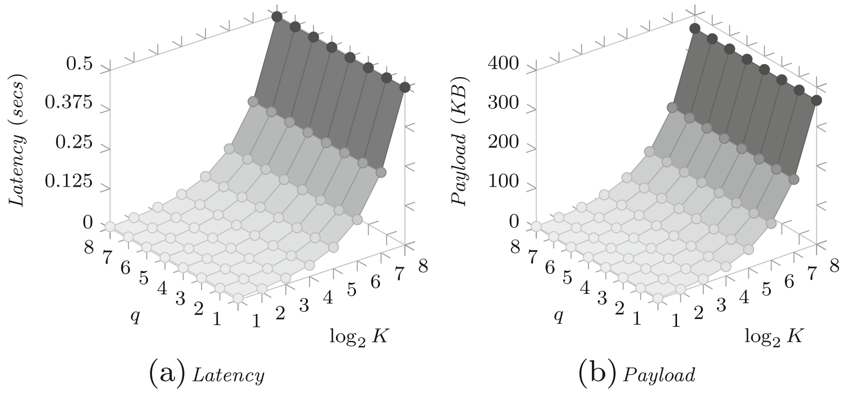

Fig. 7 Latency and Payload as a function of block (K) an field size (q)

(we also assumed that all the acknowledgments sent by the received arrives without errors). Since the performance for the two RLC approaches is rather alike, we fixed $q = 1$ and $K = 64$, and Fig. 6 shows the throughput as a function of the wireless link FER for both the short header approach and the legacy TCP. In both cases, we configured the IEEE 802.11 MAC scheme with a different number of transmissions per datagram; as can be seen, a higher number of transmissions benefit the traditional TCP performance, but this is not the case for the RLC scheme, which gets jeopardized by the MAC retransmission scheme. In any case, we can also see that the proposed solution always outperforms the traditional scheme, especially when the conditions of the wireless link get worse. In this case we have also compared the performance with the one exhibited by a more recent TCP version, TCP Westwood [23]), which was initially conceived to increase the performance over lossy wireless networks. When this TCP version is used, for the sake of clarity, we have fixed the number of IEEE 802.11 transmissions per datagram to four. We can see that the performance of this TCP alternative is indeed higher than the legacy one (NewReno), but the proposed combination of UDP and Network Coding clearly yields higher throughputs.

There might be applications, for instance online gaming or VoIP, with different *Key Performance Indicators*; in particular, they might impose a certain Quality of Service (QoS) level based on time behavior (delay, jitter, etc.). In this sense, we need to bear in mind that in the proposed scheme the application at the receiver will get all the information within a block at once, after the RLC layer has been able to decode it. We can easily see that larger blocks would eventually increase the time between consecutive receptions at the application. Figure 7 shows the average values of both the latency and payload per block as a function of the block size (K) and the order of the Galois Field (q). We can see that the former has a greater impact (there is not a strong dependency with q). This could impose a practical limit on the operational parameters of the RLC scheme; for instance, with $K = 256$, the delay between consecutive arrivals at the destination application was almost 500 ms.

4.2 Scenario 2. Fairness and load balancing

After studying the performance over a single wireless link, we also evaluated the fairness and load balancing that are promoted by the use of the proposed RLC scheme. In order to do so, we deploy two source nodes (S_1 and S_2) that use the same relay node (R_1) to reach their corresponding destinations (D_1 and D_2, respectively), as can be seen in Fig. 8. We initiate two flows in this scenario, $\mathcal{F}_1$ and $\mathcal{F}_2$. We use the Jain's index (Eq. 8) [14] to study how the resources (capacity) are shared between the two flows. In this case N equals 2, while x_i is the capacity allocated to the ith flow.

$$J(x) = \frac{\left[\sum_{i=1}^{N} x_i\right]^2}{N \cdot \sum_{i=1}^{N} x_i^2} \tag{8}$$

Figure 9 shows the evolution of the Jain's index when we decrease the quality of the communication channels for

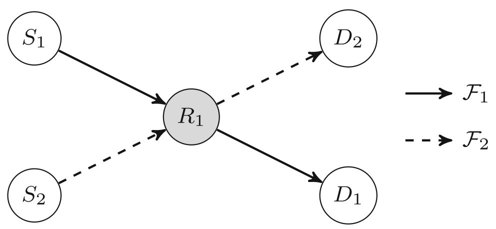

Fig. 8 Cross topology

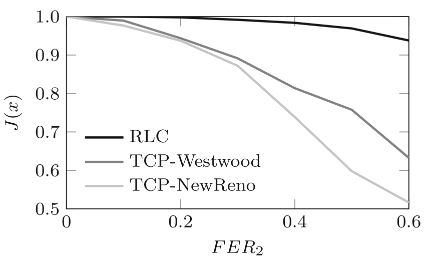

Fig. 9 Fairness Vs Link quality. $FER_1 = 0.0$ and $FER_2 = (0.0 \cdots 0.6)$

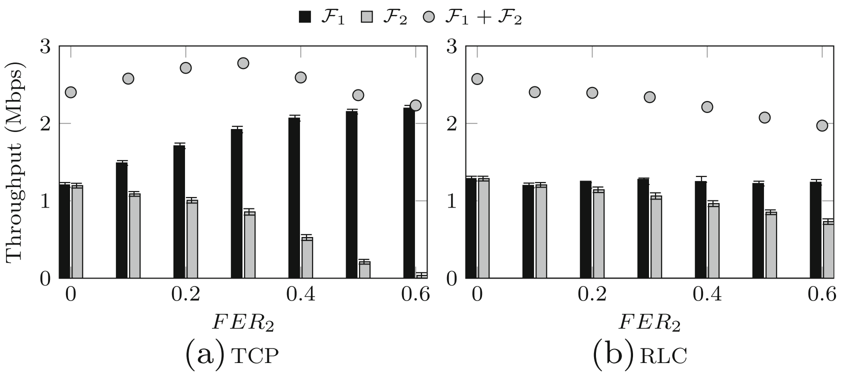

Fig. 10 Throughput distribution between two links

$\mathcal{F}_2$, by increasing the FER over $S_2 \rightarrow R_1$ and $S_2 \rightarrow R_1$ links, i.e FER$_2$; in all cases the other two links are assumed to be free of errors, i.e FER$_1 = 0$. Under these conditions, Fig. 9 represents the Jain's factor as a function of this FER. As can be seen, both TCP versions (New Reno and Westwood) strongly favor the flow traversing better links and the resources are badly shared (for 2 flows the worst Jain's index is 0.5); on the other hand, the RLC scheme ensures a fairer distribution of the capacity, even if the conditions of one of the flows get worse, and the value of J is higher than 0.9 for all FER values.

In order to broaden the previous analysis, Fig. 10 shows the individual throughput per flow, as well as the overall (aggregated) one. We can again see that TCP severely jeopardizes the flow with worse conditions (see Fig. 10a); it reduces the rate at which segments are sent, and this benefits the other flow, which almost doubles its performance when FER equals 0.6. On the other hand, *RLC* maintains the same sharing for all FER values. As a consequence, the performance of the first flow does not increase, but we do not see a remarkable decrease on the second flow throughput either. Although the overall performance is slightly higher for the TCP case, the performance observed for the second flow is rather low, while the RLC scheme is able to maintain it at a reasonable value.

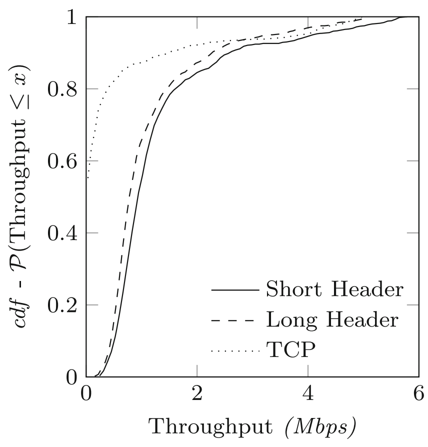

Fig. 11 *cdf* of throughput observed over random wireless networks

4.3 Scenario 3. Performance over random topologies

In this last scenario we exploit the implementation carried out within the `ns-3` framework to study the performance of the proposed schemes over more complex network topologies. We consider wireless mesh networks, by randomly deploying 32 nodes over a squared area of 100 m × 100 m; we assume a disk-radius coverage model of 20 m. Furthermore, before starting the simulation, we ensure the full connectivity of the scenario, discarding those network deployments that do not fulfil this requirement; finally we randomly establish the quality of every link within the interval $[0.0 \cdots 0.6]$. Figure 11 shows the cumulative distribution function (cdf) of the throughput. As can be seen, the short header scheme brings a performance gain of ≈16 %, as compared with the original one, while a gain of ≈×1.7 is observed against the results obtained by the legacy TCP protocol.

5 Conclusions and open research

The use of Network Coding techniques to enhance the performance over wireless networks has gathered the attention of the scientific community during the last years. The research that has been carried out has covered different aspects, ranging from the analysis of the coding/decoding procedure efficiency to the proposal of novel protocols. In this work we have proposed a novel solution, which advocates the joint operation of an RLC scheme and the UDP protocol. Two variations have been depicted, and a thorough assessment has been carried out using the `ns-3` simulator.

After discussing a model that allows the evaluation of how different coding operational parameters (block and field sizes) impact the performance of the proposal scheme, we have assessed its correctness using a simulation based analysis. We have seen that our solution exhibits a remarkable performance enhancement, over both canonical and more complex scenarios, which is ≈20 % if we compare it with the traditional TCP. We have as well looked at the latency and the fairness that are brought by the RLC; we saw that the delay between consecutive arrivals at the destination heavily depends on the block size and this might impose some practical limits to the application of this solution, particularly for time-sensitive services. On the other hand, the results showed that the proposed technique brings a good behavior in terms of fairness, which is not jeopardized when the conditions of the links get worse, as it happens with TCP.

In our future research, we will exploit the RLC scheme that has been introduced in this paper to tackle a number of different research issues. First, we will explore the interplay between the proposed solution and opportunistic routing techniques [5, 18]; these have been shown to yield significant enhancements over wireless mesh networks and we expect that their combination with the RLC scheme would be also beneficial, based on the preliminary results that have been discussed in this paper. Another line would be the integration of *on-the-fly* decoding, as proposed by Sorensen et al. [25]; these would allow decoding some packets before the coding matrix is complete, thus reducing the latency. In addition, we would also like to incorporate some congestion control mechanisms is the proposed protocol, following an approach similar to the one presented in [27].

Acknowledgments This work has been supported by the Spanish Government by its funding through the project **COSAIF**, "*Connectivity as a Service: Access for the Internet of the Future*" (TEC2012-38754-C02-01).

References

1. The ns-3 network simulator. http://www.nsnam.org/
2. Ahlswede R, Cai N, Li SY, Yeung R (2000) Network information flow. IEEE Trans Inf Theory 46(4):1204–1216. doi:10.1109/18.850663
3. Albrecht M (2015) The M4RIE Library—version 20121224. The M4RIE Team. https://bitbucket.org/malb/m4rie
4. Bianchi G (2000) Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J Sel Areas Commun 18(3):535–547. doi:10.1109/49.840210
5. Chachulski S, Jennings M, Katti S, Katabi D (2007) Trading structure for randomness in wireless opportunistic routing. SIGCOMM Comput Commun Rev 37(4):169–180. doi:10.1145/1282427.1282400
6. Feizi S, Lucani DE, Médard M (2012) Tunable sparse network coding. In: Proceedings of the international Zurich seminar on Comm, pp 107–110
7. Gomez D, Hassayoun S, Herren A, Aguero R, Ros D (2012) Impact of network coding on TCP performance in wireless mesh networks. In: 2012 IEEE 23rd international symposium on personal indoor and mobile radio communications (PIMRC), pp 777–782. doi:10.1109/PIMRC.2012.6362888
8. Gomez D, Rodríguez E, Agüero R, Muñoz L (2014) Reliable communications over wireless mesh networks with inter and intra-flow network coding. In: Proceedings of the 2014 workshop on Ns-3, WNS3 '14. ACM, New York, pp 4:1–4:8. doi:10.1145/2630777.2630781
9. Gomez D, Rodriguez E, Aguero R, Munoz L (2014) Reliable communications over lossy wireless channels by means of the combination of UDP and random linear coding. In: 2014 IEEE symposium on computers and communication (ISCC), pp 1–6. doi:10.1109/ISCC.2014.6912516
10. Heide J, Pedersen M, Fitzek F, Medard M (2011) On code parameters and coding vector representation for practical RLNC. In: 2011 IEEE international conference on communications (ICC), pp 1–5. doi:10.1109/icc.2011.5963013
11. Heide J, Pedersen M, Fitzek F, Medard M (2014) A perpetual code for network coding. In: 2014 IEEE 79th vehicular technology conference (VTC Spring), pp 1–6. doi:10.1109/VTC-Spring.2014.7022790
12. Henderson T., Floyd S, GA, Nishida Y (2004) The NewReno modification to TCP's fast recovery algorithm. Internet RFCs. ISSN 2070-1721 RFC 6582. http://www.rfc-editor.org/info/rfc6582
13. Ho T, Koetter R, Medard M, Karger D, Effros M (2003) The benefits of coding over routing in a randomized setting. IEEE international symposium on information theory. Proceedings, p 7803. doi:10.1109/ISIT.2003.1228459
14. Jain R, Chiu DM, Hawe W (1984) A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301. DEC Research Repor
15. Katti S, Rahul H, Hu W, Katabi D, Médard M, Crowcroft J (2008) XORs in the air: practical wireless network coding. IEEE/ACM Trans Netw 16(3):497–510. doi:10.1109/TNET.2008.923722
16. Khurshid A, Kabir M, Das R (2015) Modified TCP newreno for wireless networks. In: 2015 international conference on networking systems and security (NSysS), pp 1–6. doi:10.1109/NSysS.2015.7042948
17. Koetter R, Médard M (2003) An algebraic approach to network coding. IEEE/ACM Trans Netw 11(5):782–795. doi:10.1109/TNET.2003.818197
18. Krigslund J, Hansen J, Hundeboll M, Lucani D, Fitzek F (2013) CORE: COPE with MORE in wireless meshed networks. In: 2013 IEEE 77th vehicular technology conference (VTC Spring), pp 1–6. doi:10.1109/VTCSpring.2013.6692495
19. Lefevre F, Vivier G (2000) Understanding TCP's behavior over wireless links. In: Symposium on communications and vehicular technology, 2000. SCVT-200, pp 123–130. doi:10.1109/SCVT.2000.923350
20. Li SY, Yeung R, Cai N (2003) Linear network coding. IEEE Trans Inf Theory 49(2):371–381. doi:10.1109/TIT.2002.807285
21. Luby M (2002) LT codes. In: The 43rd annual IEEE symposium on foundations of computer science, 2002. Proceedings, pp 271–280. doi:10.1109/SFCS.2002.1181950
22. Lucani DE, Pedersen MV, Heide J, Fitzek FHP (2014) Fulcrum network codes: a code for fluid allocation of complexity. CoRR. arXiv: 1404.6620
23. Mascolo S, Casetti C, Gerla M, Sanadidi MY, Wang R (2001) Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In: Proceedings of the 7th annual international conference on mobile computing and networking, MobiCom '01. ACM, New York, pp 287–297. doi:10.1145/381677.381704
24. Shokrollahi A (2006) Raptor codes. IEEE Trans Inf Theory 52(6):2551–2567. doi:10.1109/TIT.2006.874390
25. Sorensen C, Lucani D, Fitzek F, Medard M (2014) On-the-fly overlapping of sparse generations: a tunable sparse network coding perspective. In: 2014 IEEE 80th vehicular technology conference (VTC Fall), pp 1–5. doi:10.1109/VTCFall.2014.6966091
26. Stewart R, Xie Q, Mornmeault K, Sharp H, Taylor T, Rytina I, Kalla M, Zhang L (2000) Stream control transport protocol. Tech. rep., RFC 2960

27. Sundararajan J, Shah D, Medard M, Mitzenmacher M, Barros J (2009) Network coding meets TCP. In: INFOCOM 2009. IEEE, pp 280–288. doi:10.1109/INFCOM.2009.5061931
28. Trullols-Cruces O, Barcelo-Ordinas J, Fiore M (2011) Exact decoding probability under random linear network coding. IEEE Commun Lett 15(1):67–69. doi:10.1109/LCOMM.2010.110310.101480
29. Zorzi M, Chockalingam A, Rao R (2000) Throughput analysis of TCP on channels with memory. IEEE J Sel Areas Commun 18(7):1289–1300. doi:10.1109/49.857929