Systematic Network Coding with Overlap for IoT Scenarios

Mihail Zverev*, Pablo Garrido*, Ramón Agüero†, Josu Bilbao*
* *Information and Communication Technologies Area*, *Ikerlan Technology Research Center*, Mondragón, Spain
Email: {mzverev, pgarrido, jbilbao}@ikerlan.es
† *Dpto. Ingeniería Comunicaciones*, *University of Cantabria*, Santander, Spain
Email: ramon@tlmat.unican.es

***Abstract*—The presence of IoT in current networking scenarios is more relevant every day. IoT covers a wide range of applications, ranging from wearable devices to vehicular communications. With the consolidation of Industry 4.0, IIoT (*Industrial IoT*) environments are becoming more common. Communications in these scenarios are mostly wireless, and due to the lossy nature of wireless communications, the loss of information becomes an intrinsic problem. However, loss recovery schemes increase the *delay* that characterizes any communication. On the other hand, both reliability (robustness) and low delay are crucial requirements for some applications in IIoT. An interesting strategy to improve both of them is the use of Network Coding techniques, which have shown promising results, in terms of increasing reliability and performance. This work focuses on a possible new coding approach, based on systematic network coding scheme with overlapping generations. We perform a thorough analysis of its behavior. Based on the results, we draw out a number of conclusions for practical implementations in wireless networks, focusing our interest in IIoT environments.**

***Index Terms*—Network Coding, Systematic Coding, IIoT**

I. Introduction

Smart homes, wearable devices, and vehicle to vehicle communications are just some examples among the growing IoT applications [1]. As the manufacturing world is evolving towards the so-called Industry 4.0 [2], the environment of Industrial IoT (IIoT) is becoming more common every day. IIoT deployments typically have multiple devices interconnected through lossy networks [3], while they have a stringent requirement of guaranteeing fast responsiveness to operators' orders. In the case of vehicle to vehicle communications, reliability and low delay are, needless to say, strict requirements.

It is well known that wireless links are prone to induce errors or erasures. The recovery of lost or corrupted information implies repeating its transmission, which would increase network load and so delay. Re-sending information on a shared access medium, such as a wireless link, increases the delay not only for the communication that actually requires such particular re-transmission, but also for all the other communications of the same network, which might need to be on hold.

There are multiple techniques that might prevent re-sending lost information, by means of coding the original messages [4]–[14]. These techniques encode the original message so that the receiver will be able to recover it from only a part of the overall transmission. Up to date this type of code-base solutions has not received enough attention in the IoT realm. The search for IoT optimized coding solutions should thus start with a study of existing approaches and coding schemes. IIoT environments are harsh and unstable, with significant changes in the loss rate [15], [16]. Error modelling is a whole different research niche. However, it is clear that the optimum coding scheme for IIoT environments should increase robustness and reduce delay both for uniform and bursty error links. This work is focused on a particular coding solution, which will allow responding to the needs of IIoT communications. This is only an initial approach to improve IIoT with coding techniques. Communication protocols with coding schemes, and their use in IIoT environments, both real and simulated, will be covered in upcoming works.

The rest of this paper is structured as follows. Section II provides a brief summary of some of the most relevant coding techniques that have been proposed in the academic literature, identifying the aspects that have not been covered yet, which we study herewith. Section III describes the implementation of the proposed coding scheme. Section IV summarizes the obtained results, and Section V concludes this work, identifying future lines of research.

II. Background

Forward Error Correction (FEC) is the technique that has been traditionally used to yield increased communication reliability [4]. The transmitter sends, along with original chunks of information or *symbols*, repair (redundant or coded) symbols. At the receiver, lost symbols are recovered from the redundant ones. Another way to increase communication reliability is using *Network Coding* (NC) [5]. One of the most popular configurations is known as *Intra-flow* NC (discussed in [6]), which shares a number of features with FEC. The transmitter also sends redundant symbols to protect the original information, but intermediate routers have the ability to *recode* such symbols, until they arrive at the receiver. Thus, NC generates coded symbols on a per-link basis, while FEC generates the required ones for the whole end-to-end communication. The use of NC instead of FEC yields a reduction of network overhead, which might be significant in particular scenarios, for instance wireless mesh networks.

One NC technique that has aroused a lot of interest is *Random Linear NC* (RLNC), first introduced in [7] and

later extended in [8]. The transmitter groups *source* (original) symbols in blocks called *generations*, and linearly combines them using random coefficients from Galois Field $GF(2^q)$. The coded generation has the same amount of symbols, plus *redundant* symbols, as many as those that are needed to cover losses in the corresponding link. The most relevant benefit of RLNC is that recoding does not necessarily require decoding the original symbols, since coded symbols can be linearly combined the same way as source symbols, being the original message recoverable as long as each router receives enough symbols for recoding. RLNC coding technique can also be applied for FEC (coding at endpoints, no recoding), in which case it is referred to as *Random Linear Coding* (RLC).

RLNC introduces computational complexity, which is not desirable at devices with computational power, processing, or/and memory constraints, such as IoT 'things'. One way to reduce this complexity is adding sparsity to the codification process, leading to *Sparse NC* (SNC) [9]. Any coded symbol is built only with some of its corresponding generation's source symbols, rather than with all of them. As a result, coded symbols are not protecting all of its generation's symbols. One example of these sparse techniques is *Systematic NC* (SysNC) [10], in which RLNC coding is applied only to the redundant symbols, while the source ones are transmitted without coding. Since the original information is sent uncoded, most of it can be received and used immediately, without decoding.

Classic RLNC approach can be described as *block based*, since the original information is coded in non-overlapping generations. However, *overlapping* generations (technique known as *convolutional coding* [4]) offers an important advantage, as can be easily observed with SysNC: overlapping shortens the period between repair symbols, which allows start repairing lost source symbols earlier, therefore reducing recovered symbols' delay. By overlapping generations, an encoding sliding window is defined, which covers the source symbols belonging to the newest generation. An example of SysNC with finite sliding encoding and decoding windows is the Caterpillar RLNC (CRLNC) protocol, presented in [11] and extended with ARQ (Automatic Repeat reQuest) functionality in [12]. CRLNC offers decoding probability very close to the block based systematic RLNC, but with much lower delay.

Another coding approach similar to overlapping is *interleaving* [13]. A block of generations (interleaving block) is sent, symbol by symbol, using time multiplexing. The transmitter initially sends the first symbol of each generation, followed by the second symbols, and so on until all symbols are transmitted, including the coded ones. The advantage of interleaving is its robustness to symbol losses happening in bursts. This approach introduces a delay, which can be avoided as shown in [14], using a technique named *Interleaving with On-the-fly Coding* (IOC). In this solution source symbols are assigned to a generation within the interleaving block in the Round Robin style, which allows the interleaving coding scheme sending source symbols in their natural order. IOC may require big amounts of memory, both for coding and decoding operations.

The robustness for burst of losses offered by IOC might be very beneficial for IoT environments. An interesting concept to explore is the combination of IOC with overlap, which may result in a highly efficient coding scheme. However, both of them may require excessive memory for coding and decoding processes. The viability of its usage in IoT scenarios is yet to be evaluated.

In view of the above, SysNC with overlapping seems to be a suitable coding scheme for IoT. In [11], the CRLNC protocol inserts only one coded packet per generation. The authors of the original study present results showing that the loss probability is smaller for bigger generation sizes, and that the end-to-end delay was lower just for smaller generations. However, it is not clear whether introducing multiple coded symbols per generation might have an impact on this scheme's efficiency. In order to clarify that, we have developed our own implementation of a SysNC solution, which we use to carry out a thorough simulation campaign.

III. Systematic NC Implementation

Depending on the particular NC algorithm implementation, it might be possible to send multiple symbols within the same network packet. However, a loss of a single packet implies a burst of losses. At first look, sending multiple symbols in one packet does not seem convenient. Hence, in this study we consider transmissions of one symbol per packet. From this moment on the meaning of *packet* is equivalent to *symbol*.

A. Coding Scheme

We have carried out a SysNC implementation in Python. Redundant symbols are built with RLNC scheme, combining source symbols multiplied by random coefficients from Galois Field $GF(2^8)$.

A *generation* is defined as the group of symbols coded within the same encoding window, which is considered of constant length, as in [11]. Each coded symbol is built from linear combination of source symbols from a specific generation. In other words, each coded symbol protects source symbols from the corresponding generation. Each generation is protected by r redundant symbols.

Overlapping generations implies that at least one source symbol in any generation will belong to 2 or more generations. We define *overlap* as the number of generations φ to which a source symbol belongs. In this work we consider a constant φ for every source symbol. Thus, each source symbol is going to be protected by $r \cdot \varphi$ coded symbols. Given this definition, classic SysNC can be seen as a particular case of overlapping NC with $\varphi = 1$. This definition implies the creation of additional partial generations to protect the last source symbols. This situation is depicted in Fig. 1, where yellow boxes are source symbols and green boxes are the coded ones. In this example of only 15 source symbols and $r = \varphi = 2$, it can be clearly seen that without the last $r = 2$ coded symbols, the last block would not comply with the definition of overlap, belonging to just $1 < \varphi = 2$ generation.

Another consequence of overlap is introducing coded symbols between source symbols belonging to the same generation. In Fig. 1 it is easy to identify groups of consecutive

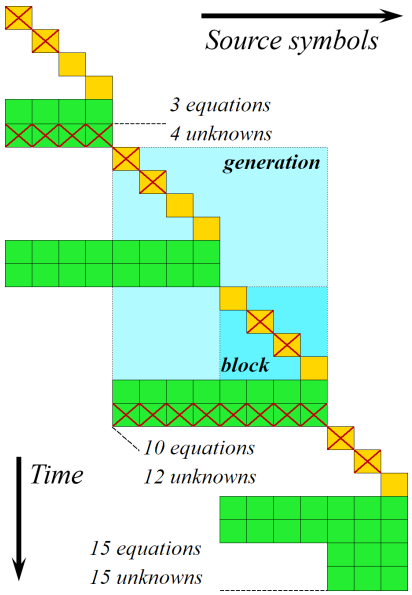


Fig. 1. Example of the implemented coding scheme. Yellow boxes are source symbols, green boxes are coded symbols, red crosses mark the lost symbols.

source symbols between the coded ones. These symbols belong to exactly the same generations. Thus, we define a *block* the consecutive source symbols belonging to the same generation between coded symbols. Generation and block sizes will be referred to as g and k, respectively.

The example presented in Fig. 1, illustrates a communication with erasures. Lost symbols are marked with a red cross. In this case none of the lost source symbols can be recovered until receiving the last coded symbol. This example can be expanded to a much greater number of source symbols with the same problem: lost source symbols still can be recovered, but at the end of communication. Depending on the particular protocol using this solution and its congestion control scheme, most of these symbols will surely be resent by the transmitter before recovery becomes possible. However, keeping all the received symbols in memory would eventually allow recovering the lost ones. In other words, *recovery probability depends on the amount of memory available at the receiver*. We conclude that the decoding window should be as long as possible. Since the goal of the current study is to explore the opportunities offered by SysNC coding scheme, a sufficiently large buffer is considered to store all source and coded symbols.

The recovery process is triggered after receiving a coded symbol. Given that the channel (described in Section III-B) delivers all symbols in order, this recovery policy is equivalent to trying to recover after each received symbol.

B. Channel

We consider that transmitter and receiver are connected through a wireless channel. The channel is error prone and without packet reordering.

In previous works, the use of uniform distribution for symbol loss modelling is quite common, as in [10]. However, burst errors might characterize real scenarios. This type or erasure distributions can be mimicked with the well-known Gillbert-Elliot model, as in [11] and [14]. In this study both options are considered. Next we detail our Gillbert-Elliot model implementation.

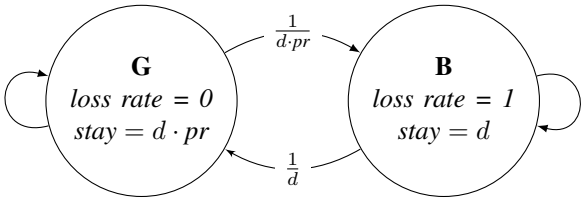


Fig. 2. Gilbert-Elliot model implementation.

In the Burst state (B) the loss rate is 100%, being 0% in the Gap state (G). Burst state duration is set to d, with the Gap lasting $pr \cdot d$. An overview of this implementation is presented in the Fig. 2. In this work pr is adjusted to match an overall loss rate, so we can compare the results to those obtained with a uniform channel model having the same loss rate.

C. Inputs and Outputs

Some of the inputs for the communication model used in this work have been already mentioned: generation size, redundant (coded) symbols per generation, overlap, loss rate, burst length, and the number of packets to store at the receiver. At this point one last input needs to be specified: the number of symbols that will be transmitted per experiment. One approach is to keep sending symbols until the decoder can receive (and recover) n source symbols. We follow an alternative approach: n source symbols are sent to the receiver along with the corresponding coded symbols.

In order to see if the coding scheme is working, we need to ascertain whether the receiver has seen all of the source symbols that were transmitted, and if not, how many of them have been correctly received. Thus, the main output is the number of source symbols that have correctly arrived at the receiver, either via regular reception or recovery. Another interesting metric is the ratio between the recovered symbols and the lost source symbols. In this work we evaluate the probability of receiving (normal reception + recovery) all source symbols (henceforth *reception probability*). This probability is calculated by dividing the number of events in which all source symbols have been either received or recovered, by the number of simulations executed for a specific set of inputs, as shown in equation 1.

$$P(reception) = \frac{\#\ successfull\ experiments}{\#\ simulations} \quad (1)$$

Delay is going to be estimated in terms of time slots required for a packet to reach the receiver. For a correct and reliable data transmission, in-order delivery is highly important. For this reason we focus on the end-to-end delay. In addition, we ignore delay of packets that could not be recovered, and so we

only consider for delay estimation those cases where reception probability equals 1.

As stated in Section I, our aim is to evaluate the impact of changing redundant symbols per generation with the overlapping SysNC coding scheme. Thus, the two main variables in our input parameters are: redundancies per generation and overlap. In order to facilitate the discussion of results, both variables are combined into one, the *overhead*, which we define as the ratio between the total number of coded symbols and the total number of symbols generated by the transmitter. As can be observed in Fig. 1, by the end of the transmission $(\varphi - 1) \cdot r$ additional coded symbols are sent. The overall number of coded packets C that protect S source symbols can be calculated as follows:

$$C = \left(\left\lceil \frac{S}{k} \right\rceil + \varphi - 1 \right) \cdot r = \left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r \quad (2)$$

From C, the overhead $\hat{O}$ can be obtained as follows:

$$\hat{O} = \frac{C}{C+S} = \frac{\left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r}{\left(\left\lceil \frac{S}{\lfloor g/\varphi \rfloor} \right\rceil + \varphi - 1 \right) \cdot r + S} \quad (3)$$

IV. Results

In order to better understand the impact of both the redundancy per generation and the overlap over the coding scheme, a simulation campaign has been carried out with the following input parameters: generations of $g = 64, 256$ symbols; overlaps of $\varphi = 1, 2, 4, 8, 16$ generations; 2000 source packets (symbols) are transmitted; loss rates of 1%, 5% and 10% with a uniform erasure distribution, and an overall loss rate of 10% with burst losses, having a mean burst duration of 5 time slots (mean gap duration: 45 time slots). In order to ease coding scheme analysis, the memory at the receiver is considered big enough to store all source and coded packets sent by the transmitter. Furthermore, to ensure statistical tightness of the results, 10000 independent experiments have been run for every configuration.

The results are shown in Fig. 3 and Fig. 4. The top rows show the probability of receiving or recovering all source packets for different loss models. The bottom row depicts the mean end-to-end delay. As mentioned in Section III-C, delays are only studied when it is worthy (i.e. all packets were received). We include in all cases the 95% confidence interval.

In Fig. **??** it can be observed that the reception probability with an overlap of 8 generations is not always higher than the one observed for overlaps of 2 and 4 generations. The same can be seen in Figure **??**, where the probability with $\varphi = 16$ at some points is lower than with $\varphi = 4$ and $\varphi = 8$. This is due to the distance between the available samples. The points in which lower overlap schemes have higher reception probability are simply not available for schemes with greater overlap. For a fair comparison we would need to focus only on those overhead values for which the samples of all curves under comparison are defined. At those points where the communication overhead generated by coded symbols is comparable, schemes with higher degrees of overlap are more likely to receive all the transmitted symbols.

In Figures 3 and 4 we can see that by having a greater overlap, the reception probability increases and so delay decreases, being the latter the main advantage of SysNC with overlap. According to these results, latency decrease is proportional to φ, as could have been expected, since the blocks (as defined in Section III-A) are precisely φ times smaller than the blocks of SysNC with no overlap. The conclusion of these observations is that within a given overhead, it is more efficient to increase overlap than to have a greater number of redundant symbols per generation.

It is important to keep in mind that *the bigger the overlap, the faster the overhead growth rate becomes, as redundancies per generation increase*. With respect to the possible implementations of this scheme in the IoT field, it should be noted that it might be impossible, due to device limitations or implementation requirements, to increase the overlap. It is thus important to bear in mind that the greater the degree of overlap to be used, the larger the overhead, since there are momre redundancies per generation.

An interesting aspect is the overhead at which reception probability reaches its maximum value. We refer to it as *saturation overhead*. This parameter is of special interest when it comes to implement NC in a network with restricted bandwidth. Given that the delay was only studied for those overheads at which reception probability reaches 1, saturation overhead corresponds to the first sample of the delay curve, shown in Figures 3 and 4. Figure 5 shows saturation overhead for all configurations. As can be observed, larger overlap does not necessarily lead to lower saturation overhead. This parameter depends both on the ability of the encoding scheme to recover lost packets, and on the overhead it introduces, so no clear conclusion can be drawn regarding this parameter.

When it comes to choosing overlap and redundancies per generation, it is important to identify the lowest possible saturation overhead. In other words, to find the configuration that will ensure the recovery of most of lost packets with the lowest possible overhead. As the overlap increases, the saturation overhead may increase rather than decrease. By optimizing the saturation overhead, the latency may not be optimal. It is therefore essential to prioritize between channel efficiency (throughput) and delay.

V. Conclusions

In this paper we have studied network coding techniques, with a special interest in identifying those that are suitable for IoT communications. We have focused on the scheme described in [11], broadening its initial operation, including the possibility of modifying the number of redundancies per generation. This implementation has been used to carry out an extensive simulation campaign.

The results show that it is more efficient to increase overlap than to have a greater number of redundant symbols per

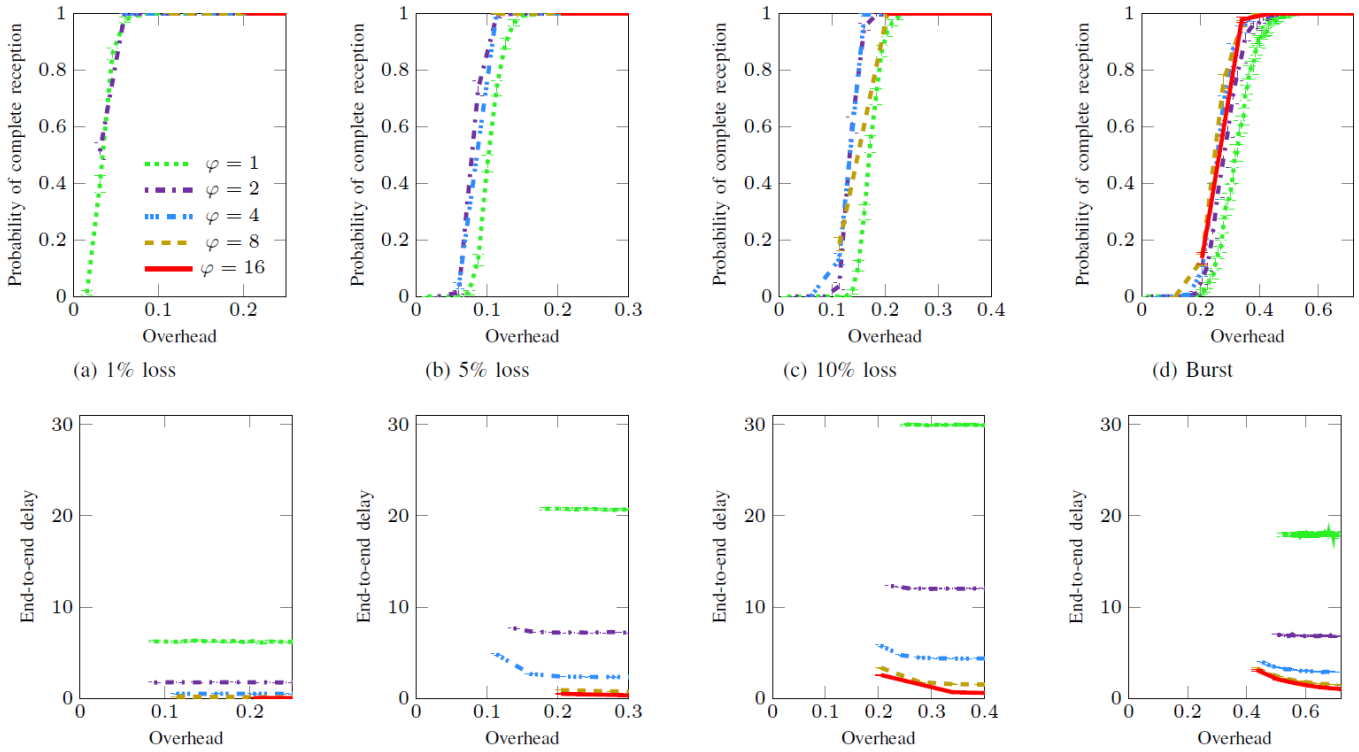


Fig. 3. Probability of receiving and recovering transmitted packets (a–d) and end-to-end delays (e–h) for generation size of 64 symbols. The plots are depicted for uniform losses of 1% (a, e), 5% (b, f) and 10% (c, g), and burst losses (d, h).

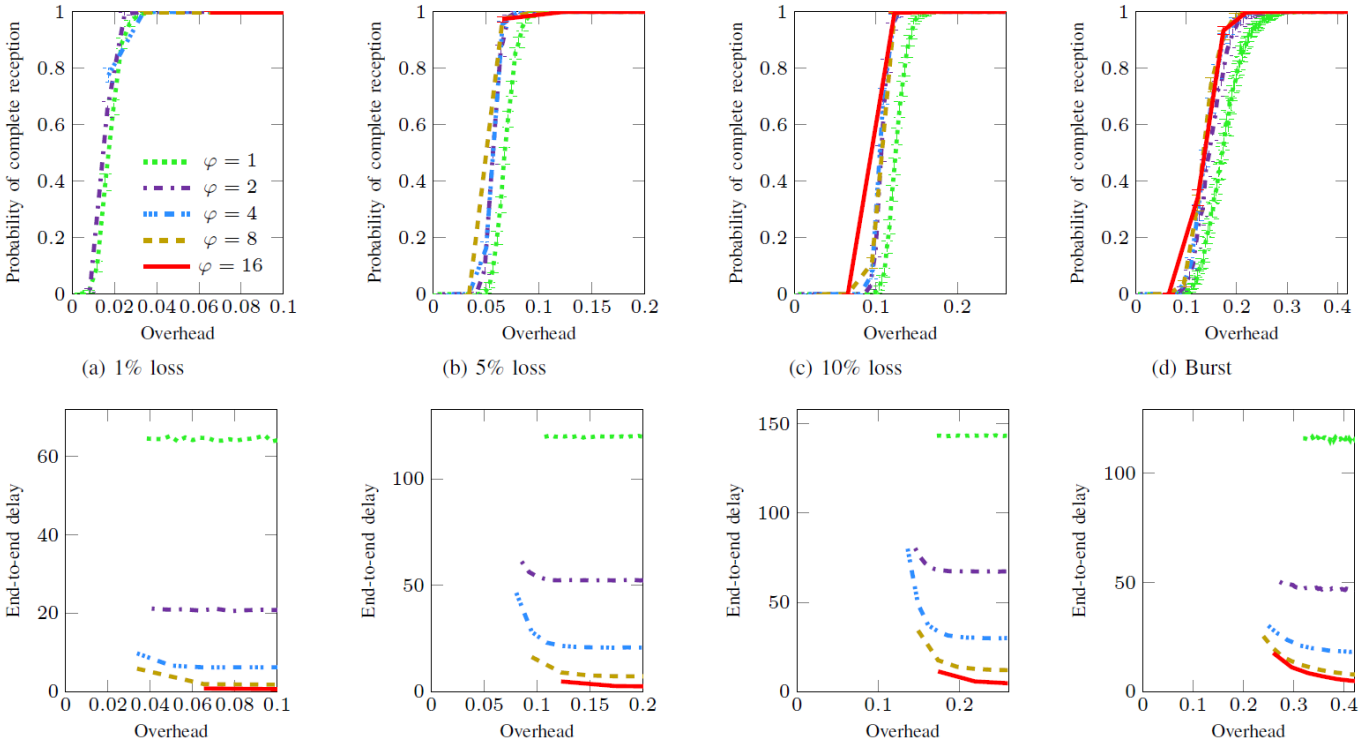
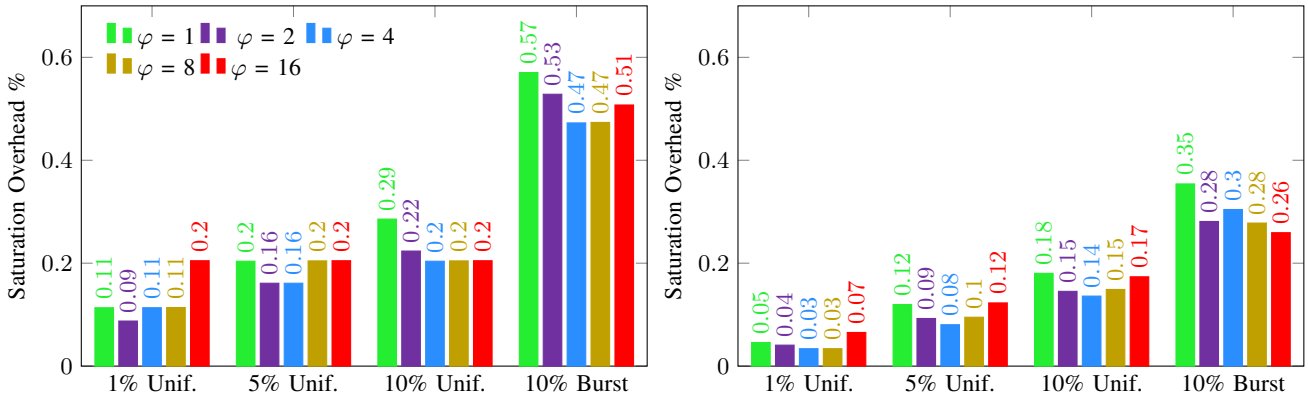


Fig. 4. Probability of receiving and recovering transmitted packets (a–d) and end-to-end delays (e–h) for generation size of 256 symbols. The plots are depicted for uniform losses of 1% (a, e), 5% (b, f) and 10% (c, g), and burst losses (d, h).



(a) g = 64

(b) g = 256

Fig. 5. Saturation overhead values for different cases covered by the present study.

generation. Another important observation is that the greater the degree of overlap to be used, the larger the overhead, by increasing the redundancies per generation.

In this work we have also identified saturation overhead, which we define as the overhead at which reception probability reaches its maximum. Minimizing saturation overhead optimizes throughput, but not the delay. It is therefore essential to find the trade-off between channel efficiency (throughput) and delay.

The next step is to implement the proposed scheme in real devices, and to analyze the impact of devices' limitations on the the coding configuration. The effect of considering networks with more than one wireless hop and the influence of not having unlimited capacity (bandwidth) will be also considered.

Acknowledgment

The authors are grateful for the funding of the Industrial Doctorates Program from the University of Cantabria (Call 2018). This work has been partially supported by the Basque Government through the Elkartek program under the DIGITAL project (Grant agreement no. KK-2019/00095), as well as by the Spanish Government (MINECO, MCIU, AEI, FEDER) by means of the projects ADVICE: Dynamic provisioning of connectivity in high density 5G wireless scenarios (TEC2015-71329-C2-1-R) and FIERCE: Future Internet Enabled Resilient Cities (RTI2018-093475-A-100).

References

[1] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78.

[2] F. Shrouf, J. Ordieres, and G. Miragliotta. Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 697–701.

[3] C. Shi, Z. Ren, K. Yang, C. Chen, H. Zhang, Y. Xiao, and X. Hou. Ultra-low latency cloud-fog computing for industrial internet of things. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6.

[4] Hang Liu, Hairuo Ma, Magda El Zarki, and Sanjay Gupta. Error control schemes for networks: An overview. *Mobile Networks and Applications*, 2(2):167–182, 1997.

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

[6] Dina Katabi, Sachin Katti, and Hariharan Rahul. *Chapter 2 - Harnessing Network Coding in Wireless Systems*, pages 39–60. Academic Press, Boston, 2012.

[7] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *IEEE International Symposium on Information Theory, 2003. Proceedings.*, page 442, 2003.

[8] 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*, 52(10):4413–4430, 2006.

[9] M. Wang and B. Li. How Practical is Network Coding? In *2006 14th IEEE International Workshop on Quality of Service*, pages 274–278, 2006.

[10] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen. Network Coding for Mobile Devices - Systematic Binary Random Rateless Codes. In *2009 IEEE International Conference on Communications Workshops*, pages 1–6, 2009.

[11] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein. Caterpillar rlnc (crlnc): A Practical Finite Sliding Window RLNC approach. *IEEE Access*, 5:20183–20197, 2017.

[12] F. Gabriel, S. Wunderlich, S. Pandi, F. H. P. Fitzek, and M. Reisslein. Caterpillar RLNC With Feedback (crlnc-fb): Reducing Delay in Selective Repeat ARQ Through Coding. *IEEE Access*, 6:44787–44802, 2018.

[13] Z. Huang, X. Wang, X. Chen, and H. Kan. Network coding with interleaving. In *2007 International Conference on Parallel Processing Workshops (ICPPW 2007)*, pages 43–43.

[14] D. Stolpmann, C. Petersen, V. Eichhorn, and A. Timm-Giel. Extending On-the-fly Network Coding by Interleaving for Avionic satellite links. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.

[15] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz. Measurements of a wireless link in an industrial environment using an ieee 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics*, 49(6):1265–1282, 2002.

[16] V. C. Gungor and G. P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265, 2009.