Rui A. Costa

Network Coding for Delay Constrained Wireless Systems with Feedback

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Outubro de 2012

Rui A. Costa

Network Coding for Delay Constrained Wireless Systems with Feedback

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Doutor em Ciência de Computadores

Orientador: Professor Doutor João Barros, Universidade do Porto

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto

Outubro de 2012

To my dearest love, Carla, and our princess, Margarida.

Acknowledgements

Tradition says that the advisor is the first in line for the acknowledgements. In my case, it makes sense to start from the root of my path in the ICT research world, Professor João Barros. Long road we traveled together, João. Almost eight years passed from our first meeting to my Ph.D. defense. And so many adventures! Some ups and downs along the way, but your enthusiasm and your constant pressure for reaching out the world were an invaluable key to keep pushing forward. At some points, guiding me through the PhD was as challenging as finding orientation in a Möbius strip. At those points, you were always able to bring me back to this sphere and point me to the right direction. Thank you, João!

Professors Lars Rasmussen, Daniel Lucani, Susana Sargento, Pedro Brandão and António Porto, thank you for the remarkably valuable questions that aggrandized my Ph.D. defense. It was an honor to have you as the jury of my Ph.D. defense.

Professor Michael Langberg, thank you for the fantastic times we shared in our quest to understand fundamental questions in Information Theory. Thank you for reminding me the pleasure in asking simple yet hard questions.

Professor Daniel Lucani, thank you for your guidance and support through the last mile in this path. Daniel, your passion for research and your integrity are inspiring. It was a privilege to share with you the joy of solving research problems.

At IT Porto, I have found a challenging environment and a fantastic team. Thank you all for so many amazing moments we spent together and for the support during my Ph.D.

I have been blessed with extraordinary partners along this walk. Those of the kind one can truly call friends. Fabio Hedayioglu, João Vilela, Matthieu Bloch, Mate Boban, Sérgio Crisóstomo, William Carson, fantastic team mates! Your support was invaluable, the fun we had together priceless! Luísa Lima, your passion for excellence is remarkable. At many moments, you reminded me of core values that the Ph.D. student life sometimes puts on the side. How about you, Lato? What an extraordinary friend you are. Your serenity, your objectiveness and specially your passion for life are a role model for me. And Tiago Vinhoza and its passion for problem solving? Inspiring, my friend! Diogo Ferreira, I hope you always keep that genuine happiness. In many dark days, your humor was the light that kept the machine moving. Your prompt help will never be forgotten. João Almeida, you insist in reminding me that we are a consequence of our ability to dream. It is a privilege to bread from your noble view of the world. And you, Paulo Oliveira? Your bulletproof faith in me makes me believe we can even move mountains. In times of trouble, you were there. Your comradeship is remarkable, my dear friend.

My closest family is the key in shaping what I am. Mariana, you are a dreamer too. You keep bringing me back to my essence. Sr. Carlos, you have taught me the value of persistence and of honor. Never stop dreaming, never stop pushing forward, you do not say. You do. I hope I can follow your path.

People say one key quality I have is the ability to smile even when the world is ending. To dream, even in the middle of the storm. That is not me. It is my father. People also insist that I am a fighter, that I keep pushing the wagon even through the storm. Again, that is not me, it is my mother. Pai, Mãe, you have fought so much just to let me and Mariana dream. All I have reached it is yours. This thesis is yours, not mine.

Margarida, I hope I am capable of providing you with the foundations my parents provided me. I hope I can let you dream as my parents let me dream. The world will then be yours to conquer, Princesa. You are a co-author of this thesis, Margarida, you are. You even wrote some lines!

My love, you were right: I made it! You never stopped believing in me. Your support kept me moving forward even through foggy days. Your love gave me a reason to stand up after every fall. You give a special meaning to my life. We fought all the battles together. And we will celebrate all the victories together!

This work was supported by the Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology), under grant SFRH/BD/27273/2006, and from Instituto de Telecomunicações, under grant BIM/n.19/2012.

Abstract

The advent of network coding brought a disruptive view to protocol design for communication networks. By combining different source packets in a single coded packet, network coding brings higher robustness and throughput for wireless networks. However, delay is not a primary concern in the design of standard network coding solutions. The topic of this thesis is how to use feedback information to design network coding solutions that can cope with traffic with delay constraints.

We start by analyzing the delay that standard network coding solutions induce, when the time necessary to decode individual packets is of crucial importance. We then propose network coding schemes that allow receivers to immediately recover a new source packet from each coded packet. From a different angle, we propose delay control mechanisms for throughput optimal network codes. The proposed schemes offer a fine trade-off between the delay experienced by the receivers and the amount of throughput sacrificed to achieve such delay performance.

We then focus our attention in scenarios where packets have to be delivered before strict deadlines. We propose a network coding protocol that makes use of periodic feedback transmissions to adapt its coding decisions in order to minimize the number of packets that miss the corresponding deadline. We further show that our proposal is suited for current wireless standards, by implementing the proposed scheme in a IEEE 802.11 wireless testbed.

Finally, we provide a characterization of optimal network codes for maximizing the mean number of delivered packets over a half-duplex channel, when the receiver requires that a subset of the source packets is delivered with high probability. By jointly optimizing the scheduling of the feedback transmissions and the code design, we show that a judicious use of the feedback channel has a significant impact both in the mean number of delivered packets to the receiver and in the reliability level achieved.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The groundbreaking work in [ACLY00] showed that, in general, the multicast capacity of a communication network cannot be achieved by the classical networking paradigm, where intermediate nodes in a communication flow simply store and forward the received information. [ACLY00] proved that we must allow intermediate nodes to combine different information packets in a single transmission in order to achieve the capacity of a network. The classical example used to exhibit this necessity is the so called *butterfly network*, depicted in Figure 1.1, where one source transmits two bits to two receivers over a network composed by links of unitary capacity. In the standard store-and-forward paradigm, the link between nodes 4 and 5 is a bottleneck, since only one bit can be forwarded by node 4, which leads to one of the receivers (either node 6 or node 7) obtaining incomplete information. The multicast capacity of this network is 2 bit per transmission and it can only be achieved if we allow node 4 to combine the two bits via a simple XOR operation, $a \oplus b$. The receivers can then combine the received data with this coded transmission in order to recover the missing information. More precisely, since node 6 receives bit a from node 2 and $a \oplus b$ from node 5, it can recover b from this coded transmission using a. Similarly, since node 7 receives b from node 3, it can recover a from $a \oplus b$.

The fact that, in the previous example, intermediate nodes combine information packets via linear operations is not accidental. In [LYC03], the authors prove that linear operations are sufficient to achieve the multicast capacity of a communication network. In these linear network coding approaches, each node transmits a coded packet that is a linear combination of the packets it has received and the packets itself generates, where the operations are considered in a finite field. The challenge for designing network coding schemes is therefore to decide which coding coefficients

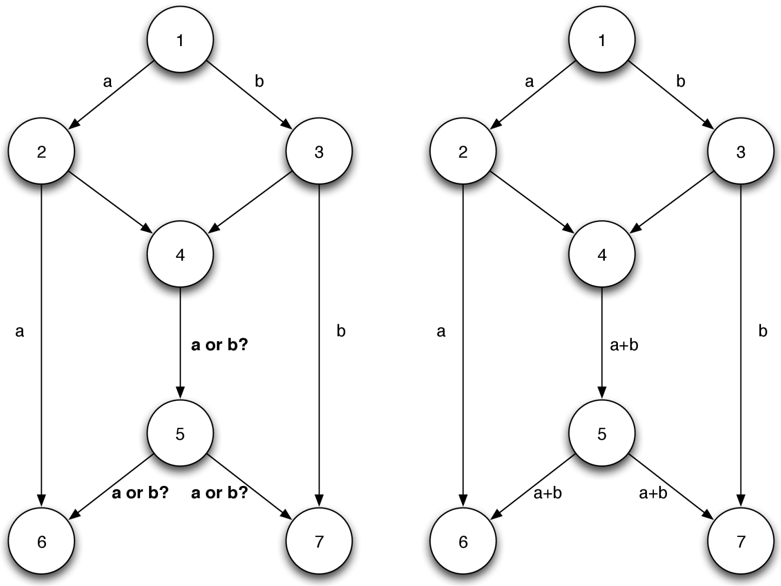


Figure 1.1: The butterfly network example. The multicast capacity of the network, which is 2, can only be achieved if node 4 combines the two incoming flows in a single transmission via coding operations.

to use in order to achieve capacity. The analysis in [HMK+06] shows that, if we consider a sufficiently large finite field, choosing the coefficients purely at random generates a network code that achieves the multicast capacity of the network, with high probability. The probability distribution for the completion time of random linear network coding is investigated in [NLC+11], showing that coding over a binary finite field exhibits a heavy tail in the distribution, while for increasing finite field sizes this tail is significantly smaller.

Network coding has also exhibited significant benefits in terms of throughput and robustness when we consider the naturally unreliable wireless networks [FLBW06, KRH+08]. The broadcast property of the wireless channel, by which a single transmission potentially reaches several receivers, opens the door for the improvements that coded transmissions can provide. A typical example of wireless network coding is depicted in Figure 1.2, where two nodes wish to communicate via an intermediate node. After receiving the two bits from the receivers, the intermediate node has to decide what to transmit next. If it transmits bit a, node B will obtain the desired packet. However, since we are considering a broadcast medium, node A also receives this transmission and, since it already has a, it finds this transmission useless. A similar phenomenon occurs when the intermediate node transmits bit b, which leads

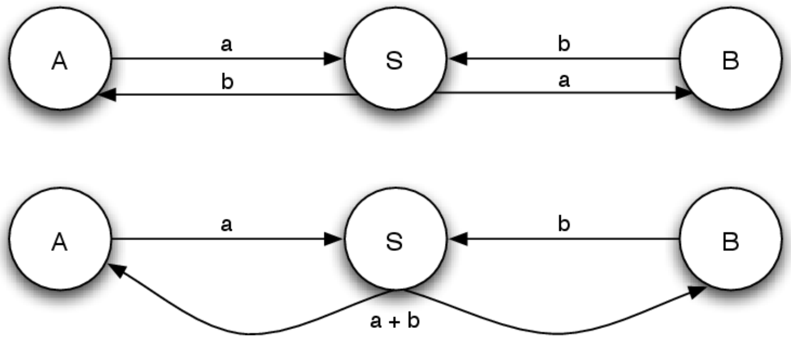


Figure 1.2: Classical example of wireless network coding. Allowing the intermediate node to combine the two flows allows saving 25% of the transmissions, when compared to the case without coding.

to a total of 4 transmissions necessary to delivered the intended bits to all the receivers. If we allow the intermediate node to combine the two bits by transmitting $a \oplus b$, the two receivers will find such transmission useful: node A has a and thus recovers b, and node B has b and thus recovers a. Therefore, with only 3 transmissions, both receivers obtained the desired information.

In the classical network coding literature, delay is not a primary concern. Typical network coding solutions are throughput optimal in the sense they provide the receivers a block of information packets in the minimum number of transmissions. However, receivers are required to wait for several packets until they are able to decode part of the information, which can be prohibitive for delay constrained applications. For instance, real-time applications require that part of the information is delivered to the application even before the entire block of data is received. Thus, the design of network coding schemes for these applications has to take into account the decoding delay that it induces at the receivers, even if at the cost of losses in terms of the throughput achieved. A first approach to reduce the decoding delay in network codes is to limit the amount of different information packets that are combined in a single coded packet [KMFR06]. Solutions of this type are heavily dependent on the network topology and dynamics [MWRZ08], which limits the application of these solutions.

The use of feedback information at the encoder allows to dynamically adapt the coding decisions to the network behavior. When the encoder has complete and immediate knowledge of the reception status of the receivers, incorporating feedback information in the coding decisions leads to significant improvements in the end-to-end reliability [FLMP07]. For the case where the feedback information and the data share the same channel, [LMS12] proposes half-duplex network coding schemes that use feedback to request additional coded packets, deriving the optimal number of coded transmissions before receiving feedback. While these schemes perform coding operations block-by-block basis, [SSM08] proposes a throughput optimal network coding mechanism that combines packets from a moving coding window, which allows to reduce the required buffer sizes at the receivers.

For the case of multiple unicast flows in a wireless mesh network, [KRH$^+$08] proposes a network coding protocol that uses the fact that nodes can overhear their neighbors transmissions in order to construct linear combinations that provide new information simultaneously for multiple flows. [SM09] extends this idea to the particular case of multiple video streams. In broadcast packet erasure channels, the typical approach [JMM93, NTNB09, Lar08] is to assume that there is an error free parallel feedback channel that allows the encoder to know which packets were lost by the receivers and then adapt the coding decisions to maximize throughput. The problem of choosing which packets to combine in this setting is known to be NP-hard [ERCS07, ERSG10], which is also the case in the case when packets have strict deadlines to be delivered [ZX10].

Main Contributions and Thesis Outline

In this thesis, we set out to explore the trade-off between the throughput achieved by network coding solutions and the corresponding decoding delay experienced. We consider scenarios with different delay constraints, from the case of reducing the decoding delay as much as possible to applications where packets have to be delivered before strict deadlines. Our goal is to design network coding solutions that exploit the availability of feedback information at the encoder to provide the delay requirements of a variety of scenarios. The main contributions of this thesis are as follows.

- *Immediately Decobable Network Codes for Minimum Decoding Delay:* We propose network coding schemes that use feedback information from the receiving nodes in order to construct coded packets that are immediately decodable, i.e. that immediately provide a new information packet to the receiver. We show the impact of using immediately decodable schemes both in throughput and delay. Our results show that, although a penalty in throughput arises, the use of feedback to design immediately decodable schemes leads to a significant improvement in the decoding delay experienced by the receivers.

- *Delay Control in Online Network Coding:* We analyze the decoding delay experienced when using throughput optimal network coding schemes that make use of acknowledgments in order to move a coding window, i.e. the set of packets that are combined in the coded packets. Although this online behavior of the coding process deems these solutions more suited to real-time data, when compared with block-by-block approaches, we show that the presence of receivers with different channel conditions leads to significantly high decoding delay in the receivers with worst channels. We propose the use of the feedback information available in order to break chains of undecoded packets, which provides a trade-off between the throughput achieved and the decoding delay experienced.

- *Network Coding for Data with Strict Deadlines:* For the transmission of data with strict deadlines over a wireless channel to multiple receivers, we propose a network coding protocol that relies on periodic feedback from the receivers to adapt its coding decisions to minimize the number of packets that miss the corresponding deadline. The proposed mechanism has tunable parameters that allow the transmitter to choose the desired operating point in the trade-off between throughput and delay. The proposed protocol was implemented in a real wireless testbed, over the IEEE 802.11 standard, which asserts for the compatibility of our proposal with current wireless networking standards.

- *Optimal Coding Mechanisms for Half-Duplex Channels with Limited Time:* We analyze the problem of delivering a set of packets over a packet erasure half-duplex channel, in a limited time interval, for applications that require the reliable delivery of a subset of the source data. We show that a judicious use of the feedback channel leads to a significantly higher mean number of packets delivered in the end of the available time slots, without jeopardizing the reliable delivery of a subset of the packets. Moreover, we show that, in the case where receivers announce a list of missing packets, the use of feedback improves both the number of packets delivered and the reliability level achieved.

The remainder of the thesis is organized as follows. In Chapter 2, we analyze the decoding delay of network coding schemes for information dissemination, and we propose immediately decodable network coding solutions that reduce the decoding delay, albeit at the sacrifice of throughput. In Chapter 3, we take the opposite perspective, by analyzing the delay experienced in online network coding schemes and proposing control mechanisms that allow a trade-off between throughput and the delay experienced. The case of traffic with strict delay constraints is considered in

Chapter 4, where we propose a coding scheme that uses periodic feedback information to minimize the number of packet deadlines broken, while ensuring an efficient use of the channel. In Chapter 5, we propose optimal coding schemes for applications where there is a limited time interval to deliver a set of packets over an half-duplex channel. Finally, Chapter 6 presents the conclusions of this thesis and a discussion on possible directions for future work.

The work described in Chapter 4 has been awarded the first prize in the *Concurso Nacional de Inovação BES 2012*, one of the most prestigious national prizes for awarding innovation.

Parts of this work has been submitted to the following journals:

- Rui A. Costa, Daniel E. Lucani, Tiago T.V. Vinhoza, João Barros, *On the Use of Feedback in Lossy Half-Duplex Channels with Time Constraints*, submitted to the IEEE Transactions on Information Theory, 2012.
- Rui A. Costa, Diogo Ferreira, João Barros, *Ensuring Reliability in Real-Time Multicast with Feedback*, submitted to the IEEE Transactions on Communications, 2012.

Parts of this work has been presented at the following international conferences:

- João Barros, Rui A. Costa, Daniele Munaretto, Joerg Widmer, *Effective Delay Control in Online Network Coding*, Proceedings of the IEEE Conference on Computer Communications (IEEE Infocom 2009), Rio de Janeiro, Brazil, April 2009.
- Rui A. Costa, Daniele Munaretto, Joerg Widmer, João Barros, *Informed Network Coding for Minimum Decoding Delay*, Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2008), Atlanta, USA, September 2008.

Parts of this work has been included in an international patent application, entitled *Feedback-Based Real-Time Network Coding*, with application number PCT/IB2012/054544.

Chapter 2

Delay Sensitive Information Dissemination

2.1 Motivation

The basic idea of network coding [ACLY00, CWJ03], by which nodes transmit packets that result from joint encoding of multiple original information units, has led to communication protocols that are applicable in a wide range of wireless communication scenarios [FLBW06]. The gains brought by network coding are most evident in applications involving multicast or broadcast sessions (in which messages are intended for multiple destination nodes) in combination with physical layer broadcast (in which neighboring nodes can overhear potentially useful information).

For delay-sensitive applications such as media streaming, it is not desirable for receivers to have to wait for the arrival of several packets until they are able to decode part of the sent data. Ideally, we would like for a coded packet to immediately provide a new source packet to the receiver upon successful reception. On the other hand, each coded packet should be useful to as many receivers as possible, thus minimizing the required number of transmissions. Similar considerations hold for distributed systems with highly constrained nodes, such as sensor networks, where it may well be impossible to store a large number of coded packets and to decode them using Gaussian elimination.

In this chapter, we consider the problem of disseminating data over wireless networks in two different scenarios. In the first scenario, the single-hop case, we have a sender transmitting a set of packets to multiple receivers over a packet erasure channel, with access to cost-free feedback information that allows the sender to know which transmissions were successful. In the second scenario, the multi-hop case, each node has a packet to deliver to all the other nodes in a multi-hop wireless network, where by overhearing each others' transmissions, nodes are aware of which packets are missing at the neighboring nodes.

Although these are different scenarios, the challenge posed to the transmitting node(s) is similar in both cases. This node has to decide which packet to transmit, given the set of packets missing at the receivers, with the objective of delivering all the data as fast as possible, which is a NP-Hard problem [ERCS07, ZX10]. We will consider this problem in a setting where small delay in decoding individual packets is also crucial for performance and nodes are constrained in the amount of memory and processing they can afford, as in the case of sensor nodes. We focus our attention on algorithms that allow immediate decoding of each incoming packet. This stringent delay constraint comes at the cost of a reduction in throughput, since we restrict the coding decisions by excluding coded packets that are innovative (i.e. contain useful information) but not immediately decodable by the receivers.

Intrigued by the behavior of network coding protocols for delay constrained information dissemination, we set out to design immediately decodable network codes to achieve reduced delay in decoding individual packets. Our main contributions in this chapter are as follows.

- *Immediately Decodable Schemes:* We present two distinct coding schemes that provide immediately decodable packets. The first scheme maximizes the number of receivers that find the coded packet immediately decodable in a greedy fashion, while the second aims to equalize the number of recovered packets among neighbors. The proposed schemes operate under the assumption that neighboring nodes exchange information about which source packets they have already recovered, for example by appending this information every time they send a data packet.

- *Performance Analysis:* We compare the proposed schemes with other solutions that use various levels of buffer state information of receiving nodes. Our results show that immediately decodable schemes outperform existing algorithms in significantly decreasing the delay in decoding individual packets.

- *Gains in using a more complex decoder:* Our schemes are designed for providing immediately decodable packets and assume that receivers store only decoded packets, which implies a smaller buffer and a significantly simpler decoding process. Nevertheless, we characterize the performance gains induced by a more complex decoder that buffers all received packets and uses gaussian elimination to recover the source data from the coded packets.

The remainder of the chapter is organized as follows. In Section 2.2, we describe the most relevant related work. Section 2.3 sets the foundations for the problem of reducing delay in network coded information dissemination. In Section 2.4, we proposed two network coding algorithms for such problem, whose performance we analyze in Section 2.5. Finally, Section 2.6 offers some concluding remarks.

2.2 Related Work

In general, there is a trade-off between delay, throughput and end-to-end quality, and different codes can try to optimize either one of these performance metrics. Priority Encoded Transmission [ABE$^+$96] provides graceful degradation by specifying different levels of coding (and consequently different minimum numbers of packets required for decoding) depending on the content and the priority of the underlying information units. Fountain codes, specially Raptor codes [Sho06], offer very low coding overhead and are (asymptotically) rate optimal when transmitting over erasure channels. However, decoding n packets is only possible after $n+\epsilon$ coded packets have been received and typically nodes cannot decode any source packet before receiving those $n+\epsilon$ coded packets. A fraction of the encoded packets can be decoded earlier, albeit at the cost of significant overhead [San07].

Fostering early or immediate decoding requires an algorithm that decides which and how many source packets should be combined in each new coded packet that is to be transmitted. Adequate design of such an algorithm is highly dependent on the state information available at the nodes. Complete lack of state information is likely to occur in highly dynamic networks, such as mobile sensor networks, where the overhead of tracking a changing neighborhood would be prohibitive. In case a node has no information about the packets that have already been recovered by its neighbors, the algorithm can only optimize how many information units to combine (i.e. the codeword degree) [KMFR06]. Each node simply combines randomly chosen information received from other nodes with its own information units, until the desired codeword degree is reached. The algorithm needs to find the right trade-off between a high codeword degree that ensures that coded packets bring new information to many of the neighbors, and a low codeword degree that allows packets to be decoded immediately using only the information that is locally available. An analysis of optimum degree distributions with respect to network dynamics and topology was carried out in [MWRZ08].

When information about the data recovery status of neighboring nodes is available, it is possible to employ more sophisticated coding algorithms. One such instance is presented in [KRH$^+$08], which proposes a protocol for unicast routing in wireless mesh networks. Routers combine packets opportunistically from different sources in order to increase the diversity of the information content of each transmission. A node chooses the source packets to combine based on the content of the neighbors' buffers. This form of state information is piggybacked onto data packets and also extrapolated from past loss rate measurements and overheard packets. The procedure ensures that coded packets are immediately decodable at the next hop with very high probability. Although the protocol in [KRH$^+$08] targets unicast traffic, very similar considerations also apply to broadcast. [KDF08] analyzes a number of simple heuristics for the online and offline version of the problem.

2.3 Problem Setup

Recall that we consider two different scenarios in this chapter. In the single-hop case, one sender has a set of packets to deliver to multiple receivers over a packet erasure channel, where feedback information is available to allow the sender to know which packets are missing at each receiver. In the multi-hop case, each node has one packet to deliver to all the other nodes in a multi-hop wireless network, where by overhearing each others' transmissions, nodes are aware of which packets are missing at the neighboring nodes. Therefore, in both cases, the transmitting node is faced with the problem of deciding which packets to combine in order to satisfy a given set of requests of the receiving nodes. We formalize this problem as follows. Consider a node x with source packets $B_x = \{p_1, \ldots, p_n\}$ in its buffer, whose broadcast transmissions over the wireless medium reach its set of neighbors $N_x = \{1, \ldots, m\}$. The transmitting node x is allowed to code across source packets, which means that the coded packet to be transmitted, denoted by c, is of the form $c = \sum_{i=1}^{n} \alpha_i \cdot p_i$, where $p_i \in \mathbb{F}_q^k$ (for some vector size k) and $\alpha_i \in \mathbb{F}_q$.

The challenge for the transmitting node x is to select the coding coefficients α_i in order to allow for a fast dissemination process with low delay in decoding individual packets. Formally, the central metric we will consider is the recovery rate, defined as follows.

Definition 1 *The recovery rate of node x is a function of the number of coded packets received by node x, t_x, given by $\Delta_x(t_x) = r_x$, where r_x denotes the number of source packets recovered by node x when t_x coded packets are received.*

The recovery rate captures the two fundamental aspects we are envisioning, namely speed of dissemination process and delay in decoding individual packets. If $|p|$ denotes the total number of packets in the system[1], the optimal dissemination process in terms of speed verifies $\Delta_x(|p|) = |p|$, which means that node x is able to recover all $|p|$ source packets with $|p|$ coded packets received.

With respect to decoding delay, notice that the ideal case would be to have $\Delta_x(t_x) = t_x$, for $t_x \leq |p|$, which would mean that every coded packet received allowed receiver x to decode a new source packet. However, in general this is not possible to achieve. Schemes that achieve optimal speed of dissemination typically do not allow for any intermediate decodability, which means that their recovery rate $\Delta_x(t_x)$ for $t_x < |p|$ is small, although $\Delta(|p|) = |p|$.

In order to evaluate the effective use of the coding capabilities provided to the nodes, we will also analyze the *codeword degree*, which is measured as the number of source packets involved in the coded packet. More precisely, if $c = \sum_{i=1}^{n} \alpha_i \cdot p_i$, then the degree of c is given by the number of non-zero coefficients, i.e. by the size of the set $\{i : \alpha_i > 0\}$.

Regarding the decoding process, we consider two different mechanisms: (1) a very simple decoding scheme, which uses only the recovered source packets at the buffer to decode a received coded packet (discarding all coded packets that do not immediately provide a new source packet); and (2) the full decoding scheme, which performs Gaussian-Jordan elimination based on both coded and recovered packets that are stored in a node's buffer.

[1]For the single-hop scenario, $|p|$ is the number of packets the sender has to deliver. For the multi-hop scenario, $|p|$ is given by the total number of nodes in the network, since each node generates one packet.

2.3.1 Review of Existing Coding Algorithms

2.3.1.1 Random Network Coding

Network coding schemes often generate each coded packet in a randomized fashion by taking into consideration all the packets that are available in the send buffer. This approach requires no knowledge about the recovery status of the neighboring nodes and no pre-established degree distribution (as explained later). The most prominent representative of this class of schemes is the *Random Linear Network Coding* (RLNC) algorithm presented in [HKM+03], where the coefficients used to generate the output linear combination are chosen randomly from the prescribed field.

As an example of a RLNC scheme, we now give a formal definition of the algorithm presented in [KDF08]. This Systematic Random Network Coding scheme was designed for the scenario of a source broadcasting to n receivers over a packet erasure channel.

Algorithm 1 Systematic Random Network Coding [KDF08]

The algorithm starts by sending every packet once in uncoded form. After this first phase, the algorithm computes the output packet as a random linear combination of all the (uncoded) packets in the buffer.

Since the systematic algorithm produces random linear combinations of all the packets in the sender's buffer, the source packets can only be recovered by means of gaussian elimination and thus requires the receivers to store in the buffer all the received packets, either coded or recovered. Hence, we can only consider this algorithm with a full decoding scheme.

2.3.1.2 Degree distribution based algorithm

Network coding schemes that combine all packets in the buffer are not always optimal in terms of delay and complexity, as observed in [KMFR06]. The authors show that depending on the number of recovered packets r at a specific node, there exists an optimal number of packets to combine to maximize the number of decodable packets at each instant in time. More precisely, the authors of [KMFR06] determine the optimum degree distribution. For a prescribed number of recovered packets r, the degree distribution $D(r)$ returns the codeword degree of the next output packet.

The scenario studied in [KMFR06] is a random encounter scenario, in which each node meets independently at random one neighbor at a time. The optimum degree distribution depends on the dynamics of the underlying network and in [MWRZ07] the authors show the deficiencies of such an algorithm in scenarios beyond the specific model they were designed for. Since [KMFR06], some other coding algorithms based on a pre-defined degree distribution were proposed. As a comparison algorithm, we use the following instance.

Algorithm 2 Adaptive Network Coding (ANC) [MWRZ08]

```
  When a transmission opportunity occurs, the node randomly combines a specific
  number of packets in its buffer, which ensures that the degree of the resulting output
  packet is as high as possible and less than or equal to $D(r)$ (as defined above).
```

2.3.1.3 Opportunistic Algorithm

The previous algorithms do not make use of feedback information in terms of the recovery state of neighboring nodes. However, if available, this additional information can be used to make more efficient coding decisions and can bring significant improvements to the overall performance. The work presented in [KDF08] analyzes how feedback information can be used and proposes a number of heuristic algorithms designed for the scenario of a source node broadcasting to n receivers over independent erasure channels. Among the algorithms proposed in [KDF08], we consider the so called opportunistic algorithm, because it is the only one that can be used in conjunction with a simple decoder. The basic idea is that each node uses the feedback received from its neighbors to compute a queue of source packets that have not yet been received by at least one node. The first packet is chosen randomly and further packets are added under the condition that the packet remains immediately decodable by all neighbors that were previously able to decode it (in other words, the number of nodes that can decode the packet can only increase). This algorithm operates over the binary finite field, $\mathbb{F}_2$.

Before defining a different set of algorithms capable of exploiting the knowledge of the recovery status of neighboring nodes, we must introduce some basic notation. Recall that we denote by x the node executing the coding algorithm, where $B_x = \{p_1, \ldots, p_n\}$ denotes the set of source packets at the buffer of x and $N_x = \{1, \ldots, m\}$ denotes the set of neighbors of x. We denote by B_j the set of source packets that are in the buffer of node j, with $j \in N_x$. The set of source packets that are in the buffer of x and that are not in the buffer of neighbor j is denoted by $\overline{B}_j$ and can be computed as $\overline{B}_j = B_x \backslash (B_j \cap B_x)$.

The algorithms presented next start by choosing the set of source packets C that will be combined in the output packet by means of an XOR operation. The set C is constructed iteratively, such that in each step of the algorithm one packet is added to the previously constructed set. $\overline{C}$ denotes the set of source packets that node x has in its buffer and that are not in C (it can be calculated as $\overline{C} = B_x \backslash C$).

Finally, we will denote by $R(C)$ the set of neighbors of node x that are able to recover a new packet from the XOR of all packets in a given set C. This means that, if there are y source packets in set C, the neighbor must have already recovered exactly $y-1$ of the packets in C. Therefore, we have that $R(C) = \{j : |C| - |B_j \cap C| = 1\}$.

We are now ready to give a formal definition of the Opportunistic scheme, presented in [KDF08].

Algorithm 3 Opportunistic algorithm [KDF08]

```
$C = \emptyset$
$S = \{p \in B_x : |R(\{p\})| > 0\}$
**while** $|S| > 0$ **do**
    choose $p^* \in S$
    add $p^*$ to $C$
    $S = \left( \bigcap_{j \in R(C)} (B_j \cap B_x) \right) \cap \overline{C}$
**end while**
$c = \bigoplus_{p \in C} p$
transmit $c$
```

As shown in *Algorithm 3*, the set C is initialized to the empty set. Throughout the algorithm, set S represents the set of packets that can be added to the current configuration of set C. Initially, all the packets that are in the buffer of node x and that are not in the buffer of at least one neighbor are deemed to be possible candidates. Thus, because $R(\{p\})$ is the set of neighbors that has not recovered packet $p \in B_x$, S is initialized according to $S = \{p \in B_x : |R(\{p\})| > 0\}$.

Next, we focus on the loop in the algorithm, which will only continue while S is non-empty. As long as there are source packets in S, the algorithm chooses one of them randomly and adds it to C. After this step, it is necessary to update S, from which the algorithm chooses new packets to add to C.

The new set S is defined as the set of source packets that satisfy the following conditions: (1) they are present in the buffers of the neighbors in $R(C)$ (i.e. those

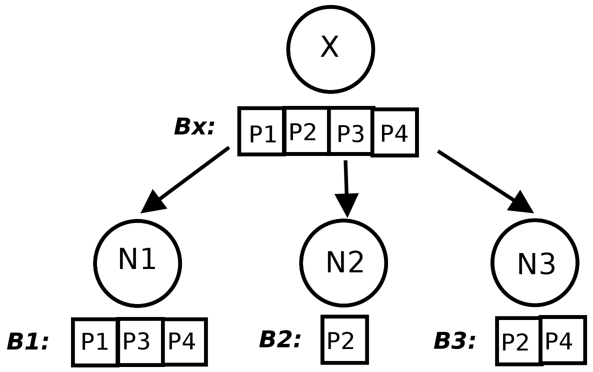


Figure 2.1: An example of a network, in which node **X** has to make coding decisions based on the buffer state of **3** neighbors: $\mathbf{N_1}$, $\mathbf{N_2}$ and $\mathbf{N_3}$. For each neighbor, the figure depicts only that part of the buffer which contains source packets that are also stored in the buffer of node **X**.

neighbors that are able to recover a new packet from the current set C), (2) they are stored in the buffer of node x; and (3) they were not chosen up to this step. Thus, the new set S can be determined according to

$$S = \left(\bigcap_{j \in R(C)} (B_j \cap B_x) \right) \cap \overline{C}.$$

When the loop finishes, all the source packets in set C are XORed together and the corresponding packet is sent to the neighbors of node x.

The ideas behind the algorithm are best described using the example shown in Figure 2.1. Node x (the node performing the coding algorithm) has in its buffer source packets p_1, p_2, p_3 and p_4. From Figure 2.1, it is clear that some of these packets can also be found in the buffers of the three neighbors of X. It follows that there are only two possible coded packets which are optimal in the sense that they maximize the number of neighbors able to recover a new packet immediately upon reception: $c = p_1 \oplus p_2$ or $c = p_2 \oplus p_3$. In both these cases, all neighbors are able to recover a new packet from the received packet, since they have one and only one of the packets therein. Moreover, no other combination of packets can provide a packet that immediately provides a new packet to every neighbor.

We now analyze the behavior of the Opportunistic algorithm. The initial set S (from which we can choose a packet to be mixed in the packet) is $S = \{p_1, p_2, p_3, p_4\}$, due to the fact that none of the neighbors has recovered all the packets in the buffer of node X. Thus, in the first iteration, each packet in S can be chosen with probability 1/4.

Suppose that the algorithm chooses $p^* = p_1$ (again with probability 1/4). Then, we have that $C = \{p_1\}$ and $R(C) = \{N_2, N_3\}$. Recall that $R(C)$ is the set of neighbors that have recovered all but one of the packets in C (in this case, it is the set of neighbors who have not yet recovered packet p_1). Since S is the set of packets that (a) all the neighbors in $R(C)$ have already recovered and (b) have not yet been chosen, we have that $S = \{p_2\}$. In the second iteration, since $S = \{p_2\}$, the algorithm chooses $p^* = p_2$ and sets $C = \{p_1, p_2\}$. Thus, $R(C)$ is equivalent to the entire set of neighbors and, since there are no more source packets recovered by the ensemble of neighboring nodes, we have that $S = \emptyset$. Hence, the algorithm stops and outputs the packet $c = p_1 \oplus p_2$, which can be classified as an ideal packet.

We have seen that the algorithm outputs an ideal coded packet if the first chosen symbol corresponds to p_1 (and that this happens with probability 1/4). Analogously, if the algorithm chooses p_3 first, then $C = \{p_3\}$ and $R(C) = \{N_2, N_3\}$, yielding $S = \{p_2\}$. Hence, in the next step the algorithm chooses packet p_2 which will lead to $S = \emptyset$. It follows that if the algorithm starts by choosing symbol p_3, then we get the ideal output packet $c = p_2 \oplus p_3$.

Suppose now that the algorithm starts by choosing packet p_2. In this case, we have that $C = \{p_2\}$ and $R(C) = \{N_1\}$. Since S is the set of unselected packets recovered by the ensemble of neighbors in $R(C)$, we have that $S = \{p_1, p_3, p_4\}$. Hence, in the second iteration, the algorithm has a probability 1/3 of choosing each of the packets in S. If the algorithm chooses packet p_1 (respectively, packet p_3), based on the same arguments as in the previous cases, we deduce that the output packet will be $c = p_1 \oplus p_2$ (respectively, $c = p_2 \oplus p_3$), which is an ideal packet. In case the algorithm chooses p_4, the output will not be an ideal packet. Thus, the probability that the algorithm outputs an ideal packet is given

$$\frac{1}{4} + \frac{1}{4} + \frac{1}{4}\left(\frac{1}{3} + \frac{1}{3}\right) = \frac{2}{3}.$$

It is worth noting that in this algorithm, the sole criterion for the choice of source packets to be mixed in the output coded packed is to ensure that a node which is able to recover a new packet from the current set C (constructed up to the given iteration), will continue to be able to recover a new packet from the instances of C that are constructed after that iteration. In other words, after the choice of the first packet (which is performed randomly), the algorithm simply ensures that the number of neighbors that are able to recover a new source packet does not decrease with the next decisions.

2.4 Optimized Coding Algorithms

In the following, we present two algorithms for the encoding process, both based on the knowledge of the recovery status of the neighboring nodes. In order to increase the speed of information dissemination, our algorithms make coding decisions that by design allow the neighboring nodes to recover another information unit immediately upon reception of a new coded packet. As in *Algorithm 3*, the coding operations amount to bit-wise XORs of the source packets.

2.4.1 Greedy algorithm

The first algorithm gives priority to the source packets that are rarest within the neighborhood. The key is to find the combination of source packets that maximizes the number of neighbors that are able to decode a new information unit.

Algorithm 4 Greedy algorithm

```
$C = \emptyset$
choose $p^* = \arg\max_{p \in B_x} |R(\{p\})|$
$q = 0$
while $|R(C \cup \{p^*\})| \geq q$ do
    $q = |R(C \cup \{p^*\})|$
    add $p^*$ to $C$
    choose $p^* = \arg\max_{s \in B_x \setminus C} |R(C \cup \{s\})|$
end while
$c = \bigoplus_{p \in C} p$
transmit $c$
```

As shown in *Algorithm 4*, the choice of the first source packet is very different from the Opportunistic algorithm (*Algorithm 3*). Instead of a random choice, the Greedy algorithm selects the source packet that maximizes the number of nodes that are able to decode a new packet if a packet of degree one is sent. This corresponds to maximizing $|R(\{p\})|$. If there are multiple packets that satisfy this condition, the algorithm chooses one of them randomly. As we will see later on, a proper choice of the first packet is crucial for a good performance. In fact, we will show that, if the nodes send packets of degree one (i.e. source packets) and use the selection criteria of our protocols, the resulting performance is already quite close to the performance of the Opportunistic algorithm.

Taking a closer look at the loop of this algorithm, we realize that after choosing the first packet, the algorithm proceeds by selecting the source packet that maximizes the number of neighbors able to decode a new source packet from the coded packet, which is obtained by XORing this new packet with all the packets selected so far. This can be written as $|R(C \cup \{p\})|$. After choosing this candidate packet ($p^* = \arg\max_{p \in B_x \setminus C} |R(C \cup \{p\})|$), the algorithm will check if there is a gain in adding this candidate packet to the set of packets to be mixed in the output coded packet. Notice that, for the algorithm to continue, we do not require that neighbors that could previously recover a new packet will continue to be able to do so; the algorithm continues as long as the number of neighbors able to recover a new packet does not decrease from one step to the next one.

Denote by c^* the coded packet obtained by XORing all the source packets chosen so far (i.e., all the packets in the current set C) and denote by p^* the candidate symbol. If the number of neighbors that are able to decode a new packet from $c^* \oplus p^*$ (which is represented by $|R(C \cup \{p^*\})|$) is less than the number of neighbors that are able to decode a new packet from c^* (which is represented by q), i.e. if $|R(C \cup \{p^*\})| \geq q$, the algorithm stops and produces a packet that combines all of the source packets selected thus far.

Going back to the scenario illustrated in Fig. 2.1, we see that the algorithm starts by choosing the packet p^* that maximizes the size of $R(\{p\})$ over all p in the buffer of node X, i.e. that maximizes the number of neighbors that do not have the source packet p^*. Clearly, p^* is the *rarest packet in the neighborhood.* It follows that the algorithm ends up choosing p_1 or p_3, since each of them is present in the buffer of only one of the neighbors. If the algorithm chooses packet p_1, we have that $|R(\{p_1\})| = 2$. In the first iteration, the algorithm sets $q = 2$ and $C = \{p_1\}$. Next, the algorithm selects the packet p^* as the one that maximizes the size of $R(C \cup \{p\})$ over all $p \neq p_1$. More specifically, it will choose the source packet that maximizes the number of neighbors that are able to recover a new source packet from the coded packet obtained when XORing this candidate packet with all the packets in C. In this case, since $C = \{p_1\}$ and all 3 neighbors can recover a new packet from $p_1 \oplus p_2$, this candidate packet is $p^* = p_2$. Now, the algorithm checks if the number of neighbors that can recover a new packet increases when compared to the previous step. In this case, since $q = 2$ neighbors recovered a new packet and adding the candidate symbol increases this number to 3 (i.e. $|R(C \cup \{p_2\})| \geq 2$), the algorithm continues by updating q to $q = 3$ and adding the candidate packet to the packet: $C = \{p_1, p_2\}$. Now, the algorithm chooses the next candidate packet using the same rule, i.e. to maximize the number of neighbors that are able to recover a new packet. In this case, this packet can be p_3 or p_4. In either case, we have that only one neighbor will be able to recover a new packet if the candidate packet is added, thus we will have $|R(C \cup \{p^*\})| = 1$. In the subsequent step, since $|R(C \cup \{p^*\})| < 3 = q$, the algorithm stops and outputs the packet $c = p_1 \oplus p_2$, which is an ideal coded packet.

Notice that in the first choice we had two options: p_1 and p_3. We saw that if p_1 is chosen, the algorithm outputs the ideal packet $c = p_1 \oplus p_2$. Using analogous arguments, it is easy to see that if p_3 is chosen in the first step, the algorithm outputs the packet $c = p_2 \oplus p_3$, which is also an ideal packet. Thus, we have that in this example, with probability 1, the Greedy algorithm outputs an ideal packet.

Similarly to the Opportunistic algorithm, the Greedy algorithm evolves in each iteration by selecting a packet to be added to the set of source packets that will form the output coded packet. After the choice of the first packet, the algorithm ensures that the number of neighbors that are able to decode new packet does not decrease with the next decisions. Beyond the choice of the first packet (which has a significant impact on the performance as we will see latter on), the selection procedure targets the source packet that will maximize the number of neighbors that are able to decode, whereas the Opportunistic algorithm make this selection in a purely random fashion.

2.4.2 Equalizing algorithm

The Greedy algorithm presented in the previous section is prone to lead to an uneven distribution of information. In the worst case, some nodes that are not well connected to the rest of the network might receive mostly packets they cannot decode, since they lack some of the information units that all the other nodes already have. These nodes would be served by the greedy algorithm only after all other nodes have decoded all of the information, leading to a high worst case delay. The way to prevent this from happening is to equalize the recovery level among the neighbors instead of maximizing it in a greedy fashion.

The Equalizing algorithm pursues mainly the goal of giving new decodable information to the neighbors that have recovered the fewest information units, thus increasing the minimum number of recovered packets per node.

In *Algorithm 5*, $R^*(C) = \{j : C \subseteq B_j\}$ represents the set of neighbors that have all the source packets in C. In each step, the algorithm chooses the neighbor that has the least recovered packets among those not yet considered. Then, the algorithm selects

Algorithm 5 Equalizing algorithm

```
$C = \emptyset$
$B = B_x$
$R^*(C) = \{j : C \subseteq B_j\}$
while $|B| > 0$ and $|R^*(C)| > 0$ do
    choose $J = \arg\min_{j \in R^*(C)} |B_j|$
    $S = B \cap \overline{B}_J$
    choose $p^* = \arg\max_{p \in S} |R(C \cup \{p\})|$
    add $p^*$ to $C$
    $B = B \cap B_J$
end while
$c = \bigoplus_{p \in C} p$
transmit $c$
```

one of the packets that this particular neighbor has not yet recovered (and that all the previously chosen neighbors did recover, thus ensuring that the previously chosen neighbors can still decode the packet). This packet is added to the coded packet to be sent.

The algorithm needs to keep track of the source packets that neighbors chosen so far have already recovered. This is captured by set B. One condition to stop the loop of the algorithm is precisely the existence of packets in B. If there are no packets in B, i.e. if there is no source packet that has been recovered by all the nodes chosen up to a certain iteration, no packet can be added to the coded packet to be sent without rendering at least one of the neighbors unable to decode. The other condition for the loop to stop is $|R^*(C)| > 0$, which means that the loop only continues if there are still neighbors that have recovered all the packets in the output packet constructed so far. If there are no neighbors in this condition, no further nodes will be able to recover a new packet, irrespective of which source packet is added to the coded packet.

In each iteration, the algorithm starts by inspecting all nodes that have recovered all the source packets in the coded packet constructed so far (i.e. neighbors in $R^*(C)$ which implies that no neighbor can be chosen twice) and finding the one that recovered the least number of packets. More specifically, we choose the neighbor J that satisfies $J = \arg\min_{j \in R^*(C)} |B_j|$. After making this selection, the algorithm calculates the set of source packets that can still be added to the coded packet. These packets must have been recovered by all the previously chosen neighbors (i.e. packets in B) and cannot have been recovered by the neighbor that was chosen in the current iteration (i.e. packets not in B_J). Thus, the set of candidates is defined by $S = R \cap \overline{B}_J$.

Next, from this set of candidate packets, the algorithm selects the one that maximizes the number of neighbors that are able to decode a new packet, assuming that the output coded packet results from the XOR of all packets in C. After this choice, the algorithm adds the packet to the set C and updates the set R. The new set R will be the set of packets shared by all the neighbors that were chosen before the current iteration (namely R) and possessed by the new chosen neighbor (B_J), i.e. $R = R \cap B_J$. When the loop is completed, the algorithm computes the packet to be sent by XORing all the packets in C.

Once again, we will use the scenario in Fig. 2.1 to clarify the main steps of the algorithm. The algorithm starts by setting $C = \emptyset$, $B = \{p_1, p_2, p_3, p_4\}$ and $R^*(C) = \{N_1, N_2, N_3\}$ (recall that $R^*(C)$ represents the set of neighbors that have already recovered *all* of the packets in C). In the first iteration, the algorithm starts by checking which neighbor has the smallest buffer, i.e. the one with the smallest number of recovered packets. In this case, the chosen neighbor is N_2, since it only recovered packet p_2. Then, the algorithm computes the set of packets that this node does not have in its buffer: $S = \{p_1, p_3, p_4\}$. The goal is to provide a new source packet to this particular neighbor. Hence, the first chosen packet is a packet from S and, since we also want to provide (if possible) new packets to other neighbors, the algorithm chooses the packet that is more rare within the neighborhood, among all the packets in S. In this case, we have two options: p_1 or p_3. Suppose that the algorithm chooses $p^* = p_1$. We have that $C = \{p_1\}$ and $B = \{p_2\}$, i.e. B is the set of packets that node N_2 has already recovered. It is necessary to keep track of the packets that all the neighbors chosen by the algorithm have already recovered to ensure that the neighbors with the smaller number of recovered packets will be able to recover a new source packet from the resulting coded packet.

Next, in the second iteration, the algorithm chooses the neighbor that has the smallest number of recovered packets among all the neighbors that have all the source packets in the current instance of set C, i.e. among the neighbors in $R^*(C)$. In this case, $R^*(C) = \{N_1\}$ and thus the chosen neighbor is N_1. Now, the set of packets that can be added to C is the set of all source packets that all the previously chosen neighbors have in their buffers, B, and that the neighbor chosen in the current iteration does not have in its buffer, $\overline{B}_1$. Thus, in this case, we have that $S = \{p_2\}$ and, hence, $C = \{p_1, p_2\}$. Notice that there are no further source packets that have been recovered by all the chosen neighbors, i.e. $B = \{p_2\} \cap \{p_1, p_3, p_4\} = \emptyset$. Thus, the algorithm cannot continue and consequently outputs the packet $c = p_1 \oplus p_2$, which is an ideal packet.

In the first iteration, we could have chosen packet p_3 instead of p_1. Using similar arguments, it is easy to see that, if p_3 is chosen, the algorithm outputs the packet $c = p_2 \oplus p_3$, which is also an ideal packet. Therefore, we again have that with probability 1 the Equalizing algorithm outputs an ideal packet in our example.

2.5 Simulation Results

In this section, we present and discuss the performance of the aforementioned coding algorithms in various scenarios. The main part of the analysis assumes the simple decoding algorithm. We discuss the enhancement of performance provided by the use of a full decoding scheme at the end of this section. Regarding the evaluation metrics, we average the recovery rates and the codeword degrees over all nodes per each number of received packets. We used a custom C++ simulator, which provides an ideal (collision-free) MAC layer, with a sequential or random scheduling of packets. We repeat the simulations as many times as necessary to get tight confidence intervals for the recovery rate. For the recovery rate shown in the next plots, the 95% confidence intervals are all within ±2% of the average value. These intervals are omitted from the figures for the sake of readability.

2.5.1 Single-hop Scenario

In this setting, a single source node broadcasts 100 source packets to its 100 neighbors over independent erasure channels and, as in [KDF08], perfect feedback is available from the receivers to the source. For all the algorithms in our analysis, i.e. Greedy, Equalizing, Opportunistic and ANC, the source node starts by sending out all the source packets in uncoded form. It is only after this initial stage that the source node sends encodings of the source packets. The erasure probability is set to 0.5. We now consider only the simple decoding case.

In Figure 2.2, the Greedy algorithm shows the best performance in the single-hop scenario. With this algorithm, all nodes achieve the full recovery of the 100 source packets within 120 packets received. From the first 100 uncoded packets the receivers miss around 50 original packets and they differ from node to node due to the random erasure pattern. This allows the Greedy algorithm to increase the degree of the coded

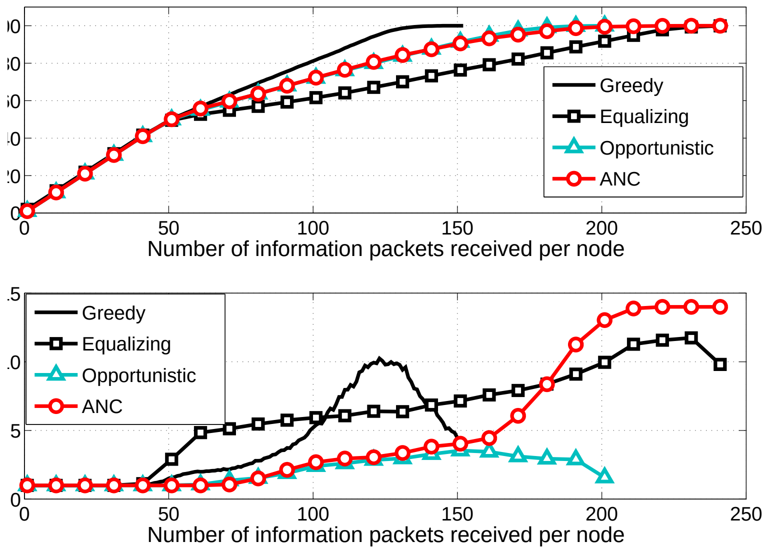


Figure 2.2: Recovery rate (top) and average codeword degree (bottom) for the single-hop scenario.

packets compared to the Opportunistic and ANC algorithms between 50 received packets and 120 received packets. When the process approaches the full recovery state, the number of nodes still missing some packets decreases and low degree codewords are sufficient to serve these nodes.

The Equalizing algorithm has a considerably worse recovery rate than the other algorithms. The reason is visible in Figure 2.2, bottom, where Equalizing starts using higher codeword degrees quite early on. Since it is designed to provide an immediately decodable packet to the neighbor(s) which recovered the least number of source packets, many other neighbors are not able to decode the packet — their composition of recovered packets differs from those poor nodes. Focusing only on the poor nodes results in a packet that despite its high degree is useful only for few receivers.

The performance of ANC and of the Opportunistic algorithm is almost the same for most of the simulation. The Opportunistic algorithm, which allows the source node to use the neighborhood status information to make the coding decisions, performs just slightly better than ANC.

Notice that these results exhibit significant losses in terms of throughput. Optimal schemes would deliver all the 100 source packes with 100 packets received. However, as we will see, these looses are due to the use of a simple decoder.

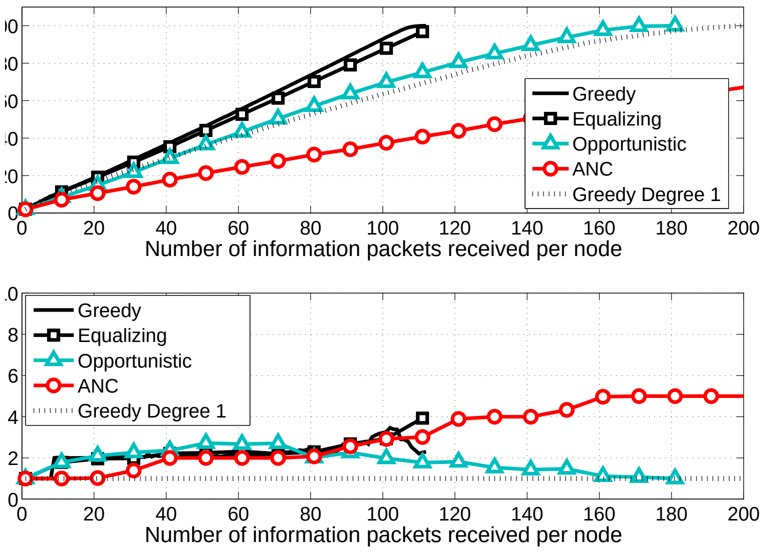


Figure 2.3: Recovery rate (top) and average codeword degree (bottom) for 100 nodes on a static grid.

2.5.2 Multi-hop Scenario

In the multi-hop scenarios, each of the 100 nodes generates one source packet that is intended to be delivered to every other node in the network.

2.5.2.1 Static grid

In this setting, nodes are placed on a static grid (that wraps around) and each node has eight neighbors to communicate with. In Figure 2.3, top, the algorithm with the best performance is the Greedy algorithm (as in the previous scenario), but now the difference to the performance of the Equalizing algorithm is much smaller. From the analysis of their respective average codeword degrees, in Figure 2.3, bottom, we see that the coding degree of our two algorithms is very similar, except in the end where the Equalizing algorithm takes longer than the Greedy algorithm to increase the coding degree for recovery of the last missing packets.

The high degree of correlation of the information recovered by the neighbors, due to the wrap around and the symmetrical topology, and the consequently minor diversity of information stored by the neighbors compared to the single-hop setting, makes the use of packets with high degree ineffective. Moreover, choosing which source

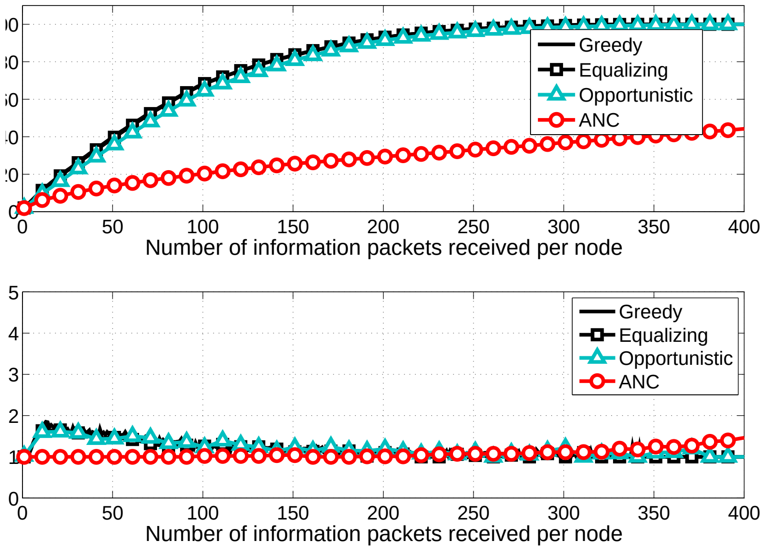


Figure 2.4: Recovery rate (top) and average codeword degree (bottom) for the static random network, 100 nodes.

packets to combine has a huge impact on the performance of the dissemination process. To visualize this, we also computed the recovery rates achieved by the algorithms corresponding to Greedy and Equalizing when we limit the codeword degree to one, i.e., only an uncoded packet is sent. In the top graph of Figure 2.3, we plot only Greedy with codeword degree one, but both of the algorithms perform the same. Even in this limited case, where no coding is allowed, the recovery rates of our algorithms are very close to the recovery rate of the Opportunistic algorithm *with* coding. The few degrees of freedom for making coding decisions, typical of this setting, limit the performance of the Opportunistic algorithm, where the first source packet is chosen randomly.

Finally, the impact of using neighborhood recovery status in the coding decisions is obvious when we compare the recovery rate of the ANC algorithm with the recovery rates of the other algorithms. For instance, the total number of received packets necessary to achieve full recovery is, in the case of ANC, several times larger than in the case of Greedy, while the average codeword degree is quite similar for most of the values of received packets.

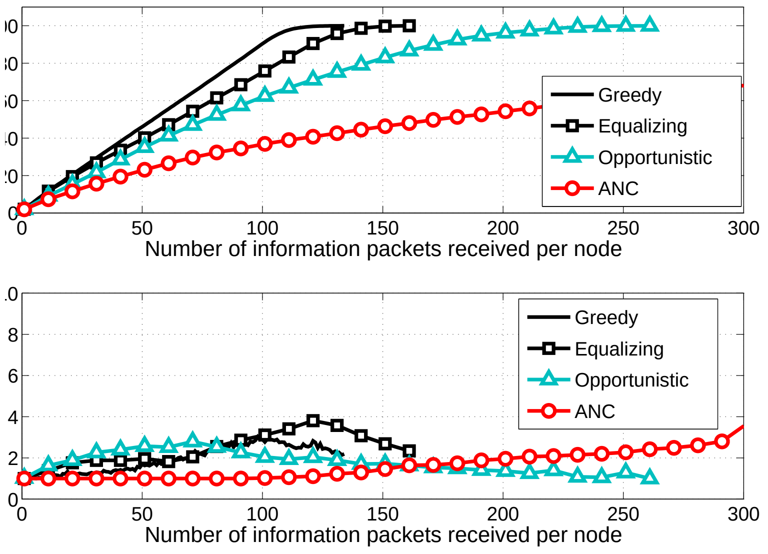


Figure 2.5: Recovery rate (top) and average codeword degree (bottom) for the mobile scenario, 100 nodes.

2.5.2.2 Static random and clustered networks

In this section, we consider two different scenarios: static random topologies with an average density of 8 nodes per communication range, and clustered networks. These static networks are relatively sparse, which means that the diversity of information stored by the neighbors is lower. The high degree of correlation among the source packets recovered by the nodes explains why the Greedy, Equalizing and Opportunistic algorithms perform similarly (Figure 2.4). No degree of freedom for making specific coding decisions is left to these algorithms, so that the differences are small. Also in such settings, ANC cannot perform well given the extremely low level of diversity of information among nodes.

2.5.2.3 Moderate mobility

In this scenario, we consider nodes moving according to a random waypoint mobility model with speeds uniformly distributed in the interval $[2, 4]$ m/s. Again, the node density allows on average for eight neighbors per node. We assume perfect information about the neighbor recovery status.

In Figure 2.5, top, we notice that the performance of the algorithms under consideration in terms of recovery rate is somewhat similar to the one observed in the static grid setting (Figure 2.3). Due to the mobility of the nodes, the correlation among the source packets recovered by neighbors is much lower in the case of moderate mobility than in the case of a static grid or static random networks.

It is also important to notice that the coding degree of the Equalizing algorithm is always higher than the coding degree of the Greedy algorithm. This observation and the fact that the recovery rate of the Greedy algorithm is higher than the recovery rate of the Equalizing algorithm let us conclude that the Equalizing algorithm builds packets with a too high codeword degree, rendering these packets not immediately decodable. We will see next that, if we allow the use of a full decoding process, the recovery rate of the Equalizing algorithm can actually surpass the recovery rate of the Greedy algorithm.

2.5.3 Performance gains using a complete buffer decoding mechanism

In this section, we investigate the benefits of using a full decoding mechanism, which is more efficient (in the sense that it does not discard useful packets) but also more costly in terms of energy, memory requirements, and processing. Up to now, we were considering a scheme were only the recovered source packets were considered for the simple decoding process. Now, all the packets received (decoded and undecoded) are stored in the buffer and thus taken into consideration when performing the decoding of the received packets, using gaussian elimination. In the following figures, we omit the plot of the average codeword degree for the sake of readability of the recovery rate. Also, the average codeword degree of the algorithms using a full decoding scheme is almost the same as the one with the simple decoder.

As we already mentioned in the previous analysis, it is expected that the recovery rate significantly increases with the full decoding scheme, since the algorithms often produce packets that are not immediately decodable for some neighbors but that are innovative. The node is not able to recover a new source packet from the received coded packet since it did not yet recover the required other source packets that form the coded packet. By storing these not immediately decodable but innovative packets in the buffers and considering these packets in the decoding process, nodes can find these packets helpful later on, when more and more (innovative or immediately decodable) packets are received, increasing the recovery rate of the algorithm. This benefit can be

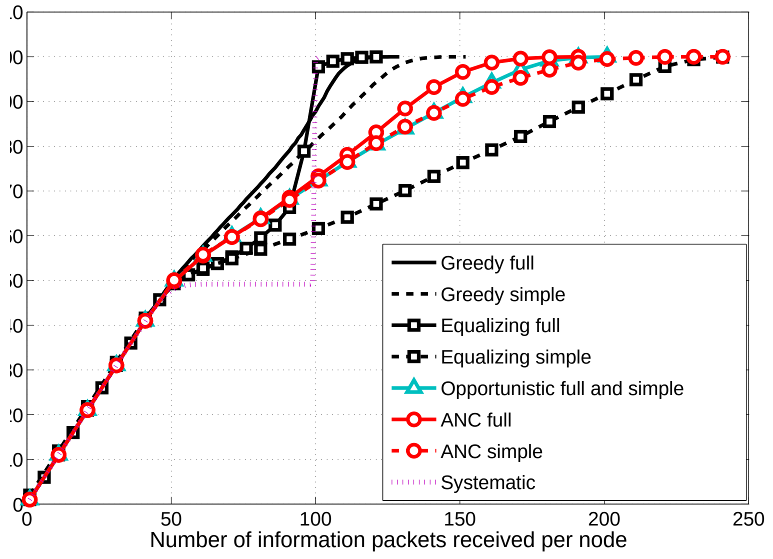


Figure 2.6: Recovery rate for the single-hop scenario with simple and full decoding schemes.

observed in Figure 2.6, where the recovery rate of the algorithms using a full decoding mechanism (including the Systematic Random Network Coding algorithm) is plotted for the single-hop scenario.

Comparing the results obtained with this full decoding scheme to the results obtained with a simple decoding scheme, we can observe a major improvement of the recovery rate of our algorithms, with special emphasis in the Equalizing algorithm. With the full decoding scheme, the Equalizing algorithm has a recovery rate that slowly increases after the initial phase (where the nodes first send the source packets uncoded once) and, at around 80 packets received, shows a smooth step behavior that is typical of the random network coding algorithms (e.g. Systematic Random Network Coding). The Equalizing algorithm reaches the full recovery state before the Greedy algorithm, with an average of 110 packets received. However, the recovery rate of the Greedy algorithm is higher than the one of the Equalizing algorithm before the step behavior takes place. The loss for throughput optimal schemes is now much smaller than in the simple decoder case, with the Systematic Random Network Coding reaching the full recovery state with 100 received packets, as expected.

After analyzing the performance enhancements achieved by using a full decoding scheme in the single-hop scenario, we now discuss the results obtained for the multi-hop scenario with moderate mobility. We have chosen this particular setting of the

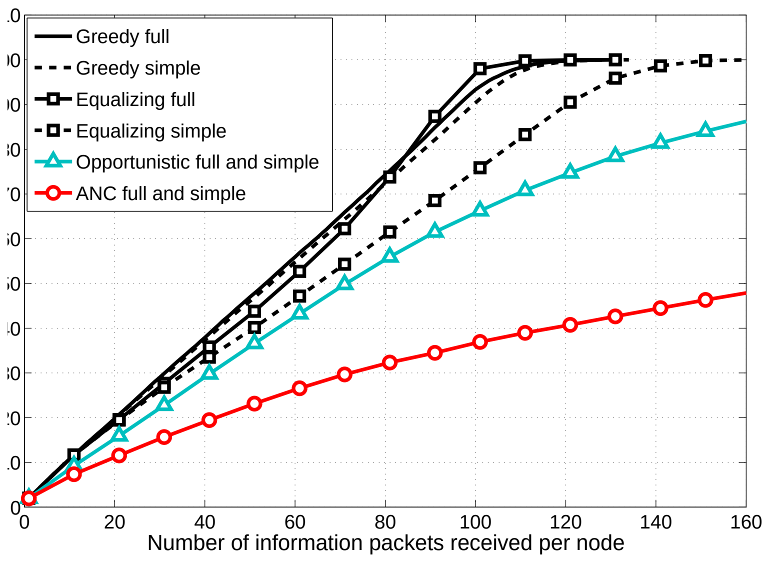


Figure 2.7: Recovery rate for the multi-hop mobile network, with simple and full decoding schemes.

multi-hop scenario because the performance of the algorithms in the other settings is similar to the one presented in Figure 2.3, although there are some differences that should be pointed out. With a full decoding scheme, the Greedy and Equalizing algorithms have quite similar performance. Moreover, the enhancements achieved by the other algorithms when using a full decoding scheme are negligible except for the ANC algorithm, for which the performance is still far from the performance achieve by our algorithms.

In Figure 2.7, we can again see that in the moderate mobility scenario and with a full decoding scheme, there is a major improvement of performance of the Equalizing algorithm. The recovery rate of the Equalizing algorithm comes very close to the recovery rate of the Greedy algorithm until around 80 packets received (similar to the behavior in the single-hop scenario). After this value, the Equalizing algorithm is faster in recovering new original packets, reaching the full recovery state 10 packets before the Greedy algorithm. It is also interesting to notice that there is no significant difference in terms of recovery rate between the two decoding mechanisms in combination with the Greedy algorithm. This behavior was expected, since the Greedy algorithm was designed for immediate decoding. Few not immediately decodable packets mean that a complete buffer decoding mechanism can just slightly outperform a simple decoder.

2.6 Concluding Remarks

In this chapter, we proposed two coding algorithms which exploit feedback information on the recovery status of neighboring nodes. Through the analysis of a wide range of settings in our simulations, we show that the Greedy algorithm consistently outperforms all other algorithms in terms of number of immediately decodable packets, which is fundamental for delay-sensitive applications in wireless networks such as real-time media streaming. Moreover, satisfactory results of the Greedy algorithm are already obtained using just a simple decoder, whereas for the Equalizing algorithm the use of Gaussian elimination improves the performance significantly.

The algorithms proposed in this chapter focus on immediate decodability and, hence, take the decoding delay as the sole optimization criterion. Naturally, this implies that we incur in a throughput loss, since the coding decisions are limited in scope. However, if we allow receivers to store all the received coded packets and use gaussian elimination for decoding, we showed that this loss can be significantly reduced, while achieving considerable improvements in the delay experienced in decoding individual source packets.

Chapter 3

Decoding Delay Control

3.1 Motivation

In Chapter 2, we have seen that using information about the decoding status of the receivers brings significant benefits in terms of the delay experienced in decoding individual information packets, although it requires sacrificing throughput. As in the classical network coding approaches, the schemes described in the previous chapter operate in a block-by-block basis, which may further deteriorate the delay experienced by the receivers, specially in real-time applications, where a sliding window approach is more suited to deal with delivering part of the data to the application, even before the entire block is received. The presence of feedback information at the encoder opens the door for the design of coding schemes that use a sliding window mechanism to decide which packets to combine, in order to provide useful coded packets to every receiver, while controlling the delay experienced in decoding individual packets. In this chapter, we will focus our attention in throughput optimal online coding schemes and the decoding delay induced by such mechanisms.

Realizing that existing solutions do not yet cover the full range of trade-offs between throughput and delay, in particular when users experience different packet loss probabilities, we set out to provide end-to-end delay control for online network coding with feedback. Our main contributions in this chapter are as follows.

- *Delay Analysis:* We provide an analytical framework to evaluate the delay performance of online network coding algorithms that leverage feedback for increased reliability. The novelty of our approach lies in observing how each erasure event affects the chains of undecoded linear combinations that build up at the receiver buffer. Moreover, we can map the information backlog between receivers to an appropriate random walk on a high dimensional lattice, which brings further insight into the delay behavior.

- *Online Network Coding Algorithms with Delay Constraints:* Using the random walk model, we propose two modifications to the coding module of online network coding algorithms, by transmitting uncoded packets at key time slots. These recovery slots allow the transmitter to alleviate the delay of weaker receivers. The number of recovery slots controls the trade-off between throughput and delay.

The remainder of the chapter is organized as follows. In Section 3.2, we discuss the most relevant related work. Section 3.3 introduces terminology and describes the core ideas of online network coding with feedback. Our analytical framework for evaluating the end-to-end delay is outlined in Section 3.4 with results on the relationship between erasure patterns, undecodable chains, and incurred delays. This framework is used in Section 3.5 to analyze the trade-off between throughput and delay when using recovery slots to tackle the differences in delay among receivers. Finally, Section 3.6 concludes the paper.

3.2 Related Work

In the seminal paper of Ahlswede, Cai, Li, and Yeung [ACLY00], which shows that network coding is necessary to achieve the multicast capacity of a general network, the problem is formulated in an information-theoretic setting, where delay and complexity are not taken into account. Delay is also not a primary concern of the algebraic framework in [KM03] and of the random linear network coding method [HKM$^+$03, CWJ03]. When intermediate nodes cannot perform coding operations and applications are able to tolerate some delay, fountain codes (e.g. [Sho06]) emerge as a viable solution offering low coding overhead as well as near-optimal throughput over packet erasure channels.

Clearly, in all of these instances, coding is performed in a feedforward fashion. The encoders upstream are oblivious to packet loss downstream and their coding decisions do not exploit any feedback information. In contrast, the property that transmitted packets are linear combinations of subsets of packets available at the sender buffer suggests that network coding protocols can be enhanced if feedback information is available at the encoder, thus allowing for informed coding decisions. Recent contributions that pursue this idea (e.g. [FLMP07, KDF08]) focus mostly on end-to-end reliability with perfect feedback, i.e., complete and immediate knowledge of the packets stored at each receiver. The source node reacts by sending the most innovative linear combination that is useful to most destination nodes. Throughput optimal network coding protocols following this concept appear in [SSM08]. By using the feedback information to move a *coding window* along the sender buffer instead of mixing fixed sets of packets (also called generations [CWJ03]), these protocols perform *online* network coding in the sense that they adapt their coding decisions based on what the destination actually receives.

Our work differs from [SSM08] in that we consider heterogeneous users with different erasure probabilities. In contrast with [SSM08] and [KDF08], which consider only two receivers, we consider also a larger number of users. We believe that our algorithms are able to reach a larger set of operating points in the delay-throughput plane and are thus well suited for streaming applications with stringent delay requirements.

3.3 Essential Background

3.3.1 System Setup

Suppose that a single queue sender wants to transmit a stream of packets to multiple receivers. For simplicity, we assume that packets arrive at the sender at a rate of one packet per time slot. Each receiver i is connected to the sender via a separate packet erasure channel i, which takes one packet per time slot and loses a packet with probability ϵ_i. Packets are lost independently across channels and time slots. Receivers are able to detect when a packet is missing. Since the sender has access to perfect feedback (without errors, losses, or delay), its encoding decisions can be based on the buffer state of each receiver.

3.3.2 ARQ for Network Coding

The reference system for our analysis is the *ARQ for Network Coding (NC-ARQ)* scheme presented in [SSM08]. This coding approach was shown to be throughput optimal for the case of perfect feedback. The sender transmits linear combinations of

Table 3.1: Example of Online Network Coding with ARQ

Time Slot	Sent Packet	Receiver 1	Receiver 2
1	$\mathbf{p_1}$	OK	E
2	$\mathbf{p_1} \oplus \mathbf{p_2}$	OK	OK
3	$\mathbf{p_2} \oplus \mathbf{p_3}$	OK	OK
4	$\mathbf{p_3} \oplus \mathbf{p_4}$	OK	OK
5	$\mathbf{p_4} \oplus \mathbf{p_5}$	OK	E
6	$\mathbf{p_4} \oplus \mathbf{p_6}$	OK	OK
7	$\mathbf{p_6} \oplus \mathbf{p_7}$	E	OK
8	$\mathbf{p_7}$	OK	OK
9	$\mathbf{p_5} \oplus \mathbf{p_8}$	OK	OK
10	$\mathbf{p_8} \oplus \mathbf{p_9}$	E	OK
11	$\mathbf{p_9}$	OK	E
12	$\mathbf{p_9} \oplus \mathbf{p_{10}}$	OK	OK

the packets in its queue, whereby the decision over which packets to combine capitalizes on the concept of *seen packets*. A packet $\mathbf{p}$ is said to be seen by a receiver, if the receiver is able to construct a linear combination of the form $\mathbf{p} + \mathbf{q}$, such that $\mathbf{q}$ is a linear combination of packets that are newer than $\mathbf{p}$. The sender always transmits a packet that is a combination of the oldest *unseen* packets of each of the receivers. This ensures that the last unseen packet will now be seen by all receivers which obtain the coded packet.

A packet can be dropped from the sender queue whenever it was *seen* (but not necessarily decoded) by all receivers. This has the agreeable property that queue sizes at the sender are kept small, since the sender can drop packets before they are decoded at all receivers. The expected queue size was shown to be $O((1-\epsilon)^{-1})$ [SSM08]. The basic operation of this scheme is illustrated in Table 3.1, which lists the sequence of packet receptions (OK) and erasures (E) [SSM08], while showing the corresponding coding decisions made by the sender for a two receiver case. Here, the receiver delays the request for packets that are unseen specifically because all combinations containing that packet were lost. In the example of Table 3.1, this happens in time slot 7, when receiver 2 requests $\mathbf{p_6}$ and not $\mathbf{p_5}$, which results in the transmission of $\mathbf{p_6} \oplus \mathbf{p_7}$. Packet $\mathbf{p_5}$ is only requested at time slot 9, after receiver 2 decoded $\mathbf{p_6}$.

Table 3.2: List of Possible Erasure Events

Event	Description		Probability
A	Receiver 1 - OK	Receiver 2 - OK	$P(A) = (1-\epsilon_1)(1-\epsilon_2)$
B	Receiver 1 - OK	Receiver 2 - E	$P(B) = (1-\epsilon_1)\epsilon_2$
C	Receiver 1 - E	Receiver 2 - OK	$P(C) = \epsilon_1(1-\epsilon_2)$
D	Receiver 1 - E	Receiver 2 - E	$P(D) = \epsilon_1\epsilon_2$

3.4 Delay Analysis of ARQ for Network Coding

In the following, we shall focus on the decoding delay of a packet, which is measured as the number of slots between the arrival of the packet to the sender's queue and its successful decoding at the receiver.

Definition 2 *Let p_i be the packet that arrived at the sender queue in time slot i. We say that p_i has a delay of d time slots at receiver R if R decodes p_i in time slot $i+d$.*

We start with the two receiver case and then proceed to develop a random walk framework for analyzing network coding delay.

3.4.1 The Two Receivers Case

Without loss of generality, suppose that the sender restricts its transmissions to uncoded packets or XORs of two packets [SSM08]. The incurred decoding delay depends only on the rules enforced by the online network coding algorithm and the erasure patterns of the two channels. In each time slot we have one of the four erasure events listed in *Table 3.2*, where OK represents an error free transmission and E represents an erasure in the channel. They occur with the given event probabilities.

An erasure event causes a receiver buffer to build up a *chain* of length K, which we define as a set of K independent linear combinations involving $L > K$ symbols, that cannot be decoded by the receiver. Going back to the example shown in *Table 3.1*, receiver 2 suffers from losses (denoted by E) in time slots 1 and 5, whereas receiver 1 obtains everything error free (OK) except for the data transmitted in slot 7. Each erasure sets a mark for a new chain of undecodable linear combinations, such that each chain begins immediately after its preceding chain has been solved. For example, up to slot 4 receiver 2 built up the chain $\{\mathbf{p_1} \oplus \mathbf{p_2}, \mathbf{p_2} \oplus \mathbf{p_3}, \mathbf{p_3} \oplus \mathbf{p_4}\}$. The erasure in slot 5 sets a mark for a new chain, which will involve $\mathbf{p_5}$ by necessity. However, before that chain begins, the first chain grows to $\{\mathbf{p_1} \oplus \mathbf{p_2}, \mathbf{p_2} \oplus \mathbf{p_3}, \mathbf{p_3} \oplus \mathbf{p_4}, \mathbf{p_4} \oplus \mathbf{p_6}, \mathbf{p_6} \oplus \mathbf{p_7}\}$. Since receiver 1 experiences an erasure in slot 7, the encoding rule forces the sender to transmit packet $\mathbf{p_7}$ in uncoded fashion, which in turn allows receiver 2 to break its first chain and recover packets $\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_6}, \mathbf{p_7}\}$. The second chain begins immediately in slot 9 with $\{\mathbf{p_5} \oplus \mathbf{p_8}\}$, because packet $\mathbf{p_5}$ was not seen by receiver 2 in slot 5. Note that an erasure event at the leading receiver 1 is not enough to allow receiver 2 to break the current chain. If the following packet is lost at receiver 2, as shown in the example with the loss of $\mathbf{p_9}$ in time slot 11, the chain will simply continue to grow. Clearly, the decoding delay is deeply influenced by the length of chains such as these and by the sender's ability to break them in a timely manner — even with randomly occurring packet erasures.

Assuming that channels 1 and 2 have erasure probabilities ϵ_1 and ϵ_2, respectively, and that the sender follows the simple ARQ rules outlined in Section 3.3, we can describe the chain duration in a probabilistic fashion.

Proposition 1 *After an erasure of type B that starts a chain at receiver 2, the chain remains unbroken for a duration of T slots with a probability given by*

$$P(T) = \epsilon_1(1-\epsilon_2)^2 \sum_{t_1=0}^{T-1} (\epsilon_1\epsilon_2)^{t_1} \cdot \sum_{t_2,t_3:2t_2+t_3=T-1-t_1} (\epsilon_1\epsilon_2(1-\epsilon_1)(1-\epsilon_2))^{t_2} \cdot (1-\epsilon_1)^{t_3}. \quad (3.1)$$

Proof: We start by observing that events of type D only increase the delay until the chain can be decoded but do not otherwise affect the recovery process or the length of the chain. Therefore, there is nothing to lose from ignoring events of type D at first and taking their impact into account only at a later stage. Taking into account events of type A, B or C only, we conclude that a chain starting with an erasure is only broken after an erasure event of type C (in which receiver 2 obtains a packet missed by receiver 1) immediately followed by an event of type A or C, in which receiver 2 obtains the uncoded symbol that will ultimately allow it to decode the chain. While the chain is unbroken, any occurrence of event pairs that are not CC or CA will add to the duration of the chain. Any occurrence of D at any slot (including between the first and second events of the pairs we considered previously) will further increase the chain duration.

Since the channel erasures are independent from slot to slot, we can think of all the occurrences of D, none of which affects the breaking of the chain in any way, as a

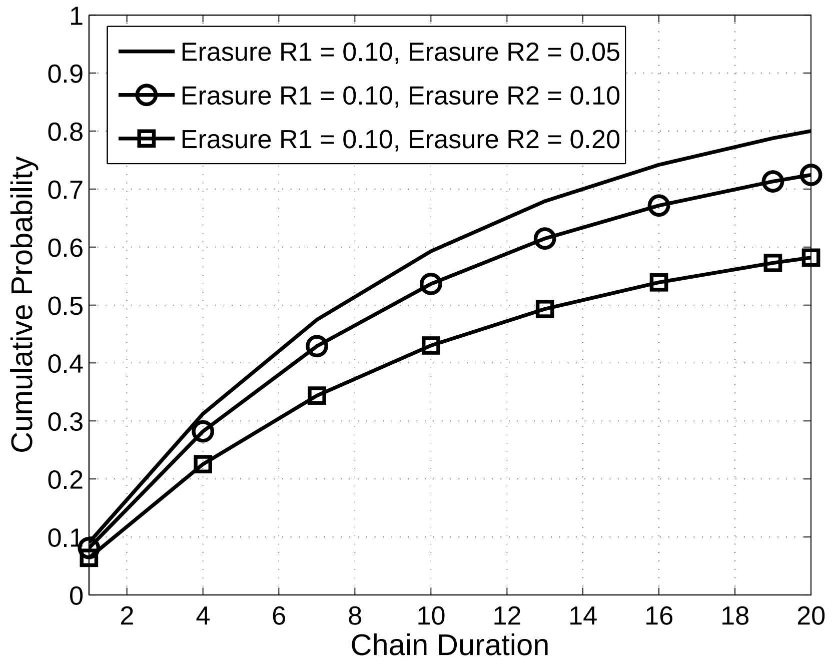


Figure 3.1: Chain Duration at Receiver 2 after an erasure of type B, from *Proposition 1*.

contiguous block in the first slots after the erasure. With this assumption in mind, notice that for the chain to be broken $T+1$ slots after the erasure, in slot T we must observe C, since the only pairs that break a chain are CA and CC. Thus, after the first slots (where D was observed) and up to and including slot $T-2$, if we observe a C, it must be followed by the event B. Regarding the remaining slots, we can have isolated events A and B (only the ones not preceded by C, because those are already taken into account). Therefore, letting t_1, t_2 and t_3 represent the number of occurrences of the events D, CB and $A \cup B$, respectively, we have that

$$P(T) = \sum_{t_1=0}^{T-1} P(D)^{t_1} \sum_{t_2,t_3:2t_2+t_3=T-1-t_1} P(CB)^{t_2}(P(A)+P(B))^{t_3} \cdot (P(CA)+P(CC)) \tag{3.2}$$

Notice that, since erasures in different slots are independent of each other, we have that $P(CA) = P(C)P(A)$, $P(CB) = P(C)P(B)$ and $P(CC) = P(C)^2$. Thus, substituting $P(A)$, $P(B)$ and $P(D)$ by the expressions in *Table 3.2* in (3.2) and simplifying the resulting terms we obtain Equation (3.1). ■

In Figure 3.1, we present the cumulative probability distribution of the chain duration T. We fix the erasure probability for receiver 1, $\epsilon_1 = 0.1$ and vary the erasure probability ϵ_2 of receiver 2. We can observe that ϵ_2 has a clear impact on the chain duration. As expected, a worst channel in receiver 2 results in a larger chain duration in case an erasure of type B occurs and receiver 1 is the leader, which is more likely to occur in cases where receiver 2 has the worst channel.

Likewise, we can compute the distribution for the chains built up at receiver 1.

Proposition 2 *After an erasure of type C, a chain at receiver 1 remains unbroken for a duration of T slots with probability*

$$P(T) \quad = \quad \epsilon_2(1-\epsilon_1)^2 \sum_{t_1=0}^{T-1} (\epsilon_1\epsilon_2)^{t_1} \cdot \sum_{t_2,t_3:2t_2+t_3=T-1-t_1} (\epsilon_1\epsilon_2(1-\epsilon_1)(1-\epsilon_2))^{t_2} \cdot (1-\epsilon_2)^{t_3}. \quad (3.3)$$

Proof: The proof follows analogously to the previous proof by swapping events B and C. ■

3.4.2 Delay Bounds Through Random Walks

As should be expected, determining the various forms of delay becomes increasingly complex for larger numbers of receivers. To gain some insight, we start by observing that, at any point in time there will be one or more leaders, defined as the receivers that obtained the maximum number of packets up to time slot t. The following proposition describes an important property of the leader status.

Proposition 3 *If a receiver becomes a leader at time t and remains a leader until getting one more packet at time $t+\delta$, then it is able to decode all packets included in any of the linear combinations transmitted until $t+\delta$.*[1] *The receiver continues to be able to decode all coded packets immediately at time $t' > t+\delta$, provided it remains a leader.*

Proof: Suppose that leaders lose a packet at time t, such that another receiver can become a leader. Assume also that they received the first d packets up to that time, which carry encodings of at most the first $d+1$ information units. Clearly, leaders would have been able to decode a new packet, had they received the current

[1] A receiver may become a leader when it receives a packet and the leaders do not. However, if the next packet or packets are lost and the receiver drops out of the set of leaders before receiving another packet, then it will not be able to break its chain. This is analogous to the events CB and CD in the example above.

transmission. One or more receivers can now become leaders upon receiving a packet at time t. Since the coding algorithms are throughput optimal [SSM08] (i.e., each received packet is innovative), new leaders that receive a packet at time $t+\delta$ will have $d+1$ (coded) packets with combinations of the first $d+1$ original packets. They can thus solve the corresponding system of linear equations and decode all packets. ■

As shown in the example of *Table 3.1* for time slot 8, also non-leaders may break chains and decode packets. However, as the number of receivers increases and/or the erasure probabilities become more heterogeneous, the probability that non-leaders can decode becomes very small.

Describing the system in terms of the packets received by each of the receivers leads to a state space that grows exponentially in the number of receivers and is therefore intractable. However, we can use the fact that leaders can decode all packets to derive an upper bound on the decoding delay in the multiple receiver case. As we have seen, the decoding delay is tightly connected to the time interval between the moment in which a receiver ceases to be a leader and the moment it is able to catch up and regain the leader status.

It is useful to describe the evolution of the differences in received packets between the receivers as a random walk in an $(n-1)$-dimensional lattice, where n is the number of receivers.

Definition 3 (Random Walk Model) *For $i = 1, \ldots, n$, let $r_i(t)$ be the number of successful receptions at receiver R_i at the end of time slot t. The $(n-1)$-dimensional random walk position in time slot t $X(t)$ is given by $X(t) = (r_1(t) - r_2(t), r_1(t) - r_3(t), \ldots, r_1(t) - r_n(t)) \in \mathbb{Z}^{n-1}$.*

To develop some intuition, consider the case of three receivers, denoted R_1, R_2, and R_3. Let $x_1(t) = r_1(t) - r_2(t)$ denote the difference of received packets between R_1 and R_2, and let $x_2(t) = r_1(t) - r_3(t)$ describe the difference between R_1 and R_3. The state of the system is thus described by the pair $X(t) = (x_1(t), x_2(t))$, which can be viewed as a point in two-dimensional space. In each time slot, there are eight possible erasure events, depending on whether each of the receivers suffers a packet loss or not. If, in a given time slot, all receivers lose a packet or if there are no packet losses, then the state does not change. In all other cases, x_1 and x_2 will increase or decrease by one unit according to the transition rules in *Table 3.3*, where once again we use OK and E to denote successful reception and packet erasure, respectively. Once we associate

Table 3.3: Transition Rules for the Three-Receiver Case

Event	R_1	R_2	R_3	Next State	Direction
E_0	OK	OK	OK	(x_1, x_2)	$\cdot$
E_1	OK	OK	E	(x_1, x_2+1)	$\uparrow$
E_2	OK	E	OK	(x_1+1, x_2)	$\rightarrow$
E_3	OK	E	E	(x_1+1, x_2+1)	$\nearrow$
E_4	E	OK	OK	(x_1-1, x_2-1)	$\swarrow$
E_5	E	OK	E	(x_1-1, x_2)	$\leftarrow$
E_6	E	E	OK	(x_1, x_2-1)	$\downarrow$
E_7	E	E	E	(x_1, x_2)	$\cdot$

the erasure events with the corresponding probabilities, we obtain a random walk on a two-dimensional lattice, as illustrated in Figure 3.2.

Clearly, R_1 is a leader if and only if the coordinates $(x_1(t), x_2(t))$ lie in the first quadrant. In this case, $x_1(t)$ and $x_2(t)$ are both non-negative and R_1 has received either the same or a higher number of packets than the other receivers by time slot t. R_1 ceases to be a leader, when its state position moves from the first quadrant to one of the other three. Conversely, it becomes a leader again if its state position moves back to quadrant one. With respect to the other receivers, R_2 is a leader if and only if the position lies in the region $\{(x_1(t), x_2(t)) : x_1(t) \leq 0 \cap x_1(t) \leq x_2(t)\}$, and R_3 is a leader in the region $\{(x_1(t), x_2(t)) : x_2(t) \leq 0 \cap x_2(t) \leq x_1(t)\}$.

The proposed random walk model proves to be very useful for computing upper bounds on the decoding delay experienced by each receiver. Given the position of the random walk at a certain time slot, we can determine which receivers are leaders and, by Proposition 3, we may conclude that (if no erasure occurred in the current time slot) a new packet is decoded by these leading receivers, with delay given by the number of erasures experienced by the leaders so far.

In the case that a receiver is not a leader in the current time slot, even if it has received a packet, we assume that it is not able to decode any new packet. Notice that non-leading receivers may decode packets, as shown in *Table 3.1* for time slot 8. Hence, with this methodology, we obtain an upper bound on the decoding delay of the packets, which follows from the assumption that a receiver can only decode packets when it reaches the leader status. Furthermore, we are now able to compute tight approximations of the decoding delay distribution of each receiver.

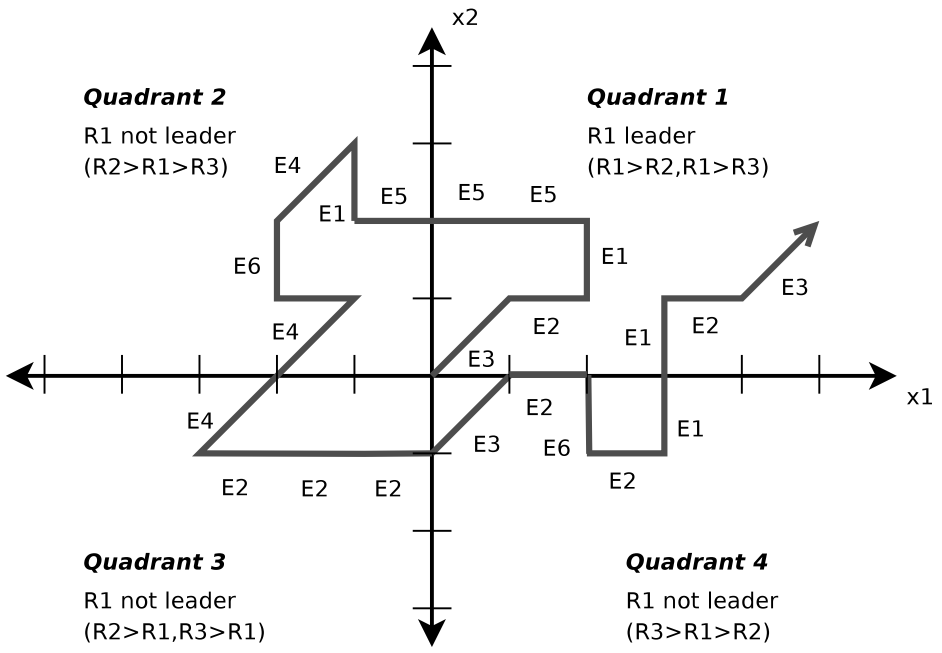


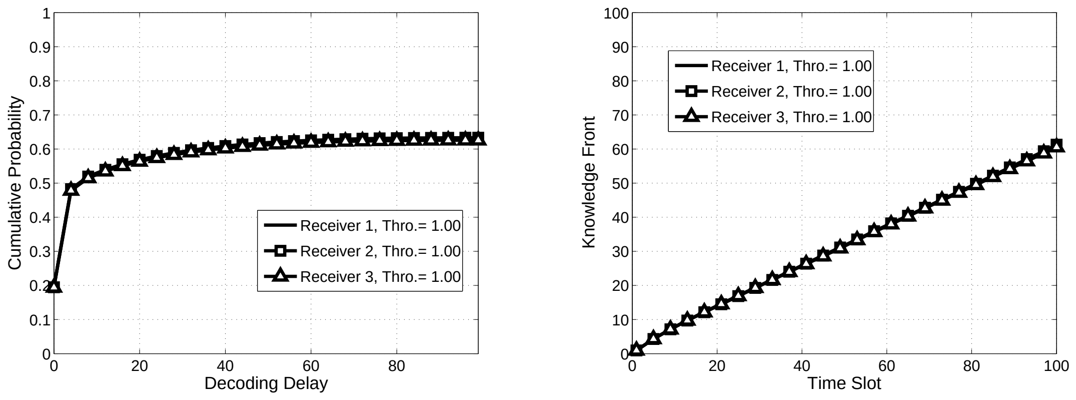
Figure 3.2: Random walk interpretation of the state evolution at receiver R_1.

Notice that *Proposition 3* taken together with the structure of NC-ARQ, which dictates that the source transmits a linear combination of the last unseen packet of each receiver, leads us to the conclusion that the *knowledge front* of a leader, as defined below, increases by one at each time slot, provided that the transmission was successful.

Definition 4 *The knowledge front of receiver r at time slot t is given by $kf_r(t) = i^*$, where i^* is the index of the most recent decoded packet whose predecessors have all been decoded by receiver r at time slot t.*

In other words, the knowledge front is equivalent to the amount of data that was already delivered to the application layer *in the correct order.* In the worst case, the knowledge front of non-leading receivers does not increase until they become leaders. Therefore, given that the leadership status depends only on the number of erasures, it follows that the random walk can be used to obtain a lower bound on the knowledge front of each receiver.

It is worth pointing out that generalizing this idea from three receivers to n receivers forces us to consider random walks in n-dimensional lattices. The resulting class of random walks can be deemed as untypical on several counts: (a) they assign non-uniform probabilities to different directions by virtue of the properties of online network coding, and (b) they admit the possibility that a node stays in the same

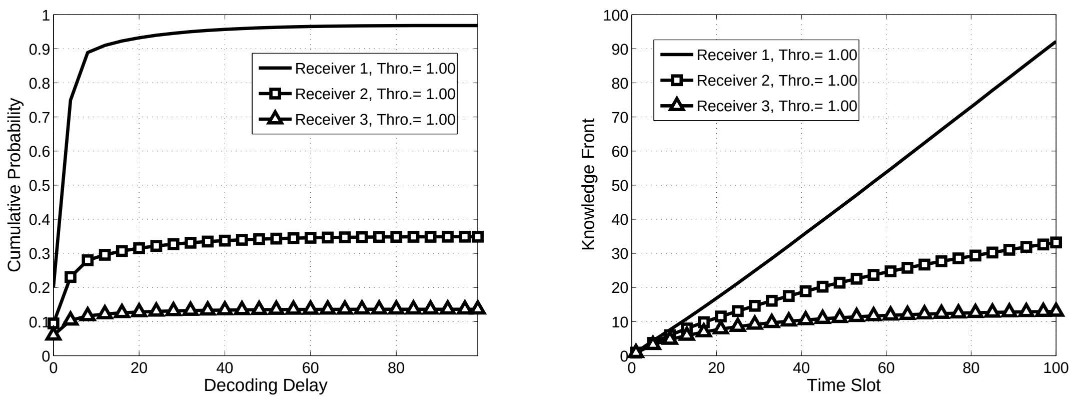


(a) CDF for Decoding Delay (b) Knowledge Front

Figure 3.3: The CDF for the decoding delay and the knowledge front for the case of a homogeneous set of receivers, with all the erasure probabilities equal to 0.05.

position. Close inspection of the related literature in probability theory reveals that the complete mathematical characterization of integer random walks — even for uniform distributions in two dimensions — offers non-trivial difficulties. A large body of work is concerned with the number of points covered by the random walk up to a certain time (see e.g. [CH96]), other contributions focus on hitting times on the coordinate axis [Coh92] or among multiple random walks [AF06]. At this time, providing a mathematical description of the crossing times between quadrants of a non-uniform random walk is clearly a daunting task, which justifies the use of numerical techniques at the final stage of the proposed analysis.

In the following, we present results obtained from simulation of the random walk model up to 100 time slots for the three receiver case, by generating 10000 independent random instances of the channels of the receivers for each of the 100 time slots. We present both the approximation for the cumulative density function (CDF) for the decoding delay and the lower bound for the knowledge front, as well as the throughput experienced by each receiver (here throughput is measured as the fraction of successful receptions that provided a new degree of freedom). Given that we are forced to limit the number of time slots, nodes may consequently decode different sets of packets. To overcome this limitation and ensure a fair comparison, each packet that is not decoded by a receiver until the last time slot is assigned a delay value equal to the number of time slots (here, equal to 100). Throughout this work, we measure the throughput of a node as the fraction of the received packets that are innovative for the receiver. As shown in [SSM08], NC-ARQ is throughput optimal, i.e. for all the receivers we have throughput equal to 1.



(a) CDF for Decoding Delay (b) Knowledge Front

Figure 3.4: The CDF for the decoding delay and the knowledge front for the case of a heterogeneous set of receivers, with $\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$.

In Figure 3.3(a) and Figure 3.3(b), we present the results obtained for the case where the three receivers have an erasure probability of 0.05. As expected, all the receivers present similar results, in both evaluation metrics. Notice that, although the algorithm is throughput optimal and that leaders are always able to decode the packets upon successful reception, after 100 time slots the receivers succeeded in decoding only approximately 60 packets. This can be explained by the fact that, although the receivers have the same erasure probability, a single extra erasure is sufficient for a receiver to lose leader status. This in turn implies that the receiver is unlikely to decode packets until it regains the leadership status.

Now, consider Figure 3.4(a) and Figure 3.4(b), where we present the results obtained for a set of heterogeneous receivers ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$). We can now observe how the heterogeneity of the receivers affects the delay experienced by the different receivers. Receiver 1 has the best channel ($\epsilon_1 = 0.05$, as in the previous case) and presents now a better performance. This is clearly visible in its knowledge front. In fact, almost every received packet resulted in a new decoded symbol, which is a direct consequence of the fact that receiver 1 has the best channel and is thus the leading receiver most of the time. As for the performance of the other receivers, we observe a clear loss in performance, which is due to the fact that, with a worse channel, these receivers are unlikely to be leaders. This results in a small number of decoded packets after 100 time slots.

We have seen that NC-ARQ, in the scenario of heterogeneous receivers, induces high decoding delay in the receivers with worst channel conditions. Any effort to control the

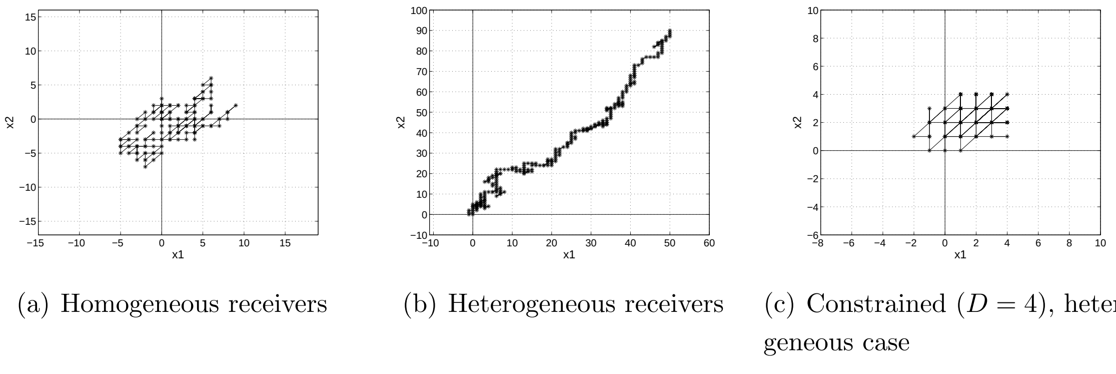

(a) Homogeneous receivers (b) Heterogeneous receivers (c) Constrained ($D = 4$), heterogeneous case

Figure 3.5: Illustration of the random walk for 1000 time slots, for (a) homogeneous ($\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.05$) and (b) an heterogeneous set of receivers ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$) without constrains and (c) with $D = 4$.

delay by means of informed coding decisions must amount to creating opportunities for non-leading nodes to achieve the leader status. In our random walk interpretation, this is equivalent to pushing the random walk to the origin, whenever it approaches the maximum acceptable distance to the origin. Achieving this goal in practice is the topic of the next section.

3.5 Delay Control Using Recovery Transmissions

As already discussed, in the presence of heterogeneous receivers, given that the probability that non-leaders decode is small, receivers with bad channels experience heavy delays. This behavior can also be observed in the corresponding random walk. For the case of homogeneous receivers (Figure 3.5(a)), the random walk naturally concentrates around the origin, which implies that the delay experienced by the different receivers are similar. But for the case of heterogeneous receivers (Figure 3.5(b)), the random walk gets uncontrollably far away from the origin, which in terms of delay corresponds to arbitrarily large differences between the delay of each of the receivers.

3.5.1 NC-ARQ with Hard Recovery

To tackle the *unfairness* in delay induced by NC-ARQ, it is natural to restrict the random walk to a region close to the origin. This clearly leads to similar delays for all receivers. A way to constrain the random walk is to ensure that the worst receiver gains advantage on the leaders, whenever the difference of the number successful receptions between the best receiver and the worst receiver grows above a certain threshold value. The following rule implements this principle.

Algorithm 6 NC-ARQ with Hard Recovery

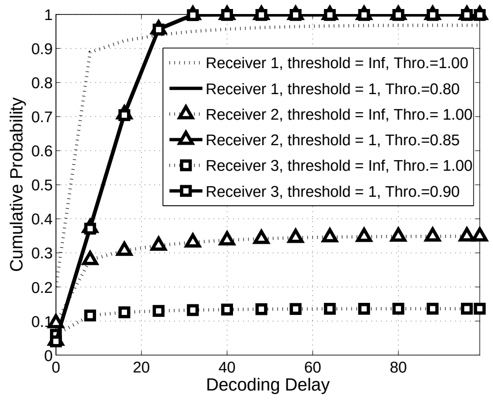
Let $S_i(t)$ be the number of successful transmissions received by receiver i up to time slot t. In each time slot t, if $\max_{i,j}(S_i(t) - S_j(t)) > D$, $D \in \mathbb{N}$, transmit the oldest undecoded packet of the worst receiver; otherwise proceed with standard NC-ARQ.

We can also use the random walk representation for this algorithm. Whenever the threshold D is reached, for all receivers except the worst, the transmitted packet will not be innovative with high probability. Therefore, we can model this recovery stage as an erasure on the channel of all receivers except for the worst. An illustration of the resulting random walk can be observed in Figure 3.5(c). With this approach, we force the random walk to concentrate around the origin, thus bringing closer the delay behavior of the different receivers. But, given that we introduce transmissions that are not innovative for every receiver, we pay a price in terms of throughput.

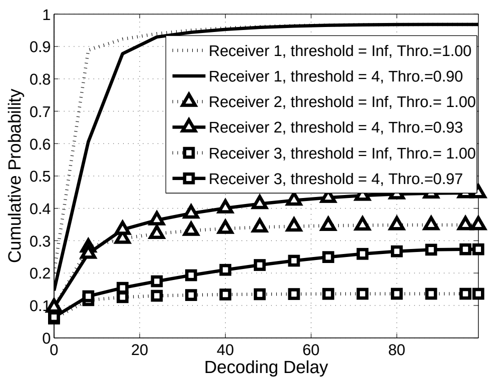
In Figure 3.6(a), we present the results obtained using the random walk methodology for the case of heterogeneous receivers, with a tight threshold value: $D = 1$[2]. The delay distribution is now the same for all the receivers, but we observe some loss in throughput, which is due to the transmission of recovery packets that may be not innovative for every receiver. Receiver 1 (the one with the best channel) has a throughput of 0.80, whereas receivers 2 and 3 show throughputs of 0.85 and 0.90, respectively. Naturally, using a higher value for the threshold will result in less recovery slots and, thus, a smaller number of non-innovative packets. But the impact on the delay is of course smaller. In Figure 3.6(b), we present the results obtained for $D = 4$. The number of non-innovative packets is reduced (receiver 1, 2 and 3 obtained a throughput of 0.90, 0.93 and 0.97, respectively). On the other hand, the delay distributions suffer a smaller improvement.

Regarding the knowledge front, for $D = 1$ the results obtained using the aforementioned approach (Figure 3.7(a)) exhibit a similar behavior for the three receivers, although their channels are different. The receiver with the best channel gets penalized, but the worst receivers get their performance clearly improved, at a price of significant losses in throughput. In Figure 3.7(b), we observe that using $D = 4$ leads to smaller losses in throughput, at the cost of a smaller increase of the knowledge front of the receivers with the worst channels.

[2]Notice that for $D = 1$ the algorithm works as a standard ARQ scheme for the broadcast channel: the source transmits always the oldest undecoded packet among the set of receivers.

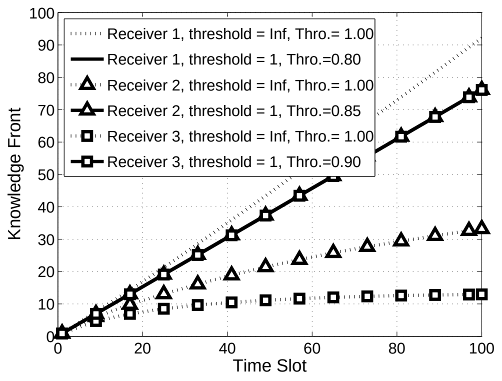


(a) $D = 1$

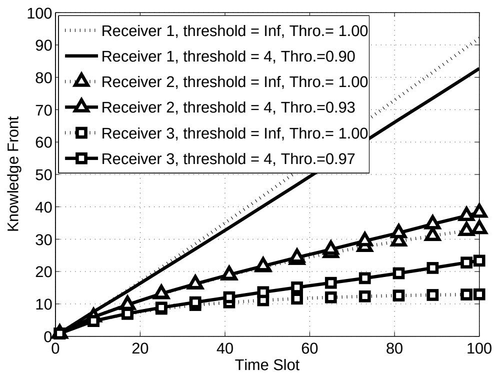


(b) $D = 4$

Figure 3.6: CDF of the decoding delay for the scenario of heterogeneous receivers ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$), for the standard NC-ARQ and the NC-ARQ with Hard Recovery with two different threshold values.

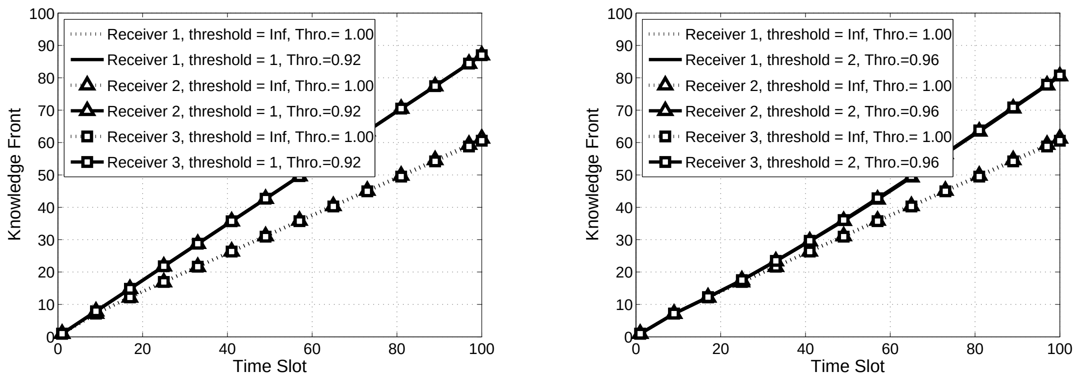


(a) $D = 1$



(b) $D = 4$

Figure 3.7: Knowledge front for the scenario of heterogeneous receivers ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$), for the standard NC-ARQ and the NC-ARQ with Hard Recovery with two different threshold values.



(a) $D = 1$ (b) $D = 2$

Figure 3.8: Knowledge front for the scenario of homogeneous receivers ($\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.05$), for the standard NC-ARQ and the NC-ARQ with Hard Recovery with two different threshold values.

We have so far seen that introducing the recovery mechanism in NC-ARQ allows us to obtain a trade-off between throughput and delay: smaller thresholds lead to higher homogeneity among the delay performance of the receivers but also to smaller throughput values, and vice-versa. In the case of heterogeneous receivers, we have seen that the receiver with the best channel experiences a decrease in the knowledge front, due to the fact that this receiver is the leading one almost every time slot and therefore receives more non-innovative packets due to recovery transmissions. In the case of homogeneous receivers, there is no node with such dominance over the others, which leaves the door open for improving the delay performance for all receivers. In fact, as exhibited in Figure 3.8(a) and Figure 3.8(b), using NC-ARQ with Hard Recovery can increase the knowledge front of all the receivers (at the cost of losses in throughput). As before, increasing the threshold results in higher throughput and smaller knowledge front.

3.5.2 NC-ARQ with Soft Recovery

With the Hard Recovery mechanism for NC-ARQ, we are able to control the delay experienced by the receivers by introducing uncoded transmissions whenever one of the receivers experiences a considerably larger number of erasures when compared to the leader. This delay control comes at the price of losses in throughput, given that the uncoded transmission only provides a new degree of freedom to the worst receiver. In the following, we present an algorithm that, during the recovery phase, ignores the leaders and uses the coding scheme of NC-ARQ. This means that each transmitted packet is innovative for all the receivers except for the leaders.

Algorithm 7 NC-ARQ with Soft Recovery

```
Let $S_i(t)$ be the number of successful transmissions received by receiver $i$ up to time slot $t$. In each time slot $t$, if $\max_{i,j} (S_i(t) - S_j(t)) > D$, $D \in \mathbb{N}$, ignore the feedback of the leaders; otherwise proceed with standard NC-ARQ.
```

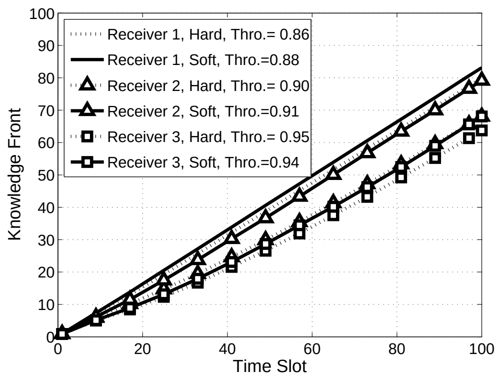
For $D = 1$, hard and soft recovery present the same performance, given that in this case there is no “middle” receiver, i.e. every receiver is either a leader or a poor receiver, and therefore the algorithms transmit the same packet in the recovery stage. In Figure 3.9(a) and Figure 3.9(b) we present a comparison of the two mechanisms in the case of heterogeneous receivers, with $D = 2$ and $D = 4$, respectively. The obtained throughput is higher for the soft recovery mechanism (except for the worst receiver, which lost throughput), as is the knowledge front for all the receivers. It is interesting to notice that, for both threshold values, the receiver that gains the most with the soft recovery mechanism is the one in the middle. This is due to the fact that the algorithm ignores the leader during a recovery phase. Consequently, coding decisions are made under the assumption that the middle receiver is the actual leader.

In the case of homogeneous receivers, the results obtained are similar for both recovery mechanisms, mainly due to the fact that we are considering the case of three receivers. It is reasonable to expect the differences between the two mechanisms to increase with the number of receivers, because with a larger number of receivers, more receivers will be neither leaders nor worst-case receivers.

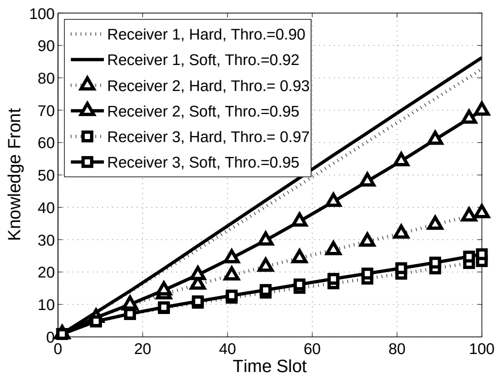
3.5.3 Tightness of the Random Walk Bounds

The random walk methodology for analyzing delay assumes that only leaders can decode new packets. This is not necessarily the case, which explains why the results obtained through the random walk correspond to worst case approximations of the decoding delay and lower bounds on the knowledge front. To verify the tightness of these bounds, we generate 100 independent random instances of the channels of the receivers for 20 time slots[3]. For each of these instances, we compute the two metrics of interest by using the random walk method, computing each packet that is transmitted in each time slot and performing the necessary decoding operations.

[3]Due to the heavy computational cost of simulating the entire system, we were forced to restrict our analysis to a small number of time slots.



(a) $D = 2$



(b) $D = 4$

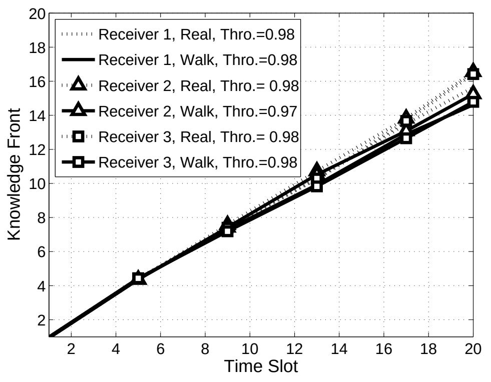
Figure 3.9: Comparison of the knowledge front for the scenario of heterogeneous receivers ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$), for the two different recovery mechanisms, with $D = 2$ and $D = 4$.

We tested several scenarios, namely all the combinations of the following set parameters: erasure probabilities $(0.05, 0.05, 0.05)$, $(0.05, 0.05, 0.1)$, $(0.05, 0.1, 0.1)$ and $(0.05, 0.1, 0.15)$; no recovery mechanism, hard recovery and soft recovery, with $D = 1, 2, 4$. In the majority of cases, the results obtained with the two different approaches were indistinguishable, with special emphasis on the case of standard NC-ARQ. For $D = 1$, our bounds match the actual value since no coded packets are transmitted and all the packets are transmitted in order. For $D = 4$, recovery transmissions are rarer, which leads to results similar to the unconstrained case. There are, however, two cases were the difference is more expressive, both of them for $D = 2$.

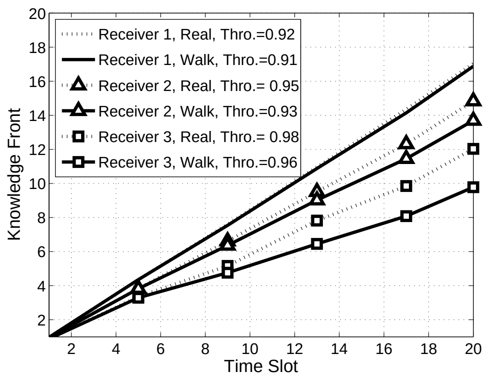
In Figure 3.10(a), we present the results obtained for NC-ARQ with Hard Recovery and $D = 2$ in the case of homogeneous receivers. The results show that the random walk provides a pessimistic result in the sense that there are time slots at which non-leaders decode a new packet. This happens more frequently in the case $D = 2$ because this is the case with more recovery stages (and also with coded transmissions), which means that in some time slots a non-leader receives an uncoded packet that was not yet decoded.

We can observe a similar behavior in NC-ARQ with Soft Recovery with $D = 2$, also in the heterogeneous receiver scenario, as exhibited in Figure 3.10(b). Among the cases we tested, this is the one that presented the largest differences.

As aforementioned, in all the other cases the differences are negligible. Thus, these results give evidence that the probability of a non-leader decoding a new packet is



(a) Hard Recovery, $D = 2$



(b) Soft Recovery, $D = 2$

Figure 3.10: Comparison of the results obtained using the random walk methodology and the ones obtained by simulating the entire system for NC-ARQ with Hard Recovery and NC-ARQ with Soft Recovery, in the homogeneous ($\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.05$) and heterogeneous ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.1$ and $\epsilon_3 = 0.15$) receivers scenarios, respectively, for $D = 2$.

indeed comparatively small. Therefore, we may conclude that the proposed random walk methodology provides tight bounds on the delay behavior of NC-ARQ, with and without recovery mechanisms.

3.6 Concluding Remarks

Network coding schemes are known to provide throughput optimal solutions for the wireless broadcast scenario, in particular in the presence of packet erasures. However, throughput optimality often comes at the cost of large delays for individual packets, which makes these solutions inadequate for applications with stringent delay constrains. This was shown to be the case for a state-of-the-art online network coding algorithm [SSM08], which is throughput optimal and offers optimum queue size yet leads to heavy delays when the communication channels have different erasure probabilities.

The analysis is non trivial, because a system description from the point of view of the packets arriving at each receiver leads to a state space that grows exponentially in the number of receivers. To overcome this computational hurdle, we proposed a new tool to analyze decoding delay. It is based on a random walk model that represents the number of erasures experienced by each receiver. The proposed approach was shown to reduce significantly the complexity of the analysis. Our results confirm the tightness of the bounds on decoding delay that can be obtained through the random walk model.

In a second step, we used the random walk model to design coding algorithms that aim to reduce the average delay experienced by the receivers, while ensuring fairness in the delay of receivers with different channel conditions. The encoding rules were thus redefined to transmit uncoded packets at crucial time slots, allowing receivers with worse channel conditions to recover from their backlog of undecoded packets. Since the transmitted packets are not necessarily innovative for all receivers, there exists a trade-off between throughput and delay.

Chapter 4

Deadline Constrained Applications

4.1 Motivation

Broadcasting real-time data streams to multiple terminals over wireless networks in a timely and reliable manner is a challenging problem due to the heterogeneity of packet losses for different receivers and consequent retransmission of packets. In the IEEE 802.11 standard, reliable data delivery is implemented for unicast connections by means of positive acknowledgments (ACK) after each transmitted packet. If an ACK does not arrive within a certain time interval, the sender considers that the packet was lost by the receiver and initiates its retransmission. When data needs to be transmitted simultaneously to several receivers in a multicast fashion, satisfying individual requests to each receiver becomes inefficient in terms of energy and prohibitive in terms of delay. The reason is that while a packet is retransmitted to a specific receiver, the other receivers are idle, particularly those who do not need that particular packet to be retransmitted. The solution for this problem is to send coded retransmissions that are maximally useful for all receivers. However, as we have seen in the previous chapters, these coding decisions must be carefully designed, to ensure timely decodability, crucial for real-time applications.

Moreover, the use of ACKs to ensure reliable data delivery in the case of multicast with feedback is not straightforward, since the sender must wait for the ACKs of each receiver before transmitting any data. This implies significantly larger delays in data delivery and requires receiver coordination. Thus, multicast data is transmitted in open-loop fashion in the IEEE 802.11 standard, i.e. without any feedback mechanism.

In this chapter, we present *FEBER*, an online coding scheme based on the following

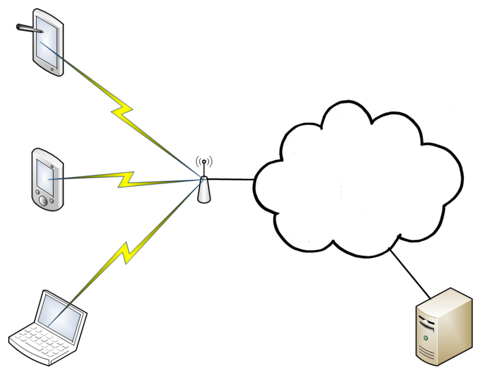


Figure 4.1: System setup. Multiple clients wish to receive real-time data, generated in a remote source, through an access point. Our goal is to design a communication scheme that allows for timely delivery of data to the receivers, while ensuring that the access point makes an efficient use of the available resources.

key idea: *If the source has partial feedback information about the packets lost at each receiver, then reliable real-time multicast can be achieved efficiently using coded retransmissions.* For the scenario depicted in Fig. 4.1, FEBER presents the following properties:

- *Timely Delivery of Real-Time Data:* In the case where data packets have strict delivery deadlines, FEBER favors the delivery of critical packets, while using free transmission opportunities to recover from previous packet losses.
- *Efficient Use of Channel Bandwidth:* FEBER allows the sender to code across data packets to construct the recovery transmissions, making it possible to simultaneously satisfy different requests from different receivers.
- *Trade-Off Between Reliability and Bandwidth Usage:* Our proposal includes tunable parameters that allow the system designer to choose the amount of efficiency he is willing to sacrifice in order to further increase the ratio of packets delivered on time.
- *Sporadic Feedback:* In contrast with previous solutions, where feedback information was necessary after each transmission, our retransmission scheme requires only sporadic feedback information.
- *Compatibility with current standards:* In order to evaluate the performance of our solution, we were able to implement FEBER on top of the IEEE 802.11 standard, which asserts the compatibility of our proposal with current wireless networking standards.

The remainder of this chapter is organized as follows. In Section 4.2, we present an overview of the related work. Section 4.3 describes the system architecture and Section 4.4 sets the notion of critical packet. We then construct our coding scheme in Section 4.5, that requires the feedback mechanism described in Section 4.6. Section 4.7 provides a description of the major challenges, as well as the corresponding solutions, when implementing our proposal in a IEEE 802.11 wireless network testbed. We then use the testbed to evaluate the performance of FEBER, in Section 4.8. Finally, Section 4.9 offers some concluding remarks.

4.2 Related Work

Several previous works have attempted to achieve reliability in IEEE 802.11 multicast connections by means of medium access techniques [TG00, SHW$^+$03] that, either using location information or randomly, attempt to coordinate the receivers to provide feedback information to the sender. This information is then used to individually retransmit lost packets.

If we empower the sender with the ability to code across packets, then Fountain Codes [BLMR98] present a solution to multicast over packet erasure channels. Fountain Codes can be extended to cope with real-time requirements. This is the case of [VSS$^+$09], which uses unequal error protection in a block-based coding scheme to increase the probability of delivering the most important video frames. In [TV10], a sliding coding window is used, with the sender transmitting the oldest packet in the window and then a recovery coded transmission, before moving the coding window forward.

Feedback information can be used to design coding solutions that allow receivers to progressively decode the information. In multi-hop wireless networks, the feedback information can be obtained by overhearing transmissions from the neighboring nodes. In [KRH$^+$08], the authors presented COPE, where intermediate nodes combine packets from different flows and hence can simultaneously satisfy the requests of several receivers. For the particular case of multiple video streams, [SM09] proposes an opportunistic network coding scheme, where intermediate nodes take into account video quality metrics in order to decide which flows to combine in a single transmission.

For broadcast packet erasure channels, the typical approach is to consider that an error free feedback channel is available, in a full duplex setting. Jolfaei *et al* in [JMM93] propose a coding scheme based on a selective repeat strategy, and [NTNB09] proposes a simpler way to choose which packets to combine, by selecting a maximum set of lost packets from different receivers. Using larger field sizes for the linear combinations of data packets, [Lar08] proposes a method that adaptively selects the weights of such linear combinations, which leads to optimal throughput.

To cope with applications with stringent end-to-end delay requirements, [ZX10] focus on the scenario where each packet has a deadline, after which it becomes useless to the receivers. The authors prove that the problem of minimizing the number of packets that miss their deadlines is NP-hard, and then propose a coding algorithm based on the maximum-weight clique problem.

The link layer solutions for reliable multicast [TG00, SHW$^+$03] incur in large delays in data delivery and, moreover, do not fully exploit the broadcast nature of the wireless medium, since each transmission provides only one distinct packet. In contrast, we enable the sender to code across source packets to simultaneously satisfy different requests from different receivers.

Standard fountain coding solutions fail to adapt to real-time requirements, since receivers need to wait until several encoded packets are received to be able to decode any part of the original data. In contrast, FEBER, transmits each source packet uncoded at least once. In contrast with [VSS$^+$09], we provide an online coding solution which is not dependent on the video encoding. [TV10] allows at most one recovery packet between the generation of two source packets, while FEBER has no such restriction. Moreover, this scheme requires a large number of transmissions in order to satisfy the real-time requirements.

Both COPE [KRH$^+$08] and [SM09] are designed for inter-session network coding, by combining different flows, whereas we consider the problem of satisfying different receivers requests regarding the same flow. [KRH$^+$08] and [SM09] use a FIFO management for the requests of each receiver and compute a packet that must be decoded by the next hop of the head of the sender's output queue. In our proposal, there is no such restriction and all the requests from each receiver are taken into consideration, which significantly changes the nature and the complexity of the coding problem.

In the online coding solutions for the broadcast packet erasure channel presented in [JMM93, NTNB09, Lar08, BCMW09], nodes still need to wait for a considerably large amount of data before being able to decode some of the information, which is not suited to real-time data. The heuristic proposed in [ZX10] is based on the instant access of the sender to feedback information from the receivers, which drastically limits the implementation of such solution. We construct a different heuristic that handles with outdated feedback information.

4.3 System Architecture

4.3.1 Network Model

The sender communicates with N receivers over a broadcast packet erasure channel channel (e.g. a IEEE 802.11 multicast connection). More precisely, each transmitted packet is correctly received at receiver i with probability $1 - \epsilon_i$, for $i = 1, \ldots, N$. We assume that the communication medium is half-duplex, which means that the sender and the receivers can transmit and receive, but not both simultaneously.

4.3.2 Sender

The sender is composed of three parts, where the first is a source that generates the data packets, sequentially numbered. Given that we are interested in scenarios with stringent delay requirements, we assume that upon the generation of a packet, the source also defines the *deadline* of the packet, that represents the time instant after which the packet becomes useless to every receiver.

The packets generated by the source arrive at the second component of the sender, the buffer, and are kept there until they have been delivered to all receivers, or their deadline is broken. Packets in the buffer are immediately available for the third component of the sender, the encoder, that computes the packets to be transmitted.

The sender keeps a list of which packets were decoded by each receiver, updated using estimation as well as feedback information from the receivers, as we describe in Section 4.6.2. For each receiver R_i and each source packet p_j, the sender also stores T_j^i, that denotes the number of linear combinations transmitted since the last feedback from R_i, that allow R_i to decode p_j if no erasure has occurred.

4.3.3 Receivers

Given that packets are numbered sequentially, receivers can detect if a source packet is missing. Each receiver is able to inform the sender of its reception status (i.e. which packets it has or has not received) by sending feedback packets. To enable the decoding of received linear combinations, the receivers store each decoded source packet (while the corresponding deadline is not reached) in a decoding buffer. Notice that, since the transmitted packets are linear combinations of source packets, we have that with respect to a given receiver, each received packet can be *innovative*, if it is linearly independent with all the coded packets available at the receiver, *immediately decodable*, if it is innovative and the receiver can decode a new source packet, or *discardable*, if it is not innovative.

4.4 Time-Critical Packets

Our retransmission scheme is based on the notion of *critical packet*. A packet becomes critical when the probability of successful delivery of the packet to all receivers within a time frame is below a given threshold, in the case where the sender transmits only linear combinations that allow the receivers to decode this packet. *Definition 5* provides a formal definition for this concept.

Definition 5 *Let $Z^i(t)$ denote the event of a packet being successfully received in t transmissions by receiver R_i. Let T_j^i denote the number of linear combinations transmitted, since the last feedback received from R_i, that allow this receiver to decode p_j if no erasure has occurred. We say that a source packet p_j is critical if*

$$P\left(\bigcap_i Z^i\left(k_j + T_j^i\right)\right) \leq \alpha, \tag{4.1}$$

for some $\alpha \in]0,1[$, where k_j is the number of transmissions available until the deadline of the packet p_j.

It is worth clarifying the role of T_j^i in (4.1). Since the last feedback received from R_i, the sender has transmitted T_j^i linear combinations that, if correctly received by R_i, allow this receiver to decode packet p_j. The sender is not yet aware of the success of these T_j^i transmissions. Thus, if there are k_j transmission opportunities available until the deadline of p_j, from the point of view of the sender, there are $k_j + T_j^i$ transmissions with the potential to deliver p_j to R_i.

4.5 Coded Retransmissions

4.5.1 FEBER

The notion of critical packet plays a leading role in our retransmission scheme. FEBER prioritizes the delivery of critical packets and uses available transmission opportunities to recover from packet losses, before the packets become critical.

Algorithm 8 FEedback Based Erasure Recovery (FEBER)

```
1: if there is a critical packet in the buffer then
2:     Run Critical Recovery Algorithm
3: else
4:     if a new packet arrived from source then
5:         Transmit the new packet
6:     else
7:         Run Recovery Algorithm
8:     end if
9: end if
```

If there exists a critical packet in the buffer, FEBER runs the *Critical Recovery Algorithm*, where the encoder constructs a coded packet that provides the critical packet to receivers that still miss it and, if possible, other source packets to receivers that have already decoded the critical packet.

If there is no critical packet, then the sender checks if there are packets in the buffer that were never transmitted before and, in the affirmative case, it transmits the older of such packets. The goal with this stage is to avoid retaining the transmission of recently generated packets to recover from older packet losses, which would penalize the receivers that have not lost those packets. Moreover, such transmission is throughput optimal, since it is useful for every receiver. If there exists no critical packet and no new source packet, FEBER runs the *Recovery Algorithm*, where the goal is to help a large set of receivers to recover from previous packet losses.

To describe the two coding algorithms used in *Algorithm 8*, we need first to set some notation. The critical packet is denoted by p_C. In case multiple critical packets exist, p_C is the packet with the highest probability of having the deadline broken, i.e. p_C is the packet that maximizes the left hand side of (4.1).

Let B denote the set of packets that are in the buffer and thus available for the encoder. We denote the set of receivers that have not decoded packet p by $U(p)$, and $A(i)$ denotes the set of packets already decoded by receiver i (and that are in B). Finally, if $C = \{p_1, \ldots, p_n\}$ is a subset of B, we denote by $idec(C)$ the set of receivers that are able to immediately decode a new source packet from the linear combination $p_1 \oplus \cdots \oplus p_n$. $|idec(C)|$ denotes the number of such receivers. The following coding algorithms are modified versions of the Greedy algorithm described in *Algorithm* **??**, in Chapter 2, to cope with the specific requirements of data with strict deadlines.

4.5.2 Recovery of Lost Packets

When neither a new packet nor a critical packet are present at the buffer, FEBER runs the *Recovery Algorithm* (*Algorithm 9*) to construct a linear combination that provides immediately decodable packets to a large set of receivers in a single transmission. The goal is to construct a set of source packets C that maximizes $|idec(C)|$, over all possible sets $C \subset B$. However, as discussed in Chapter 2, if n is the number of packets in B, we need to search among $2^n - 1$ possible subsets of B to find the solution, which is a NP-Hard problem [ERCS07, ZX10]. Hence, we use an heuristic to solve such problem.

Algorithm 9 Recovery Algorithm

```
1: $C \leftarrow \{\}$
2: $q \leftarrow 0$
3: $p^* \leftarrow \arg\max_{p \in B} |idec(\{p\})|$
4: while $|idec(C \cup \{p^*\})| > q$ do
5:     $C \leftarrow C \cup \{p^*\}$
6:     $q \leftarrow |idec(C \cup \{p^*\})|$
7:     $p^* \leftarrow \arg\max_{p \in B} |idec(C \cup \{p\})|$
8: end while
9: if $|idec(C)| \geq M$ then
10:     transmit $c = \bigoplus_{p_i \in C} p_i$
11: else
12:     Do not transmit
13: end if
```

The set C represents the set of source packets that will be XORed together to construct the packet to be transmitted. The Recovery algorithm starts by selecting the source packet in the buffer that maximizes the number of receivers that have not decoded it. This corresponds to maximizing $|idec(\{p\})|$ over all source packets p. After this first choice, the algorithm evolves by adding to C the source packet that, when added with the previously chosen packets, maximizes the number of receivers that immediately decode such linear combination ($p^* \leftarrow \arg\max_{p \in B} |idec(C \cup \{p\})|$), if it increases the number of such receivers when compared with the previous choice ($|idec(C \cup \{p^*\})| > q$). Finally, to control the amount of recovery packets that are transmitted, the sender only transmits the output packet if this coded packet is immediately decodable for at least M receivers, where M is a tunable parameter.

4.5.3 Recovery of Critical Packets

Algorithm 10 Critical Recovery Algorithm

```
 1: $C \leftarrow \{\}$
 2: $q \leftarrow 0$
 3: $p^* \leftarrow p_C$
 4: $A_C = \bigcap_{i \in U(p_C)} A(i)$
 5: while $|idec(C \cup \{p^*\})| > q$ do
 6:     $C \leftarrow C \cup \{p^*\}$
 7:     $q \leftarrow |idec(C \cup \{p^*\})|$
 8:     $p^* \leftarrow \arg\max_{p \in A_C} |idec(C \cup \{p\})|$
 9: end while
10: transmit $c = \bigoplus_{p_i \in C} p_i$
```

If a critical packet is detected, the encoder runs the *Critical Recovery Algorithm* (*Algorithm 10*), that must ensure that the receivers that lost the critical packet decode it, if no erasure occurs. Here, we follow a strategy similar to *Algorithm 9*, with some minor yet crucial differences. The first source packet added to the linear combination is the critical packet itself ($p^* \leftarrow p_C$). Given that receivers that lost the critical packet must decode it from the constructed linear combination, we are now limited to choose source packets that have been decoded by all such receivers, i.e. source packets in the set $A_C = \bigcap_{i \in U(p_C)} A(i)$. Within this set, the algorithm then evolves as *Algorithm 9* to construct the linear combination. The constructed linear combination is transmitted irrespective of the number of receivers that can immediately decode it, in contrast with *Algorithm 9*. In this stage, the goal is to ensure that the critical packet deadline is not broken at some receiver.

4.6 Multi-User Feedback

Due to the half-duplex constraint on the communication medium, transmitting feedback information from the receivers to the source is challenging. In this section, we discuss how we tackle this problem.

4.6.1 Medium Access and Scheduling

The receivers, as well as the sender, are capable of transmitting and receiving data, but not both simultaneously. Therefore, in order to receive feedback information, the sender must stop transmitting data, which means that we have a delay increment with each feedback transmission (in IEEE 802.11 multicast connections, there is also a medium access contention problem, which we discuss in Section 4.7.3). Hence, we must refrain the use of the feedback channel. We set the receivers to send a feedback packet after F generated source packets since last feedback, where F is a tunable parameter.

Moreover, since the communication medium is shared, receivers must coordinate their feedback transmissions. To avoid collisions among the feedbacks of the different receivers, we desynchronize the transmission of feedback packets. For that, we set the first feedback packet to be transmitted by receiver R_i only after $F + \lambda_i$ packets generated by the source, where λ_i is a desynchronization parameter of receiver R_i. After this first packet, the following feedback packets are transmitted every F generated source packets.

4.6.2 Reception Status Estimation

Given that feedback information is not always available to the sender, when it transmits a packet that enables receiver R_i to decode packet p_j, the encoder is not immediately aware of the success of such transmission. Thus, the sender does not know whether R_i decoded p_j or not. We make use of channel statistics to update the knowledge of the encoder on the decoding state of the receivers, as follows.

Let $S_\beta(i)$ be the minimum number of transmissions to ensure that the probability of receiver R_i successfully receiving the transmission is at least β, i.e. $S_\beta(i) = \min\{k : 1-\epsilon_i^k \geq \beta\}$. Recall that T_j^i is defined as the number of linear combinations transmitted, since the last feedback received from R_i, that allow receiver R_i to decode p_j if no erasure has occurred. Finally, to overcome the problem caused by the absence of feedback, the source assumes that packet p_j, reported as missing by R_i in its last feedback, was decoded by R_i if and only if $T_j^i \geq S_\beta(i)$.

4.7 Integration with IEEE 802.11

In this section we discuss the system aspects of the implementation of FEBER over 802.11 networks. We provide further details of our implementation of FEBER in Appendix A.

4.7.1 Transmission Duration Estimation

The construction of FEBER is based on slotted time, by requiring the knowledge of the number of transmissions available until a certain event. Thus, in IEEE 802.11 wireless networks, we need to provide the encoder with an estimation of the duration of a transmission, which we compute as follows. Let Δ_T be the time elapsed from the delivery of a packet to the 802.11 MAC layer until the next announcement of available channel received from the MAC layer[1]. Let $\overline{\Delta}_T$ denote the empiric average of Δ_T. We can see $\overline{\Delta}_T$ as the average time necessary to transmit a packet, from the network layer perspective.

4.7.2 How to determine packet criticality

The notion of critical packet is central in the construction of FEBER. In *Definition 5*, we require that the encoder knows the number of transmissions available until the deadline of the packet p_j, denoted by k_j. Using the estimated transmission duration $\overline{\Delta}_T$, we estimate k_j by $k_j = \left\lfloor \Delta_D / \overline{\Delta}_T \right\rfloor$, where Δ_D denotes the time left until the deadline of the packet p_j.

The next challenge is how to estimate the probability of a packet being lost, for each receiver. The source can empirically compute such probability from the feedback packets, in which receivers announce which source packets they have lost. Let ϵ_i denote such erasure probability for receiver i.

[1] In our implementation, we make use of Click toolkit [KMC+00] to announce a transmission opportunity to the encoder.

In order to compute (4.1) from *Definition 5*, we require some simplifying assumptions about the statistical characterization of the wireless medium. We assume that erasures occur independently across receivers and time. Therefore, due to independence across receivers, we have that $P\left(\bigcap_i Z^i\left(k+T_j^i\right)\right) = \prod_i P\left(Z^i\left(k+T_j^i\right)\right)$. Moreover, since erasures occur independently across time, $P\left(Z^i\left(k+T_j^i\right)\right) = 1-\epsilon_i^{k+T_j^i}$. Therefore, we may conclude that $P\left(\bigcap_i Z^i\left(k+T_j^i\right)\right) = \prod_{i\in\mathcal{I}}\left(1-\epsilon_i^{k+T_j^i}\right)$, where $\mathcal{I}$ is the set of receivers that have not acknowledged the decoding of packet p_j.

4.7.3 Feedback

In order to exploit the broadcast nature of the wireless medium, we set the sender to use a IEEE 802.11 multicast connection to transmit the data, which does not provide a feedback connection for receivers to acknowledge received packets. To announce lost packets, we enable a IEEE 802.11 unicast connection between each receiver and the sender.

The combination of multicast flows with unicast flows may cause a network collapse, due to the unfairness between multicast and unicast flows regarding the medium access. Multicast flows do not adjust the contention to access the wireless channel according to the network load, as shown in [DT06]. To avoid the possibility of such collapse and to prevent collisions between feedback transmissions, we use the feedback scheme discussed in Section 5.4. More precisely, receivers send feedback every F generated source packets, where for each receiver R_i, the first feedback is sent after $F + \lambda_i$ generated source packets.

Notice that the receivers do not know exactly when the source generates a packet. However, upon the reception of a source packet, receivers check the source packet sequence number, given that source packets are numbered sequentially, thus becoming aware of how many source packets were generated since the last feedback packet transmitted. If this number of generated packets is at least F, the receiver transmits a feedback packet to the sender.

4.8 Experimental Evaluation

In this section, we discuss the performance of FEBER through the results obtained from a IEEE 802.11 wireless network testbed. Our experiments reveal the following features of our proposal.

- FEBER achieves a low ratio of packets that miss the corresponding deadline, with an efficient bandwidth usage. For packet erasure ratios up to 10%, the number of packets that miss the deadline is negligible (Fig. 4.2), with practically no overhead over throughput optimal coding schemes (Fig. 4.4).

- The parameter M provides a trade-off between the number of broken deadlines and the amount of bandwidth we are willing to sacrifice. The larger the value of M, the more the sender refrains from transmitting recovery packets, by only transmitting packets that are useful to a larger set of receivers (Fig. 4.3), which comes at a cost of a larger ratio of missed deadlines (Fig. 4.2).

- The frequency of feedback transmissions has a significant impact on the number of deadlines broken. Larger values of F lead to a significantly larger ratio of missed deadlines (Fig. 4.5). However, F cannot be set too small, otherwise the network will collapse, as discussed in Section 4.7.3.

- The maximum delay allowed in packet delivery has a significant impact in the number of broken deadlines. Nevertheless, FEBER achieves a ratio of broken deadlines of 7×10^{-5} with a maximum allowed delay of 0.4 seconds between packet generation at the source and the corresponding delivery at the receiver (Fig. 4.8).

- FEBER is significantly better than a multiple unicast solution. In contrast with the ratio of $1,4 \times 10^{-6}$ achieved by FEBER, the multiple unicast approach presented a ratio of broken deadlines of $7,9 \times 10^{-4}$ (*Table 4.1*).

4.8.1 Testbed

Following the guidelines in Section 4.7, we implemented FEBER in a IEEE 802.11 testbed, composed by one access point and five receivers. The access point transmits a video encoded with MPEG-2 codec at a constant bit rate of 1375kbps and a total of 148 seconds of duration. We use the VLC Media Player (version 1.1.4) to handle the streaming of the video to the receivers, though an UDP connection. The VLC streamer divides the video in 21040 packets, each with 1344 bytes of size at the IP layer, including IP header. After being generated by VLC, we add a deadline to each packet. Given that we consider a video stream, we can view the gap between packet generation and the corresponding deadline as the maximum allowed buffering delay at the receiver, before playback starts. Thus, in order for a packet to be played at a receiver, it needs to be received before its deadline. Otherwise, the packet will be considered as lost and the receiver will skip it, which may cause VLC to produce artifacts in the image displayed. We provide further details on our testbed in Appendix A.

The experimentally observed packet erasure ratios were all close to zero, because it is challenging to obtain significant changes in the received signal-to-noise ratio that translate into packet erasures only from simply moving the nodes around [HLLS04]. To achieve a wider range of packet erasure ratios, we induce artificial erasures by dropping extra packets at the receivers, with 10% to 40% of extra randomly erased transmissions that are added to the normal channel erasures. We clustered the data into classes of packet erasure ratios. For each of these classes and for each experiment, we run 35 independent trials and we plot the results with 95% confidence intervals.

4.8.2 Evaluation Metrics

Our main goal is to provide a reliable multicast connection, in the sense that packets are delivered to the receivers before their deadline expires. Hence, the first evaluation metric to be considered is the ratio of missed deadlines at each receiver, defined as follows.

Definition 6 *Let D_i denote the number of deadlines missed at R_i and let S denote the number of source packets. Then, we define*

$$\textit{Ratio of Missed Deadlines for } R_i = \frac{D_i}{S}.$$

In our testbed, we have $S = 21040$ video packets.

The reliability achieved by FEBER is obtained through the use of recovery retransmissions, which lead to extra bandwidth consumption. Due to the broadcast nature of the wireless channel, each transmission reaches all the receivers, which may deem some transmissions useless for some of the receivers. To measure the efficiency of the transmissions with respect to a receiver, we look at the ratio of received packets that are innovative to the receiver, as follows.

Definition 7 *Let $Inov_i$ denote the total number of bytes of packets received by R_i that were innovative and let Rec_i denote the total number bytes of received packets. We define*

$$\textit{Transmission Efficiency for } R_i = \frac{Inov_i}{Rec_i}.$$

To evaluate the overhead in the amount of transmitted data experienced by FEBER, we compare our proposal against a throughput optimal coding scheme, i.e. coding that ensures that every transmission is useful to every receiver and, hence, each receiver can decode 21040 video packets upon the correct reception of 21040 transmissions. Examples of such schemes include MDS and Fountain codes, as well as random coding in a sufficiently large finite field.

We did not implement any of such schemes in our testbed, since they do not adapt directly to the deadline constrained case. Thus, we do not know the corresponding packet size. We assume that the packets in such schemes do not have any overhead and, thus, their size is the same as the uncoded packets, i.e. 1344 bytes. In such schemes, the sender would stop transmitting only upon the correct reception of 21040 by all the receivers. Hence, the number of transmissions is governed by the last receiver to reach 21040 successful receptions. Therefore, we measure the overhead in the amount of transmitted data as follows.

Definition 8 *Let T denote the number of bytes transmitted by the sender and let S denote the number of source packets, each of size b bytes. Let $Trans_i$ denote the number of sender transmissions until R_i successfully received S of such transmissions. Then, we define*

$$\textit{Transmission Overhead} = \frac{T - b \cdot \max_i Trans_i}{b \cdot \max_i Trans_i}.$$

In our testbed, we have $S = 21040$ and $b = 1344$.

Notice that in such ideal schemes, the deadlines of the packets are not taken into account. The minimum amount of transmitted data in the deadline constrained case is still an open question, to the best of our knowledge.

4.8.3 Impact of the Choice of M

Experiment: Recall that, in FEBER, a recovery transmission only occurs if the number of receivers that would find such transmission useful is at least M. The role we envision

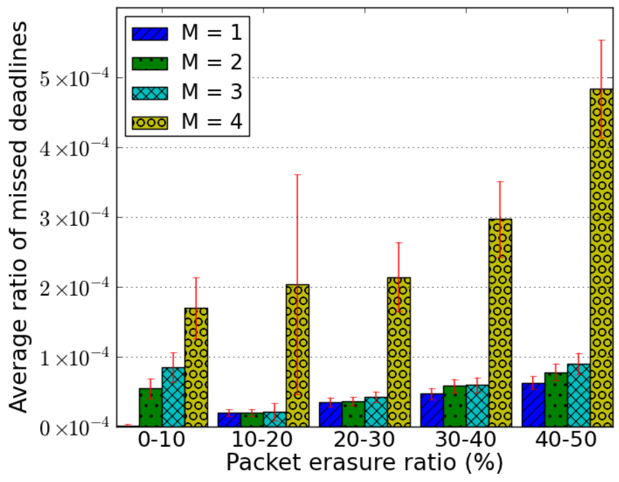


Figure 4.2: Average ratio of packets that missed their deadline, for different values of M. FEBER achieves less than 2 packets with broken deadlines, out of the 21040 packets, for packet erasure ratios up to 50%.

for M is to provide a trade-off between the number of packets that reach the receivers on time and the bandwidth usage necessary to provide them. To evaluate the impact of M and of coding across packets, we set the deadline of each packet to be 1 second after its arrival at the sender's buffer, $\alpha = \beta = 95\%$, $F = 15$, and we analyze the performance obtained for $M = 1, \ldots, 4$, as a function of the packet erasure ratio.

Results: The results in Fig. 4.2 show that FEBER can achieve a low ratio of packets that miss their deadline, even for high packet erasure ratios. For $M \leq 3$ and for packet erasure ratios from 10% to 40%, our scheme achieves a ratio of roughly 0.5×10^{-4} of the source packets with broken deadlines, which translates to less than a single packet out of the 21040. Still for $M \leq 3$, for the entire range of packet erasure ratios considered, FEBER is capable of providing less than two packets with broken deadline.

Refraining the source from transmitting at every opportunity by choosing a larger M leads to a larger ratio of missed deadlines. The choice $M = 4$ has a considerable impact on the achieved ratio of missed deadlines.

From 0% up to 10% of packet erasure ratio, for $M = 2$ and 3, FEBER achieves higher ratio of missed packets when compared against the ratio of missed packets for the same values of M with packet erasure ratio from 10% up to 40%. This behavior is due to the lack of diversity of packets lost in the receivers, which limits the use of recovery transmissions when $M > 1$.

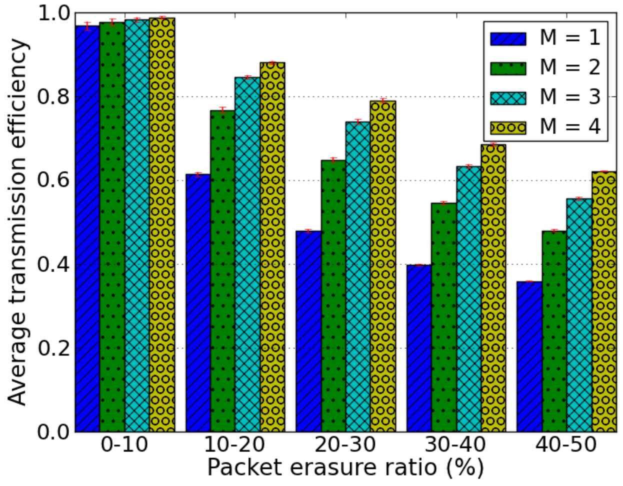


Figure 4.3: Average transmission efficiency, for different values of M. Refraining the use of bandwidth by increasing the value of M leads to more efficient transmissions, at the cost of an higher ratio of missed deadlines, as we can observe in Fig. 4.2.

Regarding the transmission efficiency (*Definition 7*), FEBER provides with high efficiency values, even with high packet erasure ratios, as we can see in Fig. 4.3. Clearly, larger values of M lead to an increase in efficiency, specially for packet erasure ratios above 10%. However, as we observed in Fig. 4.2, this increase in efficiency comes at the cost of higher ratio of missed deadlines. Therefore, the parameter M provides a trade-off between reliability and bandwidth usage.

The packet erasure ratio has a significant impact on the transmission efficiency, with the increase of this ratio leading to a decrease in the efficiency, due to the high diversity among the requests of the receivers. Nevertheless, even in the worst regime of packet erasure ratios considered (from 40% up to 50%), the choice $M = 3$ leads to less than two packets with broken deadlines out of the 21040, with a transmission efficiency above 50%, which means that the receivers found more than half of the packets useful, on average.

With respect to transmission overhead (*Definition 8*), Fig. 4.4 exhibits the obtained results. As in the case of transmission efficiency, in larger packet erasure ratios regimes, we experience larger transmission overheads, due to the diversity in the missing packets among the receivers. With respect to the impact of M, increasing the transmission efficiency by increasing M naturally leads to a smaller overhead.

In the worst packet erasure ratio regime considered, with $M = 2$, FEBER uses twice as much bandwidth as throughput optimal coding schemes to achieve less than two

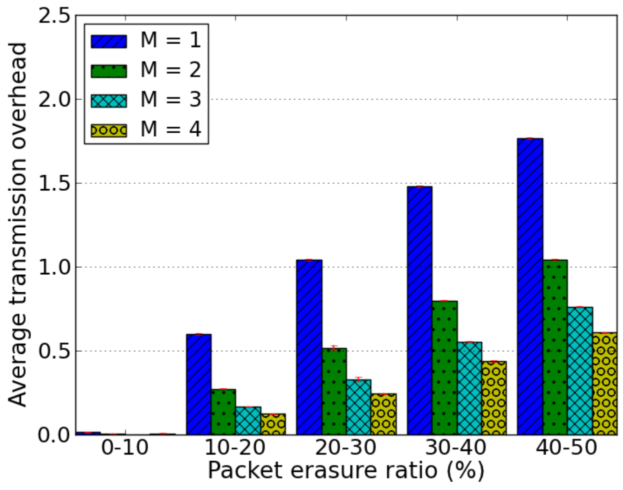


Figure 4.4: Average transmission overhead, for different values of M. Higher values of M lead to smaller overhead, at the cost of more deadlines missed, as we can see in Fig. 4.2.

packets with broken deadlines, out of the 21040. Recall that these throughput optimal coding schemes do not comply with the delivery of the packets before their deadline is broken.

4.8.4 Influence of F

Experiment: FEBER relies on a periodic feedback mechanism to estimate the reception status of the receivers, which is then used to perform coding decisions suited to the individual needs of each receiver. To evaluate the impact of the frequency of the feedback transmissions, we set $M = 2$, the deadline of each packet to be 0.4 seconds[2] and $\alpha = \beta = 90\%$, and we analyze the performance obtained for $F = 15, 30, 45, 60, 75, 90$, as a function of the packet erasure ratio.

Results: In Fig. 4.5, we can observe that FEBER is highly influenced by the value of F. Since feedback transmissions are more frequent, smaller values of F present a significantly smaller ratio of broken deadlines.

The results obtained for the transmission efficiency were similar for all values of F

[2]The maximum deadline allowed is set to be 0.4 seconds to allow a range of experiments containing cases that goes from very frequent feedback transmissions, with $F = 15$, which corresponds to an average of 4 feedback opportunities for each packet before its deadline, to very scarce feedback information, with $F = 90$, which corresponds to no opportunity to announce losses, on average.

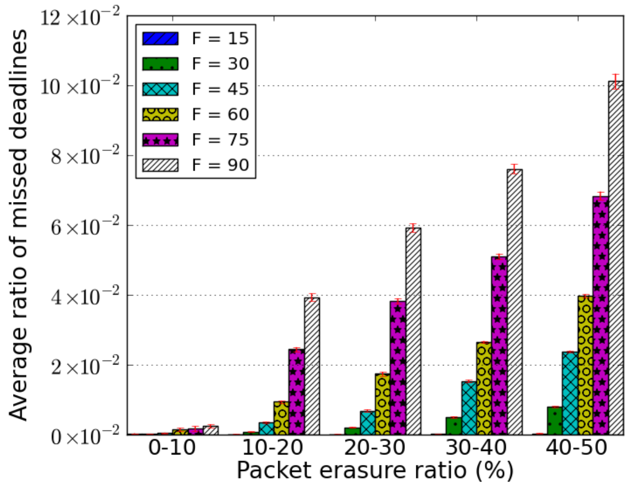


Figure 4.5: Average ratio of packets that missed their deadline, for different values of F. Highly frequent feedback information reduces significantly the number of broken deadlines. However, recall that if F is to small, it can cause the network to collapse, as discussed in Section 4.7.3.

and, hence, we do not present the corresponding plot. In contrast, the impact of F on the transmission overhead is considerable, as we can observe in Figure 4.6. Larger values of F lead to smaller overhead. However, as depicted in Fig. 4.5, this comes at a cost of a considerably higher ratio of broken deadlines. Nevertheless, for the worst packet erasure ratio regime considered, for all considered values of F, the achieved transmission overhead over throughput optimal coding schemes is between 0.5 and 0.9.

4.8.5 Influence of the maximum delay allowed

Experiment: The time elapsed from packet generation to the corresponding deadline is a fundamental parameter in real-time system. Smaller values of this maximum allowed delay diminish the time available for the sender to recover missing packets. To evaluate how does the maximum delay allowed influences the performance of FEBER, we set $M = 2$, $\alpha = \beta = 95\%$ and $F = 15$, and we analyze the performance obtained for $0.1, 0.4, 1.0$ seconds of maximum delay allowed, as a function of the packet erasure ratio.

Results: The maximum delay allowed between packet generation and the corresponding delivery to the application at the receiver has a significant impact in the number

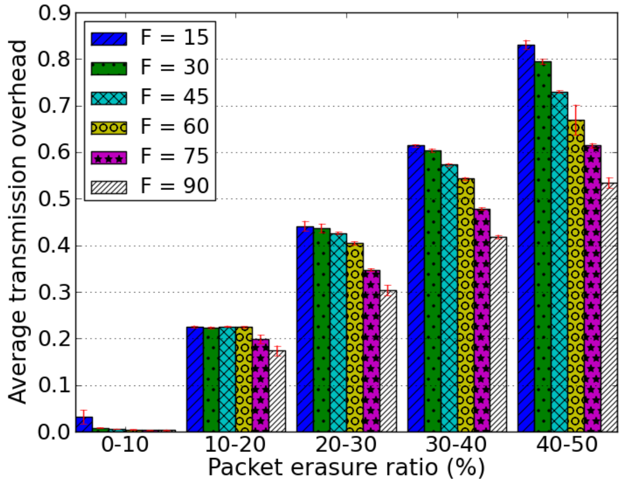


Figure 4.6: Average transmission overhead, for different values of F. Higher values of F lead to smaller overhead, at the cost of significantly more deadlines missed, as shown in Fig. 4.5.

of broken deadlines. For a maximum delay of 0.1 seconds, the obtained ratio of missed deadlines, Fig. 4.7, is significantly higher than the ratios obtained for the other maximum delays considered, Fig. 4.8. For instance, for packet erasure ratios between 10% and 20%, we obtain a ratio of 11×10^{-3} broken deadlines for 0.1 seconds of maximum delay, whereas for maximum delay of 0.4 and 1 seconds, the obtained ratio is 2×10^{-5}. Notice that, even for high packet erasure ratios, with a maximum delay of 0.4 seconds, we can obtain a ratio of missed deadlines below 15×10^{-5}, which corresponds to less than three packets with broken deadlines, out of the 21040.

Fig. 4.7 presents higher drop packet ratios when compared against Fig. 4.8 because for 0.1 seconds of maximum delay, with $F = 15$, receivers are not able to announce the loss of packets before the corresponding deadline is broken. With a video composed by 21040 packets and a duration of 148 seconds, for each 0.1 seconds of maximum delay, the source generates on average less than 15 source packets. Even if the receivers are able to report a loss of a packet on time, this will occur very close to the deadline of the packet, causing FEBER to immediately consider it as a critical packet. Therefore, for 0.1 seconds of maximum delay FEBER requires smaller values of F to achieve lower ratios of missing deadlines.

The results obtained for the transmission efficiency were similar for all maximum delay values and, thus, we do not present the corresponding plot. Regarding the transmission overhead, Fig. 4.9 shows that the increase of the maximum delay allowed leads to a

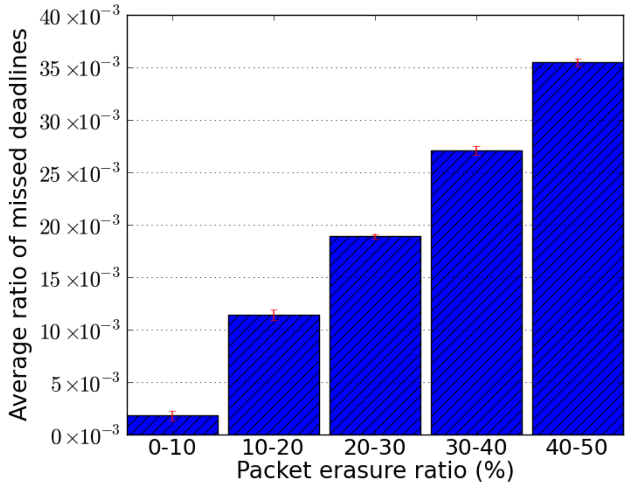


Figure 4.7: Average ratio of packets that missed their deadline, for 0.1 second of maximum delay. With $F = 15$, the receivers have very few opportunities to announce lost packets before the corresponding deadline, which induces an higher number of broken deadlines.

smaller transmission overhead. For packet erasure ratios from 40% up to 50%, for any maximum delay value, FEBER uses roughly 2 times the transmissions done by throughput optimal schemes to achieve less than three packets with broken deadlines, out of the 21040, for maximum delay of 0.4 and 1.0 seconds.

4.8.6 Benefits over multiple unicast

Experiment: The entire construction of our proposal is based on the use of a lossy multicast connection to exploit the broadcast nature of the wireless medium. The typical approach to obtain a reliable connection to multiple receivers over IEEE 802.11 networks is to use multiple unicast connections, where each receiver receives the data through an individual connection. This naturally leads to higher bandwidth consumption, due to the replication of transmitted data. However, it is necessary to evaluate the ratio of missed deadlines achieved with this approach. To evaluate how does FEBER compare against the multiple unicast approach, we developed the following experiment. We set the deadline of each packet is set to be 1 second, $\alpha = \beta = 95\%$ and $F = 15$. Then, we analyze the performance obtained by FEBER and the performance obtained through a multiple unicast solution

Due to implementation limitations in the multiple unicast scenario, we were unable

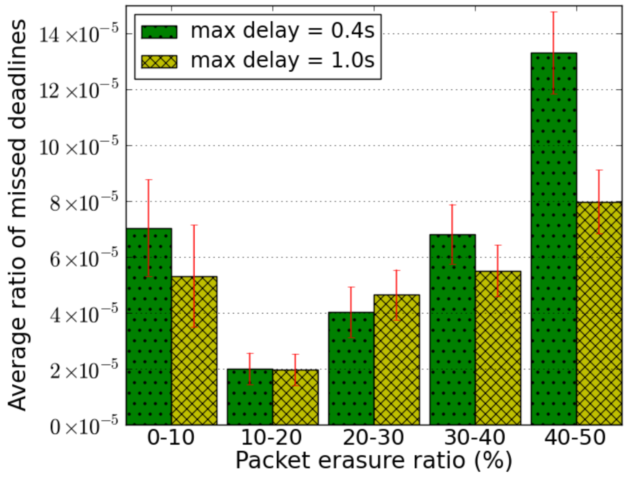


Figure 4.8: Average ratio of packets that missed their deadline, for different maximum delay values. FEBER achieves less than 3 packets with broken deadlines, out of 21040 packets, for packet erasure ratios up to 50%.

to test this scenario with induced errors in the wireless channel. Moreover, since the retransmissions of the unicast transmissions occur at the link layer, we are unable to measure the number of transmissions made by the sender. Thus we were not able to measure the transmission efficiency and the transmission overhead.

Results: The results in *Table 4.1* show the obtained results. While the multiple unicast approach achieves a ratio of broken deadlines of 790.6×10^{-6}, the best ratio experienced by FEBER is 1.432×10^{-6}, for $M = 1$. Even for $M = 4$, where the efficiency of FEBER is maximized at the cost of more deadlines broken, FEBER achieves a ratio of broken deadlines of 169.5×10^{-6}, more than 4 times better than multiple unicast.

4.9 Concluding Remarks

We propose a coding scheme that enables reliable real-time multicast communication over wireless networks. Based on the notion of critical packet, our protocol operates by constructing coded packets that provide valid information to several receivers in a single transmission, ensuring that the amount of packets that do not reach the receivers on time is small. In contrast with previous approaches, where feedback information was considered to be readily available at the sender, we consider the case where the receivers periodically yet sporadically provide feedback information to sender. This

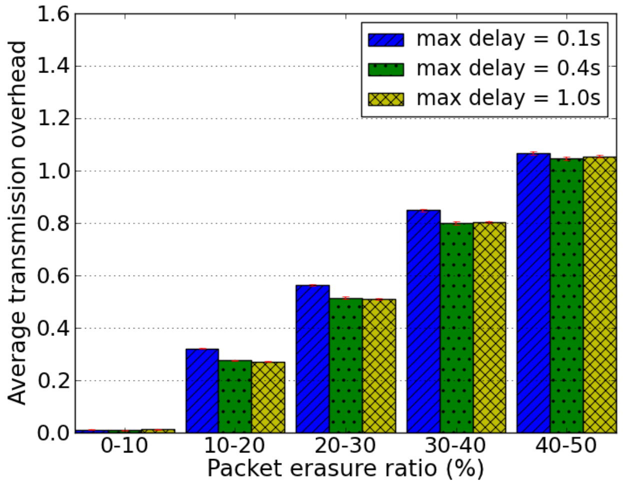


Figure 4.9: Average transmission overhead, for different maximum delay values. Tighter delay constraints lead to a small increase in the experienced transmission overhead.

	Ratio of Missed Deadlines	Deviation
Multiple Unicast	7.906×10^{-4}	8.611×10^{-6}
$M = 1$	1.432×10^{-6}	1.868×10^{-6}
$M = 2$	5.440×10^{-5}	1.436×10^{-5}
$M = 3$	8.487×10^{-5}	2.143×10^{-5}
$M = 4$	1.695×10^{-4}	4.382×10^{-5}

Table 4.1: Average ratio of packets that missed their deadline, for the multiple unicast case and for FEBER. The gains of FEBER over multiple unicast solutions are considerable.

deems our coding scheme suited to standard wireless networks, which we show through the implemention of our solution in a IEEE 802.11 testbed. The obtained results show that the proposed coding scheme is capable of providing the information packets to the receivers on time, with efficient bandwidth usage. Our protocol also provides a trade-off between the level of reliability required and the amount of bandwidth used to achieve such reliability level.

Chapter 5

Half-Duplex Channels with Limited Time

5.1 Motivation

In applications where data has strict deadlines, receivers may require the correct reception of at least a part of the data in order to have useful information. For example, when using multiple description codes or layered codes to transmit multimedia streams, the receiver may demand for a minimum quality, for which the sender has to guarantee the reliable delivery of a subset of the data. Given that this minimum quality is provided, the sender can then aim to deliver more packets in order to further enhance the quality of the stream at the receiver. Meeting such requirements over half-duplex wireless networks is particularly challenging due to the presence of random packet losses and the inherent cost of feedback, specially in the case where feedback information shares the same channel with the data. In the presence of traffic with hard-deadlines, a judicious choice of when to request feedback is instrumental towards achieving these goals. The challenge for the transmitter is then to decide when to stop for receiving feedback information and what to transmission policy to follow next.

In contrast with the previous chapters, we will now jointly consider the choice of the coding policy to follow and the scheduling of the feedback transmissions. Aiming to characterize fundamental limits and practical mechanisms that leverage feedback and coding to increase packet delivery while ensuring that at least a subset of the data is delivered, we make the following contributions:

- *Concept of Reliability Capacity with Limited Transmissions:* We propose a metric for measuring the capacity of a channel given a reliability constraint and the maximum number of allowed transmissions. This metric allows us to illustrate that f feedback packets do not compromise reliability capacity beyond f slots.

- *Systematic Codes with Feedback:* We propose two systematic coding schemes that make use of feedback information to increase the number of delivered packets, without jeopardizing the reliable delivery of a given subset of packets. We consider both the case where the receiver announces a list of missing packets and the case where it announces only the number of missing degrees of freedom.

- *Optimal Policies:* We model such schemes as Markov Decision Processes, which enables us to compute the strategies for when to transmit feedback that optimize the number of delivered packets, while delivering a subset of packets reliably. Although computationally hard to compute, these optimal policies can be computed offline and stored at the transmitter before deployment.

- *Impact of Feedback:* We illustrate that the judicious use of the feedback channel allows to significantly increase the mean number of packets delivered to the end receiver.

The remainder of this chapter is organized as follows. In Section 5.2, we provide an overview of the related work. In Section 5.3, we formalize the problem at hands in this chapter, and we introduce the notion of reliability capacity. Section 5.4 describes two systematic coding schemes that make use of feedback information to increase the number of delivered packets, while in Section 5.5 we analyze the performance of such schemes. Finally, Section 5.6 offers some concluding remarks.

5.2 Related Work

One of the key questions to ask is how many packets can we send reliably, if we are limited in the number of transmissions. This question has a parallel at the physical layer, where recent work [CLB10, PPV10] analyzes the maximum coding rate achievable at a given blocklength and error probability. For multiaccess fading channels, [HT98] investigates the maximum achievable rate with delay independent of how slow the fading is. In [JS09], the authors study the trade-off between blocklengh and decoding delay for a large delay deadline regime, showing that a exponential decay of the probability of violating the deadline can be achieved when the block length scales linearly with the deadline. The role played by the use of periodic and cost-free feedback in this trade-off is analyzed in [SJ08], for the case where the encoder has perfect knowledge of the channel characterization. Our work goes beyond these ideas to incorporate in the problem formulation the cost of having a feedback mode that shares the same channel as the data links. The transmitter still gets a limited number of transmissions but it can adapt to the events at the receiver with a negotiation cost (channel usage).

Related work on joint design of scheduling and feedback for real-time systems has considered dynamic policies to decide when to send the control messages [MYV$^+$04], or how to adapt the rate of control messages [APAM07], but without incorporating the benefits of coding across packets.

In [ERCS07], the authors show that the problem of finding the minimum number of transmissions when using network coding is NP-complete on the binary finite field $\mathbb{F}_2$. In [ERSG10], the broadcast scenario is seen as an instance of the index coding problem, further extending the understanding the hardness of solving such problem.

Even a small amount of feedback information is known to provide significant improvements at the physical layer of wireless communication systems (see [LHL$^+$08] and references therein). With respect to coded networks, the use of feedback is known to enhance network coding performance. Online network coding mechanisms rely on feedback transmitted in parallel channels for maintaining manageable queues [SSM08] and to reduce the decoding delay of individual packets, as seen in Chapter 3. In [SSM$^+$11], the authors discuss how to integrate these network coding mechanisms with TCP mechanisms, with only minor changes to the protocol stack.

Recent work on half-duplex network coding mechanisms [LMS12] propose the use of feedback to request additional coded symbols and prove that there exists an optimal number of coded symbols that can be transmitted before the sender receives an acknowledgment. The work in [NNY07] considers the problem of multimedia wireless broadcast using network coding over $\mathbb{F}_2$. Considering layered codes for the source model, the authors model the problem using a finite horizong Markov Decision Process in order to minimize the mean expected distortion at the receivers. However, the authors assume that the receivers can acknowledge the reception of every transmission, which collides with our model for the wireless channel, where we assume an half-duplex constraint.

5.3 Problem Setup

Consider a sender with N packets, $p_1, \ldots, p_N$, to transmit over a packet erasure channel with loss probability e, and that there are T time slots available to deliver those packets to the receiver. The channel is half-duplex, which means that the receiver can provide feedback information to the sender, but each such transmission requires the use of one of the T time slots available.

The sender has to ensure that at least a subset of $M \leq N$ packets are delivered to the receiver. More precisely, for $\epsilon > 0$, the sender requires that $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$, where D_T denotes the number of packets delivered to the receiver at the end of the T time slots. The sender aims to deliver as many packets as possible in the average while respecting this reliability requirement. Therefore, the problem at hands can be described as follows:

$$\begin{aligned} \text{maximize} \quad & E\{D_T\} \\ \text{subject to} \quad & \mathcal{P}(D_T \geq M) \geq 1 - \epsilon \end{aligned} \tag{5.1}$$

We empower the sender with the ability to code across source packets. More precisely, if $p_1, \ldots, p_N$ are the packets to be delivered to the receiver, the sender transmits $c_1, \ldots, c_l$, with $(c_1, \ldots, c_l) = (p_1, \ldots, p_N) \cdot G$, where G is a $N \times l$ coding matrix. These linear operations are performed over some finite field $\mathbb{F}_q$. We say that a $N \times l$ matrix is *throughput optimal* if and only if any $N \times N$ sub-matrix is invertible. This means that the receiver is able to recover all the N packets upon the successful reception of N coded packets. We start by analyzing the fundamental limit for the number of packets the receiver aims to deliver reliably, M.

5.3.1 Reliability Capacity with Limited Time

The sender aims to deliver M packets reliably and, hence, it requires at least M successful transmissions in the available T time slots. If the sender employs throughput optimal coding, M successful transmissions are sufficient to provide M source packets to the receiver. Therefore, the maximum number of packets M that the sender can aim to deliver reliably, i.e. with probability at least $1 - \epsilon$ in T time slots, can be defined as follows.

Definition 9 *Let T denote the number of available time slots and let e denote the* *packet erasure probability. Let $\epsilon > 0$. The ϵ-reliability capacity of a packet erasure channel is defined as $C_\epsilon^T = \max\{M : \mathcal{P}(S_T \geq M) \geq 1 - \epsilon\}$, where S_T denotes the number of successes in T Bernoulli trials, with success probability $1 - e$.*

The combinatorial nature of C_ϵ^T does not allow for a closed form expression. However, we can make use of the standard normal approximation technique to compute the ϵ-reliability capacity.

Lemma 1 *The ϵ-reliability capacity, C_ϵ^T, verifies*

$$C_\epsilon^T \approx \lfloor T(1-e) + 0.5 + \Phi(\epsilon)\sqrt{Te(1-e)} \rfloor,$$

where $\Phi(\epsilon)$ is such that $\mathcal{P}(X \leq \Phi(\epsilon)) = \epsilon$ for a normal variable X with zero mean and standard deviation 1. The absolute error in the approximation, $|\Delta_C|$, verifies $|\Delta_C| \leq |\phi(\epsilon - \Delta_b) - \phi(\epsilon)|\sqrt{Te(1-e)}$, with $|\Delta_b| \leq \frac{c_0 \cdot (e^2 + (1-e)^2)}{\sqrt{Te(1-e)}}$, where c_0 is the constant in the classic Berry-Esseen [Ber41, Ess42] inequality.

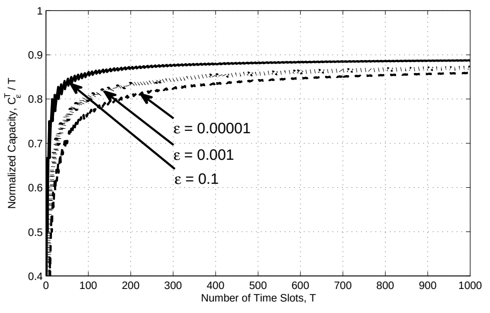
Proof: Let S_T the number of successes in T Bernoulli trials with success probability $1 - e$. Using a standard normal approximation to the binomial distribution with a continuity correction, we have that $\mathcal{P}(S_T \leq k - 1) \approx \mathcal{P}(Z \leq k - 0.5)$, where Z is a normal random variable with mean $T(1-e)$ and variance $Te(1-e)$. Therefore, we have that $\mathcal{P}(S_T \leq k-1) \approx \mathcal{P}\left(X \leq \frac{k-0.5-T(1-e)}{\sqrt{Te(1-e)}}\right)$, where X is a normal random variable with zero mean and variance 1. Now, notice that $C_\epsilon^T = \max\{M : \mathcal{P}(S_T < M) \leq \epsilon\}$. Therefore, since $\mathcal{P}(X \leq \Phi(\epsilon)) = \epsilon$, we have that $\frac{C_\epsilon^T - 0.5 - T(1-e)}{\sqrt{Te(1-e)}} \approx \Phi(\epsilon)$ and the approximation result follows.

With respect to the approximation error, we have that

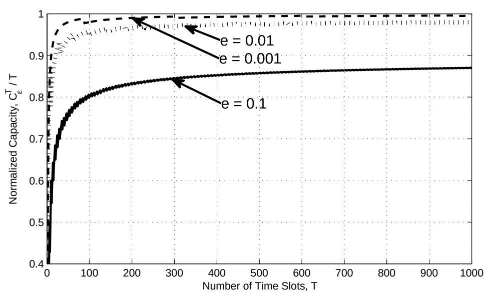
$$\mathcal{P}(S_T \leq k-1) = \mathcal{P}\left(X \leq \frac{k - 0.5 - T(1-e)}{\sqrt{Te(1-e)}}\right) + \Delta_b.$$

Thus, $\mathcal{P}(S_T \leq C_\epsilon^T - 1) = \epsilon$ is equivalent to $\mathcal{P}\left(X \leq \frac{C_\epsilon^T - 0.5 - T(1-e)}{\sqrt{Te(1-e)}}\right) = \epsilon - \Delta_b$, which implies that $\frac{C_\epsilon^T - 0.5 - T(1-e)}{\sqrt{Te(1-e)}} = \Phi(\epsilon - \Delta_b)$ and the bound on the absolute error for C_ϵ^T, Δ_C, follows. Finally, from the Berry-Esseen inequality, we have that the error in the normal approximation for the binomial distribution, Δ_b, verifies $|\Delta_b| \leq \frac{c_0 \cdot (e^2 + (1-e)^2)}{\sqrt{Te(1-e)}}$, for some constant c_0. ■

Over the years, the value of c_0 has been sharpened several times. To the best of our knowledge, [She11] currently provides the most accurate estimate with $c_0 = 0.4748$,



(a) $e = 0.1$ and $\epsilon \in \{0.1, 0.001, 0.00001\}$.



(b) $\epsilon = 0.01$ and $e \in \{0.1, 0.01, 0.001\}$.

Figure 5.1: Normalized Reliability Capacity, C_ϵ^T/T, as a function of the number of time slots T.

which is the value that we will use in the remainder of the chapter.

In Figure 5.1, we can observe the ϵ-reliability capacity (divided by the number of available time slots) as a function of T, as well as the upper and lower bounds. As expected, the normalized capacity increases if we allow higher failure probability ϵ. We also observe the convergence of C_ϵ^T/T to $(1-e)$ as T goes to infinity, the classical capacity for the memoryless packet erasure channel.

5.3.2 Systematic Codes

We focus our attention to the use of systematic codes, since these codes include the uncoded transmissions, which immediately provide a new packet to the receiver upon successful reception, without jeopardizing throughput optimality. These codes with a coding matrix G of the form $G = [I_N|A]$, where I_N is the $N \times N$ identity matrix and A is a $N \times (l-N)$ matrix, with N denoting the number of packets to be delivered and $l(\leq T)$ denotes the number of transmissions. We will consider only throughput optimal systematic codes, i.e. codes for which any subset of N columns of G is linearly independent. From [MS77, p. 321], this implies that every square submatrix of A, of any size, is nonsingular. Low complexity throughput optimal systematic code constructions can be found in [LF04].

In *Table 5.1*, we illustrate the behavior of a systematic code without feedback via a simple example. The sender applies a systematic code over a coding set of 6 packets, for transmissions in the 9 available time slots. The sender starts by transmitting the 6 packets uncoded and, afterwards, applies a coding matrix $A = [\alpha_i^j]$, with the property that any square submatrix is nonsingular.

Table 5.1: Example of a Systematic Code without Feedback

Time Slot	Coding Set	Sent Packet	Channel	Decoded Packets
1	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_1}$	OK	$\mathbf{p_1}$
2	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_2}$	E	$\mathbf{p_1}$
3	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_3}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$
4	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_4}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$
5	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_5}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$
6	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\mathbf{p_6}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$, $\mathbf{p_6}$
7	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\alpha_{\mathbf{1}}^{\mathbf{1}}\mathbf{p_1} + \ldots \alpha_{\mathbf{6}}^{\mathbf{1}}\mathbf{p_6}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$, $\mathbf{p_6}$
8	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\alpha_{\mathbf{1}}^{\mathbf{2}}\mathbf{p_1} + \ldots \alpha_{\mathbf{6}}^{\mathbf{2}}\mathbf{p_6}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$, $\mathbf{p_6}$
9	$\{\mathbf{p_1}, \ldots, \mathbf{p_6}\}$	$\alpha_{\mathbf{1}}^{\mathbf{3}}\mathbf{p_1} + \ldots \alpha_{\mathbf{6}}^{\mathbf{3}}\mathbf{p_6}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$, $\mathbf{p_6}$

Each uncoded transmission provides a new packet to the receiver, upon correct reception, but this is not the case for the later coded transmissions. In fact, notice that, although the receiver has successfully received 5 transmissions, it was able to recover only 3 source packets. Since the coding set has 6 packets, the receiver requires 6 successful transmissions to decode all the packets. Notice that if the sender had chosen a different coding set for its coded transmissions, for instance the coding set $\{p_2, p_4\}$, the 2 successful coded transmissions would allow the receiver to decode this coding set and, thus, recover a total of 5 source packets. However, this optimization requires the sender to know the channel behavior. In Section 5.4, we will see how to use feedback information to adapt the coding set in order to maximize the expected number of delivered packets.

The following lemma exhibits the probability distribution of the number of delivered packets in the end of T time slots, D_T.

Lemma 2 *For a throughput optimal system code over K packets, the probability distribution of D_T is given by*

$$\mathcal{P}(D_T = d) = \begin{cases} \mathcal{P}(S_K = d) \cdot \mathcal{P}(S_{T-K} < K - d), & \textit{if } d < K \\ \mathcal{P}(S_T \geq K), & \textit{if } d = K \end{cases}$$

where S_t denotes the number of successes in t trials with success probability $1 - e$.

Proof: For the case $d = K$, since the code is throughput optimal, any K successful transmissions allow the receiver to recover the K original packets and, thus, $\mathcal{P}(D_T =$

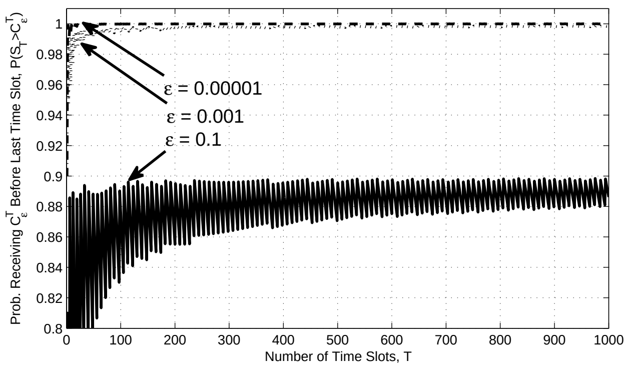


Figure 5.2: $\mathcal{P}(S_T > C_\epsilon^T)$ as a function of the number of time slots T, for $e = 0.1$ and $\epsilon = \{0.1, 0.001, 0.00001\}$.

$K) = \mathcal{P}(S_T \geq K)$. Now, consider the case $d < K$. Given that the matrix A has no singular square submatrix, from [OLV+12, Lemma 2], we have that the (uncoded) packets received in the first K slots do not allow to decode any extra packet from the received columns of A, unless the total number of successful receptions is at least K, in which case we have $D_T = K$. Therefore, for $d < K$, we have that $P(D_T = d) = \mathcal{P}(S_K = d) \cdot \mathcal{P}(S_{T-K} < K - d)$. ■

With the probability distribution for D_T, we can compute $E\{D_T\}$ and solve the optimization problem described in (5.1) (for systematic codes without feedback) by selecting the number of packets to code over, K, that maximizes $E\{D_T\}$, while respecting $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$.

5.3.3 Delivering M packets before the last time slot

The ϵ-reliability capacity represents the maximum number of packets we can provide with probability at least $1 - \epsilon$ to a receiver through a packet erasure channel, within a limited time interval. However, the receiver may get C_ϵ^T successful transmissions before time slot T, in which case the sender could use the remaining time slots to transmit extra packets. In Figure 5.2, we present the probability of the reception of the C_ϵ^T packets being concluded before the end of the available time slots, i.e. $\mathcal{P}(S_T > C_\epsilon^T)$. As we can observe, the probability of such event is significantly high, which means that the sender could typically hope to transmit more than C_ϵ^T packets. If the sender is aware that the receiver has already finished the reception of C_ϵ^T packets, it could use the remaining time slots to deliver any extra packets available. However, since we are considering half-duplex channels, the sender has to stop transmitting to receive feedback information. This leads to a smaller number of time slots available for transmission and to a potentially smaller ϵ-reliability capacity.

Theorem 1 *Consider a packet erasure channel with erasure probability e. The ϵ-reliability capacity in T time slots, C_ϵ^T, verifies $C_\epsilon^T - f \leq C_\epsilon^{T-f} \leq C_\epsilon^T$.*

Proof: We start by proving that $C_\epsilon^{T-f} \leq C_\epsilon^T$. We have that $P(S_{T-f} \geq C_\epsilon^{T-f}) = P(S_{T-f} + S_f \geq C_\epsilon^{T-f})$ and thus, since $S_f \leq f$, $P(S_{T-f} \geq C_\epsilon^{T-f}) \leq P(S_{T-f} \geq C_\epsilon^{T-f})$. By definition, we have that $P(S_{T-f} \geq C_\epsilon^{T-f}) \geq 1-\epsilon$. Therefore, $P(S_T \geq C_\epsilon^{T-f}) \geq 1-\epsilon$, which implies that $C_\epsilon^{T-f} \leq C_\epsilon^T$.

Now, notice that $P(S_T \geq C_\epsilon^T) = P(S_{T-f} \geq C_\epsilon^T - S_f)$. Therefore, since $S_f \leq f$, we have that $P(S_T \geq C_\epsilon^T) \leq P(S_{T-f} \geq C_\epsilon^T - f)$. By definition, we have that $P(S_T \geq C_\epsilon^T) \geq 1-\epsilon$. Therefore, $P(S_{T-f} \geq C_\epsilon^T - f) \geq 1-\epsilon$ and, thus, $C_\epsilon^T - f \leq C_\epsilon^{T-f}$.
■

Theorem 1 asserts that a decrease of f available time slots, for feedback transmissions, does not decrease the capacity by further than f packets.

Notice that, if the number of packets the sender aims to deliver reliably verifies $M \leq C_\epsilon^{T-f}$, the use of f feedback transmissions does not jeopardize the reliability requirement set by the sender. In the following, we will discuss how the sender can effectively use such feedback information to deliver more than C_ϵ^T packets.

5.4 Feedback to Deliver More Packets

Recall that the sender has N packets to deliver to the sender in T time slots over an half-duplex channel, and it aims to deliver at least M of such packets reliably, i.e. the sender requires that $\mathcal{P}(D_T \geq M) \geq 1-\epsilon$. We have seen that, if $M \leq C_\epsilon^{T-f}$, we can use f of the T time slots for feedback transmissions without jeopardizing the aforementioned reliability requirement. We will now describe two different strategies, for different kinds of feedback information, to make use of such information to increase $E\{D_T\}$.

5.4.1 Full Feedback Information: Which Packets are Missing at the Receiver

Now, consider the case where the receiver announces which packets are still missing. We will use a systematic coding structure as described in Section 5.3.2, using the feedback information to adapt the set of packets over which we apply such code. More precisely, the sender will start by transmitting a subset of packets uncoded, after which it either stops transmitting and receives a feedback packet or it proceeds with a systematic code over such packets. If the sender chooses to stop for receiving feedback information, it can then update the set of missing packets at the receiver and repeat the process, by first transmitting a subset of such packets uncoded and then either stop for a new feedback packet or apply a systematic code over such subset of packets.

Upon receiving feedback information, if the sender chooses to make less transmissions than the number of missing packets, it will make uncoded transmissions; otherwise, it will transmit until the last time slot using a throughput optimal systematic code over a subset of missing packets. The sender is thus required to make a decision: how many transmissions it will perform until stopping to receive the next feedback packet. The sender must make this choice with the goal of maximizing $E\{D_T\}$, without compromising the delivery of at least M packets. We formalize this idea in *Algorithm 11*.

Algorithm 11 Systematic Coding with Full Feedback

```
$t = T$ ($t$ represents the number of remaining time slots)
$n = N$ ($n$ represents the number of missing packets)
while $t > 0$ and $n > 0$ do
    According to the selected policy, choose between:
      (a) Choose $k \in \{1, n\}$ packets and transmit them uncoded.
      (b) Choose $k \in \{1, n\}$ packets and perform $t$ transmissions using a
          throughput optimal systematic code over the $k$ packets.
    Update $t$
    Update $n$ via feedback
end while
```

At each decision point (either at the beginning or upon receiving feedback), the sender has to choose between two different strategies: either transmit in all the remaining time slots and, thus, no more feedback is received (option (b)), or transmit a certain number of packets and then receive a feedback packet from the receiver (option (a)). In option (b), the sender applies a throughput optimal systematic code over a subset of missing packets, whereas in option (a) the sender transmits the selected number of packets uncoded.

Table 5.2: Example of a Systematic Code with Full Feedback

Time Slot	Coding Set	Sent Packet	Channel	Decoded Packets
1	$\{\mathbf{p_1}, \dots, \mathbf{p_4}\}$	$\mathbf{p_1}$	OK	$\mathbf{p_1}$
2	$\{\mathbf{p_1}, \dots, \mathbf{p_4}\}$	$\mathbf{p_2}$	E	$\mathbf{p_1}$
3	$\{\mathbf{p_1}, \dots, \mathbf{p_4}\}$	$\mathbf{p_3}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$
4	$\{\mathbf{p_1}, \dots, \mathbf{p_4}\}$	$\mathbf{p_4}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$
5	**Feedback:** Add $\mathbf{p_5}$ to the coding set, and remove $\mathbf{p_1}$ and $\mathbf{p_3}$			
6	$\{\mathbf{p_2}, \mathbf{p_4}, \mathbf{p_5}\}$	$\mathbf{p_2}$	OK	$\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$
7	$\{\mathbf{p_2}, \mathbf{p_4}, \mathbf{p_5}\}$	$\mathbf{p_4}$	OK	$\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$
8	$\{\mathbf{p_2}, \mathbf{p_4}, \mathbf{p_5}\}$	$\mathbf{p_5}$	E	$\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$
9	$\{\mathbf{p_2}, \mathbf{p_4}, \mathbf{p_5}\}$	$\alpha_2\mathbf{p_2} + \alpha_4\mathbf{p_4} + \alpha_5\mathbf{p_5}$	OK	$\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$

In the example in *Table 5.2*, we use the same setting as in *Table 5.1* (more precisely, the same erasure pattern and the same number of packets 6) to illustrate the functioning of a systematic code with full feedback. We start with $t = 9$ and $n = 6$. Here, the sender chooses to start with a coding set composed of 4 packets. The sender transmits these packets uncoded and then stops transmitting in order to receive feedback in time slot 5. Notice that this corresponds to taking option (a) in *Algorithm 11*, with $k = 4$.

After receiving the feedback information, where the receiver announces that it is missing packets p_2 and p_4, the sender updates the current status to $t = 4$ and $n = 4$, and can then remove the already decoded packets (p_1 and p_3) from the coding set. Moreover, in this example, the sender decides to add p_5 to the coding set and uses a throughput optimal systematic code in the 4 time slots remaining, which corresponds to taking option (b) in *Algorithm 11*.

The adaptation of the coding set via full feedback information has lead to more packets delivered to the receiver (5, in contrast with the case without feedback, with 3). The challenge is thus to derive the optimal choices in order to maximize the number of packets delivered, while ensuring the reliable delivery of a subset of packets.

5.4.2 Markov Decision Process Model for the Full Feedback Case

The sender has to find an optimal *policy* (set of decision rules) in order to maximize $E\{D_T\}$, while respecting the reliability requirement $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$. In order to compute such optimal policy, we will model the problem using a *Markov Decision Process* [Bel57], as follows.

5.4.2.1 States

Each state is of the form (n, t), where n represents the number of packets yet to deliver to the receiver and t represents the number of available time slots to deliver the remaining n packets. Notice that the initial state is (N, T) and the state space is $\mathcal{S} = \{(n, t) : n \leq N, t \leq T\}$.

5.4.2.2 Actions

We represent the actions that the sender can take in each decision epoch by $a = (a_1, a_2)$, where a_1 represents the number of transmissions the sender will perform before stopping to receive feedback, and a_2 represents the number of original packets in the coding set used for such transmissions. We denote by $\mathcal{A}^{(\mathrm{a})}_{(n,t)}$ and $\mathcal{A}^{(\mathrm{b})}_{(n,t)}$ the set of possible actions from state (n, t) corresponding to option (a) and option (b) in *Algorithm 11*, respectively, and $\mathcal{A}_{(n,t)} = \mathcal{A}^{(\mathrm{a})}_{(n,t)} \cup \mathcal{A}^{(\mathrm{b})}_{(n,t)}$ denotes the entire set of possible actions from state (n, t).

Recall that the sender requires that $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$. However, it is possible that, at a certain decision epoch, such requirement is unfeasible. More precisely, at state (n, t), if the number of packets so far delivered, $N - n$, verifies $M - (N - n) > C^t_\epsilon$, we have that $\mathcal{P}(D_T \geq M) < 1 - \epsilon$, irrespective of future actions. In this case, the sender does not stop transmitting: it will nevertheless aim to maximize $E\{D_T\}$, but now without any reliability constraint. In such case, i.e. if $M - (N - n) > C^t_\epsilon$, the set of possible actions from state (n, t) corresponding to option (a) in *Algorithm 11* is given by $\mathcal{A}^{(\mathrm{a})}_{(n,t)} = \{(a_1, a_2) : 0 \leq a_1 < t - 1, a_2 = a_1, a_2 \leq n\}$. The condition $a_1 < t - 1$ means that, if the sender is to wait for more feedback, it has to have at least one time slot available after the feedback for transmission, otherwise it would not act upon the received feedback and, thus, the feedback would be useless. In option (a), the sender transmits uncoded packets, which means that $a_2 = a_1$. For option (b), we have that

$\mathcal{A}^{(\mathrm{b})}_{(n,t)} = \{(a_1, a_2) : a_1 = t, a_2 \leq t, 0 \leq a_2 \leq n\}$, where the condition $a_1 = t$ ensures that the sender will use all available time slots for transmission and the condition $a_2 \leq t$ ensures that the transmissions will be performed by using throughput optimal coding over no more packets than the number of available time slots, otherwise the receiver would not be able to decode the received packets.

In case the reliability requirement is still achievable, i.e. if $M-(N-n) \leq C^t_\epsilon$, the sender has to ensure that its choice does not jeopardize such requirement. If $M-(N-n) > C^{t-1}_\epsilon$, the receiver cannot afford to have more feedback transmissions, since it would make $\mathcal{P}(D_T \geq M) \geq 1-\epsilon$ unfeasible. Thus, we have that, if $C^{t-1}_\epsilon < M-(N-n) \leq C^t_\epsilon$, then $\mathcal{A}^{(\mathrm{a})}_{(n,t)} = \emptyset$ and $\mathcal{A}^{(\mathrm{b})}_{(n,t)} = \{(a_1, a_2) : a_1 = t, \mathcal{P}(D^{\mathrm{Syst}}_t(a_2) \geq M-(N-n)) \geq 1-\epsilon\}$, where $\mathcal{P}(D^{\mathrm{Syst}}_t(a_2) = k)$ denotes the probability of delivering k packets in t time slots using a throughput optimal systematic code over a_2 packets, which can be computed from *Lemma 2*.

In case $M-(N-n) \leq C^{t-1}_\epsilon$, we can make use of at least one time slot for feedback transmission without compromising $\mathcal{P}(D_T \geq M) \geq 1-\epsilon$ and, therefore, we have that $\mathcal{A}^{(\mathrm{a})}_{(n,t)} = \{(a_1, a_2) : M-(N-n) \leq a_1 < t-1, a_2 = a_1, a_2 \leq n\}$ and $\mathcal{A}^{(\mathrm{b})}_{(n,t)} = \{(a_1, a_2) : a_1 = t, \mathcal{P}(D^{\mathrm{Syst}}_t(a_2) \geq M-(N-n)) \geq 1-\epsilon\}$.

These restrictions on the set of possible actions ensure that *Algorithm 11* complies with the reliability requirement of delivering at least M packets, as stated in the following result.

Proposition 4 *In the Systematic Code with Full Feedback, the a priori probability of delivering at least M packets verifies $\mathcal{P}(D_T \geq M) \geq 1-\epsilon$, for $M \leq C^T_\epsilon$.*

Proof: The proof follows from the construction of the set of possible actions. If at epoch 0, the sender decides to take $a_1 = t$, it will use a throughput optimal systematic code over a_2 packets during T time slots, where a_2 verifies $\mathcal{P}(D^{\mathrm{Syst}}_T \geq M) \geq 1-\epsilon$, and, thus, we have that the reliability requirement is respected. If the sender decides for $a_1 \leq t-1$, the future actions limit the sender to use a systematic code over a_2 packets, where a_2 is such that the probability of delivering in the remaining time slots what is missing to M packets is at least $1-\epsilon$, i.e. $\mathcal{P}(D^{\mathrm{Syst}}_t \geq M-(N-n)) \geq 1-\epsilon$. Thus, the *a priori* reliability requirement is again respected. ■

5.4.2.3 Transition Probabilities

We now need to compute the transition probability distribution for each state (n, t) and action (a_1, a_2). In the case $a_1 < t - 1$, the sender transmits a_1 uncoded packets and then receives feedback. Then, denoting the next state by (n', t'), we have that $n' = n - S_{a_1}$, i.e. the number of missing packets is the current number of missing packets minus the number of packets successfully delivered in the a_1 transmissions. Moreover, since the source will transmit a_1 packets and then consume one time slot to receive feedback, we have that the number of remaining time slots verifies $t' = t - a_1 - 1$. Therefore, the probability of being in state (n', t') in the next decision epoch, when taking action (a_1, a_2) (with $a_1 < t - 1$) in state (n, t), is given by

$$\mathcal{P}\left((n',t')|(n,t),(a_1,a_2)\right) = \begin{cases} P(S_{a_1} = n - n'), & \text{if } t' = t - a_1 - 1 \text{ and } n - n' \leq a_1 \\ 0, & \text{otherwise} \end{cases}$$

In the case $a_1 = t$, the sender will transmit in all the available time slots and, thus, we must have $t' = 0$. Regarding the next number of missing packets, n', recall that the sender will code over a_2 packets using a throughput optimal systematic coding matrix. Therefore, the probability distribution for n' follows from *Lemma 2* and, thus, the probability of being in state (n', t') in the next decision epoch, when taking action (a_1, a_2) (with $a_1 = t$) in state (n, t), is given by

$$\mathcal{P}\left((n',t')|(n,t),(t,a_2)\right) = \begin{cases} P(S_{a_2} = n - n') \cdot \mathcal{P}(S_{t-a_2} < a_2 - (n - n')), & \text{if } t' = 0 \text{ and} \\ & n - n' < a_2 \\ P(S_t \geq a_2), & \text{if } t' = 0 \text{ and} \\ & n - n' = a_2 \\ 0, & \text{otherwise} \end{cases}$$

5.4.2.4 Rewards

Recall that the sender aims to maximize the number of delivered packets. We will take as optimality criteria for our Markov Decision Process the maximum expected total reward criteria. Thus, we will take as reward the number of packets delivered in the transition from one state to the other, which means that the expected total reward is the total number of packets delivered in the T time slots. More precisely, denoting the reward of going from state (n, t) to state (n', t'), when taking action (a_1, a_2), by

$r((n,t),(a_1,a_2),(n',t'))$, we have that $r((n,t),(a_1,a_2),(n',t')) = n - n'$. The expected reward of taking action (a_1,a_2) when in state (n,t), denoted by $R((n,t),(a_1,a_2))$, can be computed as follows:

$$R((n,t),(a_1,a_2)) = \sum_{(n',t')} r((n,t),(a_1,a_2),(n',t')) \cdot \mathcal{P}\left((n',t')|(n,t),(t,a_2)\right).$$

5.4.2.5 Decision Epochs

The decision epochs in this model occur after each reception of feedback information, as well as at the beginning of the process (which corresponds to epoch 0). Notice that the number of decisions epochs is finite (given that it is upper bounded by T) and, thus, we are in presence of a Markov Decision Process with finite horizon. In fact, given that after each feedback we have at least one transmission, the number of epochs is upper bounded by $1 + \lfloor (T+1)/2 \rfloor$. Notice that we do not force the sender to use $\lfloor (T+1)/2 \rfloor$ feedbacks, since the only allowed action for state $(n,0)$ is $(0,0)$. Thus, the states of the form $(n,0)$ are absorbent. This adaptation allows us to include in our model a variable number of feedback transmissions.

5.4.2.6 Optimal Policy

A policy is a mapping $\pi_j : \mathcal{S} \to \mathcal{A}_{\mathcal{S}}$ that, for each decision epoch j, defines the action to be taken by the sender as a function of the current state. Each policy has a corresponding expected total reward, which is the sum of the rewards throughout the realization of the Markov chain induced by the policy. The goal is to find a policy that maximizes the expected total reward, which in our model corresponds to maximizing the expected number of delivered packets. In order to compute the optimal policy and the corresponding expected number of delivered packets, we make use of the backward induction algorithm [Bel57], which is known to be the an efficient method for solving finite-horizon Markov Decision Processes. Although computationally hard to compute, we can compute the optimal policy offline and then store it at the transmitter before deployment.

5.4.3 Partial Feedback Information: Number of Decoded Packets and Number of Missing Degrees of Freedom

Previously, we assumed that the receiver announced the list of missing packets through feedback transmissions. We further assumed that the receiver was able to convey such list in a single time slot, which is may be hard to comply with in many practical scenarios. To reduce the size of the feedback information, we now consider the case where the receiver announces only the number of missing degrees of freedom, as well as the number of packets of the current coding set it has already decoded. The number of degrees of freedom available at the receiver is the number of independent linear combinations that the receiver has received, which means that if the receiver has n degrees of freedom of n unknown packets it can decode all the n packets.

Now, the sender is not aware of which packets the receiver has already decoded, unless the receiver announces that it has no degrees of freedom missing. Nevertheless, the sender can use the knowledge on the missing degrees of freedom to tune the coding set in order to increase the number of delivered packets, while ensuring that at least M packets are delivered, with probability $1 - \epsilon$.

Similar to Section 5.4.1, we will use a systematic coding structure, while using the feedback information to adapt the set of packets over which such coding is applied. The major difference comes on the reaction to the feedback information: the sender is not able to remove packets from the coding set, since it does not know which packets were already decoded. The exception is the case where the receiver announces zero degrees of freedom missing, which implies that the receiver has recovered the entire coding set and, thus, the sender can use a systematic code over a subset (of its choice) of the packets not yet recovered by the receiver. Otherwise, upon receiving feedback, the sender can choose how many packets it will add to the current coding set (if any) and transmit packets coded over such set. In either case, the sender will have to decide how many transmissions it will perform until stopping to receive feedback. We formalize this scheme in *Algorithm 12*.

Again with the same erasure events and the same number of source packets (6) as in *Table 5.1*, we use the example described in *Table 5.3* to illustrate the behavior of a systematic code with partial feedback. We start with $t = 9$, $n = 6$ and $c = d = r = 0$. Similar to the case of full feedback (*Table 5.2*), the sender starts by transmitting 4 packets uncoded in the first 4 time slots and then stops in order to receive feedback information.

Algorithm 12 Systematic Coding with Partial Feedback

```
$t = T$ ($t$ represents the number of remaining time slots)
$n = N$ ($n$ represents the number of missing packets)
$c = 0$ ($c$ represents the size of the current coding set)
$d = 0$ ($d$ represents the number of undecoded degrees of freedom at the receiver)
$r = 0$ ($r$ represents the number of packets of the current coding set already decoded by the receiver)
while $t > 0$ and $n > 0$ do
    if c = 0 then
        Choose $k \in \{1, n\}$ packets, choose $t'$ and perform $t'$ transmissions using a throughput optimal systematic code over the $k$ packets.
    else
        Choose $k \in \{0, n-c\}$ packets, add them to the coding set, choose $t'$ and perform $t'$ throughput optimal coded transmissions.
    end if
    Update $t$
    Update $n$, $c$, $d$ and $r$ via feedback
end while
```

Upon receiving feedback information in time slot 5, where the receiver announces that it has decoded 2 packets of the coding set used and it is missing 2 degrees of freedom in order to decode the entire coding set, the sender updates the current status to $t = 4$, $n = 4$, $c = 4$, $d = 0$ (notice that the receiver has no linear combination undecoded at its buffer) and $r = 2$. Based on this information, the sender then decides to add another packet to the coding set, p_5, but, in contrast with the full feedback case, it cannot remove the already decoded packets, since it is not aware of which packets are these. Therefore, the sender will then apply coding over the new coding set in the 4 time slots remaining.

The use of feedback information has again lead to a larger number of packets delivered to the receiver (5, in contrast with 3) when compared to the case without feedback. Now, the challenge is to derive the optimal strategies in order to solve the optimization problem at hands, (5.1).

Table 5.3: Example of a Systematic Code with Partial Feedback

Time Slot	Coding Set	Sent Packet	Channel	Decoded
1	$\{\mathbf{p_1}, \ldots, \mathbf{p_4}\}$	$\mathbf{p_1}$	OK	$\mathbf{p_1}$
2	$\{\mathbf{p_1}, \ldots, \mathbf{p_4}\}$	$\mathbf{p_2}$	E	$\mathbf{p_1}$
3	$\{\mathbf{p_1}, \ldots, \mathbf{p_4}\}$	$\mathbf{p_3}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$
4	$\{\mathbf{p_1}, \ldots, \mathbf{p_4}\}$	$\mathbf{p_4}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$
5	**Feedback:** Add $\mathbf{p_5}$ to the coding set			
6	$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	$\alpha_1^1\mathbf{p_1} + \ldots \alpha_5^1\mathbf{p_5}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$
7	$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	$\alpha_1^2\mathbf{p_1} + \ldots \alpha_5^2\mathbf{p_5}$	OK	$\mathbf{p_1}$, $\mathbf{p_3}$
8	$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	$\alpha_1^3\mathbf{p_1} + \ldots \alpha_5^3\mathbf{p_5}$	E	$\mathbf{p_1}$, $\mathbf{p_3}$
9	$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	$\alpha_1^4\mathbf{p_1} + \ldots \alpha_5^4\mathbf{p_5}$	OK	$\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$

5.4.4 Markov Decision Process Model for the Partial Feedback Case

Recall that the sender aims to maximize the expected number of delivered packets, $E\{D_T\}$, while respecting the reliability requirement, $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$. As before, in order to compute the set of optimal choices, we will model the problem as a Markov Decision Process, as follows.

5.4.4.1 States

Given that the sender only has access to the number of degrees of freedom at the receiver, we now also need to keep track of the size of the coding set and the number the degrees of freedom at the receiver. Thus, each state is of the form (n, t, c, d, r), where n represents the number of packets yet to deliver, t the number of remaining time slots, c the size of the current coding set, d the number of undecoded degrees of freedom of the current coding set the receiver already has and r the number of packets of the current coding set already decoded by the receiver. The initial state is $(N, T, 0, 0, 0)$ and, for all states (n, t, c, d, r), we have $n \leq N$, $t \leq T$ and either $c = d = r = 0$ or $c > 0$ and $r + d < c$.

5.4.4.2 Actions

The actions are represented by (a_1, a_2), where a_1 represents the number of transmissions the sender will perform before stopping to receive feedback, and a_2 represents the number of packets the receiver will add to the current coding set.

As in Section 5.4.1, if the reliability requirement is unfeasible at some epoch, the sender will continue transmitting with the goal of maximizing the number of delivered packets, but without any reliability requirements. If $c > M - (N - n)$, i.e. if the current coding set is already larger than what we need to provide M packets, the reliability requirement $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$ is unfeasible if and only if $c - (r + d) > C_\epsilon^t$, since we need $c - (r + d)$ successful transmissions to decode the current coding set. If $c \leq M - (N - n)$, then the reliability requirement is unfeasible if and only if $M - (N - n + d) > C_\epsilon^t$. For the states where the reliability requirement is unfeasible, we have that the set of possible actions is given by $\mathcal{A}_{(n,t,c,d)} = \{(a_1, a_2) : a_1 \in [1, t-2] \cup \{t\}, a_2 \in [0, n - c]\}$. Notice for the states where the reliability requirement is already meet, i.e. $M \leq N - n$, the set of possible actions is the same.

In the states where the reliability requirement is still feasible, but not reached (i.e. $M > N - n$), the sender has to take such requirement into account, which will affect the set of possible actions. First, let us consider the case $c = 0$, where the receiver will employ a throughput optimal systematic code. If $M - (N - n) > C_\epsilon^{t-1}$, the sender cannot afford another feedback transmission and, thus, we must have $a_1 = t$ and a_2 is such that $\mathcal{P}(D_t^{\text{Syst}}(a_2) \geq M - (N - n))$. If $M - (N - n) \leq C_\epsilon^{t-1}$, we can use a feedback transmission and, thus, $a_1 \in [1, t - 2] \cup \{t\}$. Here, if $a_1 \leq t - 2$, we must have a_2 such that $\mathcal{P}(D_{t-1}^{\text{Syst}}(a_2) \geq M - (N - n))$, whereas if $a_1 = t$, a_2 is such that $\mathcal{P}(D_t^{\text{Syst}}(a_2) \geq M - (N - n))$.

For $c > 0$, we need first to identify the states where no more feedback transmissions can be afforded without jeopardizing the reliability requirement. For $c \leq M - (N - n)$, we need at least $M - (N - n + d)$ successful transmissions to deliver M packets and, thus, if $M - (N - n + d) > C_\epsilon^{t-1}$, the receiver has to transmit in all the remaining t time slots. For $c > M - (N - n)$, we need to decode the current coding set in order to deliver M packets and, thus, if $c - (r + d) > C_\epsilon^{t-1}$, no more feedback is allowed. In these two cases where no extra feedback is affordable, we must have $a_1 = t$, while in the cases where we can have more feedback we have that $a_1 \in [1, t-2] \cup \{t\}$. Regarding a_2, since the new coding set has $c + a_2$ packets and the receiver already has d degrees of freedom, we must have $c + a_2 - (r + d)$ successful transmissions to decode this coding set. Thus, we have that, if $a_1 = t$, then a_2 must verify $c + a_2 - (r + d) \leq C_\epsilon^t$, whereas if $a_1 \leq t-2$, we have that a_2 must verify $c + a_2 - (r+d) \leq C_\epsilon^{t-1}$.

These restrictions on the set of possible actions ensure that *Algorithm 12* complies with the reliability requirement of delivering at least M packets, as stated in the following result.

Proposition 5 *In the Systematic Code with Partial Feedback, the a priori probability of delivering at least M packets verifies $\mathcal{P}(D_T \geq M) \geq 1-\epsilon$, for $M \leq C_\epsilon^T$.*

Proof: The proof follows from the construction of the set of possible actions and is analogous to the proof of *Proposition 4*. ■

5.4.4.3 Transition Probabilities

We now need to compute the transition probabilities. For brevity, we denote the state (n,t,c,d,r) by s and the action (a_1,a_2) by a. For $c=0$, the sender will employ a throughput optimal systematic code, which implies that

$$\mathcal{P}\left(s'|s,a\right) = \begin{cases} P(S_{a_2} = n-n') \cdot \mathcal{P}(S_{a_1-a_2} = d' - (n-n')), & \text{if } n' > n - a_2,\ c' = a_2, \\ & r' = n - n' \text{ and} \\ & n - n' \leq d' < a_2 \\ P(S_{a_1} \geq a_2), & \text{if } n' = n - a_2,\ c' = 0, \\ & r' = 0 \text{ and } d' = 0 \\ 0, & \text{otherwise} \end{cases}$$

where, for brevity, we omitted the fact that for $a_1 \leq t-2$, we must have $t' = t - a_1 - 1$, and for $a_1 = t$, we have $t' = 0$.

For the case $c > 0$, the sender transmits coded packets over the current coding set, which means that the receiver will only decode any new packet if it receives the missing degrees of freedom to decode the entire coding set, which is given by $c + a_2 - (r+d)$. Therefore, we have that

$$\mathcal{P}\left(s'|s,a\right) = \begin{cases} P(S_{a_1} = d' - d), & \text{if } n' = n,\ c' = c + a_2, \\ & r' = r \text{ and } d \leq d' < c + a_2 - r \\ P(S_{a_1} \geq c + a_2 - (r+d)), & \text{if } n' = n - (c + a_2),\ c' = 0, \\ & r' = 0 \text{ and } d' = 0 \\ 0, & \text{otherwise} \end{cases}$$

where, for brevity, we again omitted the fact that for $a_1 \leq t-2$, we must have $t' = t - a_1 - 1$, and for $a_1 = t$, we have $t' = 0$.

5.4.4.4 Rewards

As before, the goal is to maximize the number of delivered packets. Thus, denoting by $r(s, a, s')$ the reward of going from state $s = (n, t, c, d, r)$ to state $s' = (n', t', c', d', r')$ when taking action $a = (a_1, a_2)$, we have that $r(s, a, s') = n - n'$. The expected reward of taking action a in state s is thus given by $R(s, a) = \sum_{s'} r(s, a, s') \cdot \mathcal{P}(s'|s, a)$.

5.4.4.5 Decision Epochs

As in the previous model, we are in presence of a finite horizon Markov Decision Process, with the number of decisions epochs upper bounded by $1 + \lfloor (T+1)/2 \rfloor$.

5.4.4.6 Optimal Policy

We will use again the backward induction algorithm to find the policy that maximizes the expected total reward, which corresponds to maximizing the expected number of delivered packets.

5.5 Impact of Feedback on the Number of Delivered Packets

We have described two systematic coding schemes that make use of feedback information in order to maximize the number of delivered packets, while ensuring that at least M of those packets are delivered with probability at least $1 - \epsilon$. In the Systematic Coding with Full Feedback scheme, the receiver announces a list of missing packets, while in the Systematic Coding with Partial Feedback the receiver announces only how many degrees of freedom are missing and how many packets from the current coding set it has already decoded. We will now evaluate the impact of providing these two different kinds of feedback information to the sender, by using the backward induction algorithm [Bel57] to compute the optimal policy for each of these schemes and the corresponding expected number of delivered packets.

5.5.1 Expected Number of Feedback Transmissions

We are also interested in measuring the use of the feedback channel in order to understand what is the effective role feedback transmissions are playing. To compute the expected number of feedback transmissions, we first need to notice that, in a given realization of the Markov chain induced by the optimal policy for each of the schemes, we have at least f feedback transmissions if and only if in epoch f we are in a state with $t > 0$, i.e. the sender still has time slots available at epoch f. Here we are considering the initial epoch to be epoch 0. Let $t_f > 0$ denote the event of having $t > 0$ time slots available in epoch f, and let F denote the number of feedback transmissions. We have that, for any of the coding schemes considered, $\mathcal{P}(F \geq f) = \mathcal{P}(t_f > 0)$ and, therefore,

$$\mathcal{P}(F = f) = \mathcal{P}(t_f > 0) - \mathcal{P}(t_{f+1}),$$

which enables the computation of $E\{F\}$.

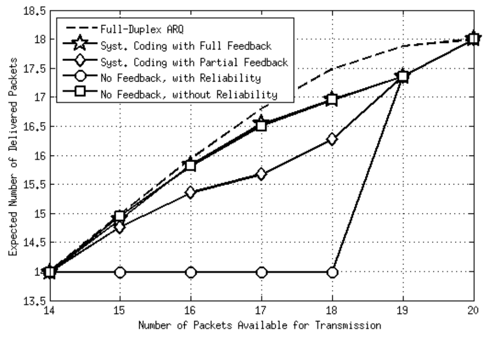
5.5.2 Comparison Schemes

5.5.2.1 Systematic Codes without Feedback

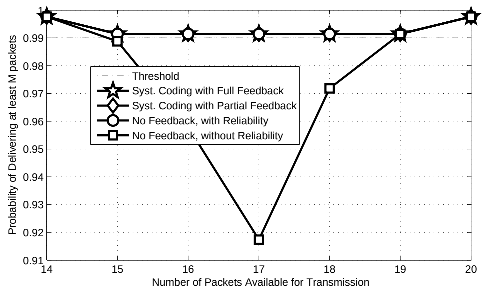
In order to further stress the impact of providing the sender with feedback information, we will compare our schemes against systematic codes without feedback, described in Section 5.3.2. Let k denote the number of packets over which the sender applies a systematic code without feedback. By *Lemma 2*, we can compute the probability distribution for the number of delivered packets by the systematic code over k packets without feedback, D_T. Thus, we compare our schemes against the best choice for k that maximizes $E\{D_T\}$, while satisfying the reliability requirement $\mathcal{P}(D_T \geq M)$. We will also consider the best choice for k for maximizing $E\{D_T\}$, but without the reliability constraint.

5.5.2.2 Optimal Full-Duplex ARQ

This scheme assumes that both the receiver and the sender are capable of receiving and transmitting information at the same time, which in our model means that, within each time slot, the sender can transmit data and the receiver can acknowledge the reception of such transmission. This scheme is thus ideal, in the sense that feedback transmissions do not cost any of the available time slots and, thus, every transmission



(a) Expected Number of Delivered Packets.



(b) Probability of delivering at least M packets.

Figure 5.3: Expected number of delivered packets and probability of delivering at least M packets, for $M = C_{\epsilon}^{T-1}$, $N \geq M$ and for the optimal policy in each scheme. Parameters: $T = 20$, $e = 0.1$, $\epsilon = 0.01$.

can be acknowledged. The sender will select a packet and transmit it until the receiver acknowledges such packet, which then leads the sender to move for the transmission of the next packet. This implies that every successful transmission provides a new packet to the receiver. Therefore, we have that the probability distribution for the number of delivered packets is given by

$$\mathcal{P}(D_T = k) = \begin{cases} \mathcal{P}(S_T = k) & \text{if } k < N \\ \mathcal{P}(S_T \geq N) & \text{if } k = N \\ 0 & \text{otherwise} \end{cases}$$

We can thus easily compute the expected number of delivered packets for the full-duplex ARQ.

5.5.3 Results

In Figure 5.3(a), we present the expected number of delivered packets for the optimal policy in each scheme. We consider only $T = 20$ time slots due to memory and processing constraints, since the state space for the Systematic Coding with Partial Feedback scheme is of the order of magnitude of T^5 and, thus, quite demanding in terms of memory. We will consider the case $T = 100$, but only for the full feedback case, where the state space is of the order of T^2. In both cases, we have chosen $M = C_{\epsilon}^{T-1}$, in order to leave space for at least one feedback transmission without compromising the delivery of at least M packets.

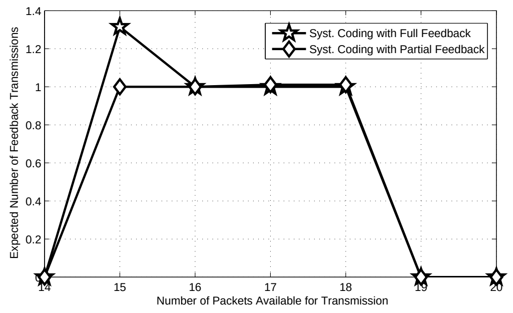
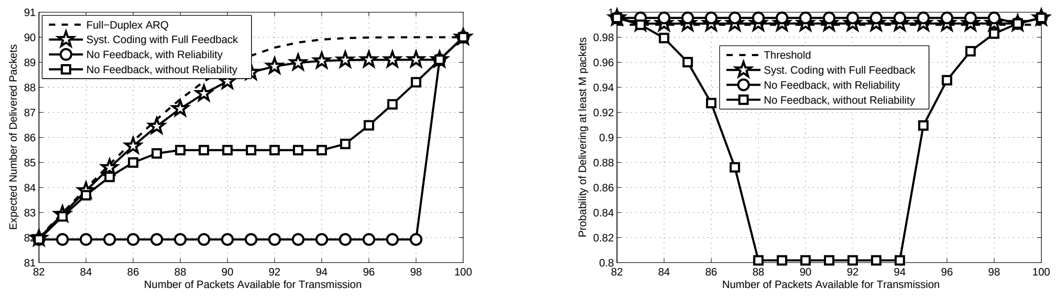


Figure 5.4: Expected number of feedback transmissions, for $M = C_{\epsilon}^{T-1}$, $N \geq M$ and for the optimal policy in each scheme. Parameters: $T = 20$, $e = 0.1$, $\epsilon = 0.01$.

The results exhibit the clear impact of allowing feedback communication, with both feedback enabled schemes delivering significantly more packets than the case without feedback, without jeopardizing the reliable delivery of a subset of M packets. The Systematic Coding with Full Feedback scheme obtains a performance identical to the no feedback case without any reliability constraint, which means that the delivery of at least M is at risk, as depicted in Figure 5.3(b). Thus, the use of full feedback information allows us to improve both the mean number of delivered packets and the reliability level achieved. With respect to the partial feedback case, the performance is worst than the full feedback case, as expected. Nevertheless, the gains over the constrained scheme without feedback are considerable. These results are achieved with the average number of feedback transmissions depicted in Figure 5.4, where we can see that at least one feedback transmission is necessary to achieve the optimal performance (except for the limiting cases of $N = C_{\epsilon}^{T-1}$ and $N \geq T - 1$).

Now, we consider the case of having $T = 100$ time slots available, where due to memory constraints we analyze only the full feedback case. The results depicted in Figure 5.5(a) show the significant impact that the proper use of feedback transmissions has on the number of delivered packets, without compromising reliability. For $T = 100$, the use of full feedback provides significant gains even when compared with the case with no feedback and without any reliability constraint. The gains of using feedback announcing the list of missing packets go up to 8,50% for $N \in [92, 98]$, when compared to the systematic code with reliability constraint. Moreover, the Systematic Code with Full Feedback performs close to the upper bound provided by the full-duplex ARQ scheme, with a difference of at most 1,09%.

Notice that even when compared with systematic coding without feedback and without any reliability constraint, the gains of using feedback information go up to 4,9% in



(a) Expected Number of Delivered Packets. (b) Probability of delivering at least M packets.

Figure 5.5: Expected number of delivered packets and probability of delivering at least M packets, for $M = C_{\epsilon}^{T-1}$, $N \geq M$ and for the optimal policy in each scheme. Parameters: $T = 100$, $e = 0.1$, $\epsilon = 0.01$.

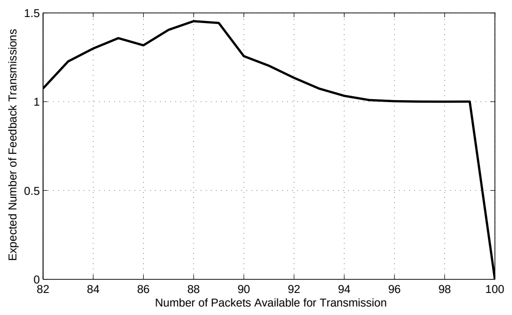


Figure 5.6: Expected number of feedback transmissions, for $M = C_{\epsilon}^{T-1}$, $N \geq M$ and for the optimal policy in each scheme. Parameters: $T = 100$, $e = 0.1$, $\epsilon = 0.01$.

terms of the mean number of delivered packets. Moreover, while the use of feedback ensures that $\mathcal{P}(D_T \geq M) \geq 1 - \epsilon$, such reliability requirement is far from being meet by systematic codes without feedback and without reliability constraints, as depicted in Figure 5.5(b).

These significant gains both in the expected number of delivered packets and in the reliability level achieved are obtained with the average of feedback transmissions depicted in Figure 5.6. With the exception of the case $N \geq T - 1$, the optimal solution makes use of at least one feedback transmission. These results exhibit clearly the advantage of using feedback in our setting, where by using less than 2% of the available time slots for feedback, we can achieve significant gains in terms of both the mean number of delivered packets and the reliability level achieved.

5.6 Concluding Remarks

Interested in the fundamental limits for the transmission of packets within a limited time frame in half-duplex channels, we started by introducing the concept of reliability capacity that describes the maximum number of packets we can aim to deliver reliably over a packet erasure channel. We showed that the use of feedback causes some degradation on the reliability capacity, although bounded by the number of slots dedicated to feedback. We then proposed optimal systematic coding mechanisms to leverage feedback in order to improve the mean delivery of packets, while ensuring the reliable delivery of a subset of the available packets. We considered feedback of two different natures: announcing a list of missing packets and announcing only the number of missing degrees of freedom and the number of decoded packets. For both cases, our numerical results show the considerable gains in incorporating feedback information in the systematic coding schemes, by significantly improving both the mean number of delivered packets and the reliability level achieved. Perhaps the most valuable implication of these results relies on the fact that, even if the feedback channel has a very small rate, a judicious use of this channel yields considerable gains on the system performance. Although the modelling methodology requires highly complex Markov Decision Processes, the optimal policies can be computed offline, which deems the proposed solutions as readly applicable to current wireless systems.

Chapter 6

Conclusions and Future Work

The large potential exhibited by network coding solutions for improving the performance of wireless networks, combined with the growing number of scenarios where time is crucial, motivated the work in this thesis. We set out to analyze the delay performance of network coding solutions and to propose new network coding mechanisms that use feedback information to bring the benefits of network coding to the transmission of data with delay constraints. We showed that standard network coding mechanisms, although being throughput optimal, require that receivers obtain a large set of coded packets before being able to recover any part of the original data. This induces large delay in decoding individual packets, which is not suited for delay constrained scenarios. We exhibited the role that feedback information has on reducing the decoding delay of network coding mechanisms. With information about the reception status of the receivers, the encoder can adapt its coding decisions in order to foster earlier decoding of part of the data, which enables network coding mechanisms to cope with traffic with strict delay restrictions. The proposed solutions allow the system designer to tune how much throughput and bandwidth it sacrifices in order to achieve the desired delay experience. Moreover, our results show that even a small amount of feedback information is enough to significantly improve the delay performance of network coding mechanisms. We now describe the main original contributions of this thesis.

Main Contributions

Immediately Decodable Network Codes for Reducing Decoding Delay

We proposed two network coding algorithms that make use of feedback information to construct coded packets that, upon successful reception, allow receivers to immediately recover a new source packet. In a wide range of scenarios, we showed that restricting network codes to be immediately decodable can have a significant impact in reducing the delay experienced in decoding individual source packets. This improvement in delay comes at the cost of losses in throughput, which are mitigated in the case where receivers can store all the received coded packets and perform gaussian elimination for decoding.

Delay Control Mechanisms for Online Network Codes

We analyzed the decoding delay induced by online network coding mechanisms. We showed that, while being throughput optimal, the delay experienced by the receivers in decoding individual packets can be prohibitive for delay constrained applications, specially in the presence of receivers with different channel conditions. We proposed control mechanisms that use the feedback information available to transmit uncoded packets that allow some receivers to break chains of undecoded packets, thus providing a trade-off between the amount of throughput sacrificed and the corresponding delay experienced.

Network Coding for Traffic with Strict Deadlines over Wireless Networks

For the delivery of multicast traffic with strict deadlines over wireless networks, we proposed a coding scheme that operates on top of the notion of critical packet to provide a small number of packets that do not reach receivers on time, while ensuring an efficient use of the channel bandwidth. The transmitter uses feedback information to carefully select which packets to combine, ensuring that transmissions provide useful information to a large set of receivers, without violating the deadlines of the packets. In contrast with previous approaches, where it is assumed the availability of a parallel cost-free feedback channel, we considered the case where the receivers periodically yet sporadically provide feedback information to the transmitter. This design consideration deems our proposal as readily applicable to current wireless standards, which we showed through the implementation of the proposed scheme in a IEEE 802.11 testbed. The experiences we conducted in this testbed show that our protocol is capable of delivering the information to the receivers on time, with an efficient bandwidth usage. Our protocol also provides a trade-off between the level of reliability required and the corresponding bandwidth consumption.

Optimal Network Codes for Half-Duplex Channels with Limited Time

Finally, we considered the case where a transmitter has a limited amount of time to deliver a set of packets to a receiver, over a wireless channel with half-duplex constraints. We introduced the notion of reliability capacity, i.e. the maximum number of packets the transmitter can aim to deliver if it requires a certain probability of reliable delivery. We then showed that if the reliability level required is below this capacity, the use of feedback information to tune systematic codes can lead to a significantly larger mean number of packets delivered to the receiver, without jeopardizing the reliable delivery of a subset of source packets. For two different types of feedback information, we modeled the problem through Markov Decision Processes, which enabled the derivation of optimal policies for when to use the feedback channel. Perhaps the most striking implication is that, even if the feedback channel allows for a very small rate, a judicious use of this channel can lead to significant improvements in the system performance, both in terms of number of packets delivery and the reliability level achieved.

Future Work

We have considered different approaches to tackle the problem of high decoding delay in standard network coding solutions. In Chapter 2, we focused on immediately decodable schemes that aim at reducing the decoding delay by providing coded packets to the receivers from which they can immediately recover a new source packet, although incurring in throughput looses. In Chapter 3, we took the opposite approach, by providing mechanisms to allow for a fine trade-off between reducing delay and the amount of throughput sacrificed. In these approaches, one common aspect is that packets have equal importance to the receivers. For instance, standard video codecs generate packets that have different relative importance for decoding the video stream. Thus, one natural next step is to include in the proposed delay control mechanisms the different priorities of the source packets.

In Chapter 4, we considered a different kind of delay constraint, by assuming that each packet has a strict deadline, after which the packet becomes useless to the receivers. Apart from this notion of deadline, the solution we proposed is agnostic to the underlying nature of the traffic. Although this deems our solution as widely applicable, for the specific case of video streams, incorporating in the encoding decisions the priority of the different packets with respect to the impact each packet has on the video playout quality may lead to even further gains in the quality of experience provided to the receivers.

The problem analyzed in Chapter 5 takes a first step in this direction, by considering the case where a transmitter has a set of packets to transmit, of which a subset has to be delivered with high probability, thus modeling the case where receivers demand a minimum quality level in the video playout. Including more quality layers and the possible dependencies between source packets is an important step forward to understand the benefits of joint design of network coding mechanisms and the video codec used.

The way multiple receivers provide feedback information in a shared medium also yields interesting challenges, specially in terms of scheduling and medium access. Both in Chapter 2 and Chapter 3, we assumed a perfect cost-free feedback channel, which allows the transmitter to have instantaneous access to the recovery status of each receiver. In practical systems, providing this feedback information is unaffordable. Understanding the impact of imperfect feedback information is thus an important extension to our work.

In Chapter 4, we relaxed the requirements for the feedback information. Receivers provide feedback information periodically yet sporadically, and we showed that this model works in practice, since we were able to implement the proposed protocol in a IEEE 802.11 wireless testbed. Nevertheless, the scheduling of feedback transmissions and the coding decisions are considered independently. Therefore, an important line of research lying ahead is the joint design of the scheduling of feedback and the encoding process. In Chapter 5, we take a first step in this direction, by considering these two problems jointly in order to optimize the number of packets delivered over an half-duplex channel with limited time available for transmission. The next natural step is to extend our models to cope with the presence of multiple receivers.

Appendix A

Further Testbed Details

The analysis of the network coding scheme proposed in Chapter 4 was performed in a IEEE 802.11 wireless testbed. We will now provide some extra details of our implementation.

Characteristics

The access point in the testbed is a PC equipped with a 802.11 wireless card attached to an omni-directional antenna. The wireless card uses the Atheros AR5001X+ chipset. It transmits at a power level of 18 dBm, and operates in the master mode, with RTS/CTS disabled. All receivers transmit at a power level of 20 dBm, and operate in the managed mode, with RTS/CTS disabled. The access point and stations are set to operate in the 802.11g mode. The access point transmits the packets through a multicast connection to the receivers with 36Mbps channel rate. The receivers transmit the feedback information through a unicast connection to the sender, also with 36Mbps channel rate. The disposition of the nodes in the room is illustrated in Fig. A.1.

Packet Format

We use the Click toolkit [KMC$^+$00] to process and manipulate IP packets. We use new packet structures for the feedback packets and for the recovery packets. The respective format for each packet is described next.

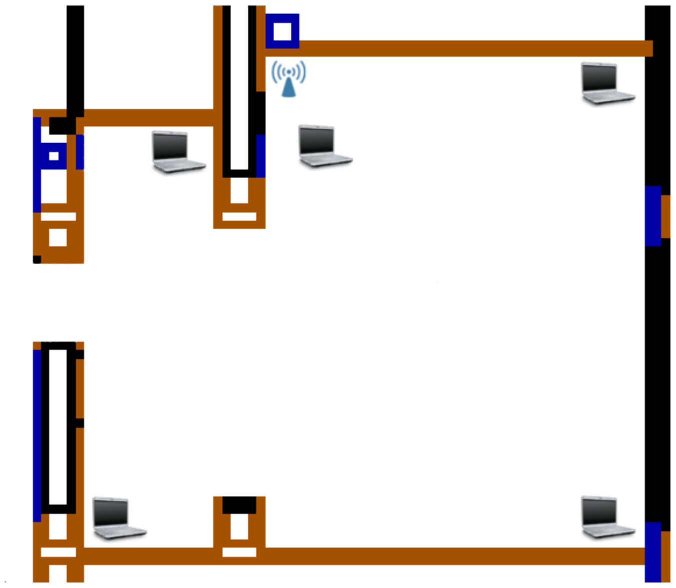


Figure A.1: Disposition of testbed nodes in the room. The testbed is composed of one access point and five receivers. The wireless environment suffers interference from adjacent channels, with 32 different access points broadcasting in the 2.4 Ghz frequency.

Feedback Packet

Each receiver sends information about its buffer status to the sender through a feedback packet, which has the structure described in *Table A.1*.

8 bits	8 bits	8 bits	8 bits
MAC HEADER ...			
SRC_IP			
ENTRIES	LAST_PKT_RX		P_ERASURE
LOST_PKT_ID_0		LOST_PKT_ID_1	
...			

Table A.1: Feedback packet format.

The feedback packet is composed of 5 main components. The *SRC_IP* (32 bits) contains the IP address of the receiver that created the feedback packet. The *ENTRIES* component (8 bits) indicates the number of lost packets that are being announced in the current feedback packet. This is followed by *LAST_PKT_RX* (16 bits), used to inform the sender which was the last source packet successfully received. The *P_ERASURE* (8 bits) refers to the average packet erasure ratio that the receiver is experiencing. Finally, the feedback packet is composed by several *PKT_ID*$_i$ (16 bits each) containing the ids of the source packets that are missing in the receiver's buffer.

Recovery Packet

The recovery transmissions use the packet format described in *Table A.2*.

8 bits	8 bits	8 bits	8 bits
MAC HEADER			
...			
ENTRIES	PKT_ID_0		...
...			
ENCODED DATA			
...			

Table A.2: Recovery packet format.

The recovery packet is composed by 3 main components. The *ENTRIES* (8 bits) refers to the number of packets that are combined in the recovery packet. Then, the recovery packet is filled by *PKT_ID*$_i$ (16 bits), that contains the ids of the packets that are combined in this recovery packet. Finally, the recovery packet is completed with the data containing the bit-wise XOR of the packets aforementioned.

Transmission Queue

Ideally, the encoder would run *Algorithm 8* at every transmission opportunity. However, this approach is unfeasible, because the frequency of the free medium events from Click is too high. The average time between two of these events is significantly smaller than the time necessary to execute one iteration of FEBER, specially if the wireless medium is not very congested. To overcome this issue, we use a transmission queue. When a transmission opportunity is reported, the transmitter pushes the first packet in this queue and sends it to the link layer for transmission.

The transmission queue is constructed as follows. Upon the reception of a new packet from the source, the encoder estimates the maximum number of transmissions available until the next source packet generation by $n_{\text{trans}} = \lfloor \Delta_S / \overline{\Delta}_T \rfloor$, where Δ_S denotes the expected time until the next source packet generation and $\overline{\Delta}_T$ denotes the estimated transmission duration, from Section 4.7.1. Then, for each of these n_{trans} transmission opportunities, the encoder runs FEBER and fills the transmission queue with the constructed packets.

When a new feedback packet is received, the packets in the queue become obsolete, since they were constructed based on now out-of-date information. Thus, the encoder then updates the transmission queue, by removing the packets that involve source packets that were acknowledged with the received feedback packet. If the packets were directly sent to the link layer, the encoder could not cancel the transmission of these obsolete packets.

References

[ABE$^+$96] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. *IEEE Transactions on Information Theory*, 42(6):1737–1744, November 1996.

[ACLY00] R. Ahlswede, N. Cai, S.Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[AF06] A. Asselah and P.A. Ferrari. Hitting times for independent random walks on Zd. *Annals of Probability*, 34(4):1296–1338, 2006.

[APAM07] A. Antunes, P. Pedreiras, L. Almeida, and A. Mota. Dynamic rate and control adaptation in networked control systems. In *Proceedings of the IEEE Conference on Industrial Informatics*, Vienna, Austria, July 2007.

[BCMW09] J. Barros, R.A. Costa, D. Munaretto, and J. Widmer. Effective delay control in online network coding. In *Proceedings of the IEEE Infocom*, pages 208 –216, Rio de Janeiro, Brazil, April 2009.

[Bel57] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, (6), 1957.

[Ber41] A. C. Berry. The accuracy of the gaussian approximation to the sum of independent variables. *Transactions of the America Mathematical Society*, 49:122–136, 1941.

[BLMR98] J.W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28:56–67, October 1998.

[CH96] S. Caser and H. J. Hilhorst. Topology of the Support of the Two-Dimensional Lattice Random Walk. *Physical Review Letters*, 77:992–995, August 1996.

[CLB10] R.A. Costa, M. Langberg, and J. Barros. One-shot capacity of discrete channels. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 211–215, Austin, USA, June 2010.

[Coh92] J. W. Cohen. On the random walk with zero drifts in the first quadrant of R2. *Stochastic Models*, 8(3):359–374, 1992.

[CWJ03] P.A. Chou, T. Wu, and K. Jain. Practical network coding. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, Monticello, US, October 2003.

[DT06] D. Dujovne and T. Turletti. Multicast in 802.11 WLANs: an experimental study. In *Proceedings of the 9th ACM international Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 130–138, Terromolinos, Spain, 2006.

[ERCS07] S.Y. El Rouayheb, M.A.R. Chaudhry, and A. Sprintson. On the minimum number of transmissions in single-hop wireless coding networks. In *Proceedings of the IEEE Information Theory Workshop*, pages 120 –125, Lake Tahoe, USA, September 2007.

[ERSG10] S.Y. El Rouayheb, A. Sprintson, and C. Georghiades. On the index coding problem and its relation to network coding and matroid theory. *IEEE Transactions on Information Theory*, 56(7):3187–3195, July 2010.

[Ess42] C. G. Esseen. On the Liapunoff limit of error in the theory of probability. *Arkiv fr Matematik, Astronomi och Fysik*, A28(9):1–19, 1942.

[FLBW06] C. Fragouli, J. Le Boudec, and J. Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, January 2006.

[FLMP07] C. Fragouli, D. Lun, M. Medard, and P. Pakzad. On feedback for network coding. In *Proceedings of the 41st Annual Conference on Information Sciences and Systems*, pages 248 –252, Baltimore, USA, March 2007.

[HKM$^+$03] T. Ho, R. Koetter, M. Medard, D.R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE*

International Symposium on Information Theory, page 442, Yokohama, Japan, July 2003.

[HLLS04] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Sips. Hybrid rate control for IEEE 802.11. In *Proceedings of the 2nd International Workshop on Mobility Management and Wireless Access Protocols*, pages 10–18, Philadelphia, USA, 2004.

[HMK$^+$06] T. Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, Jun Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, October 2006.

[HT98] S.V. Hanly and D.N.C. Tse. Multiaccess fading channels. ii. delay-limited capacities. *IEEE Transactions on Information Theory*, 44(7):2816–2831, nov 1998.

[JMM93] M. Jolfaei, S. Martin, and J. Mattfeldt. A new efficient selective repeat protocol for point-to-multipoint communication. In *Proceedings of the IEEE International Conference on Communications*, volume 2, pages 1113 –1117, Geneva, Switzerland, May 1993.

[JS09] T. Javidi and R.N. Swamy. Optimal code length for bursty sources with deadlines. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 2694 –2698, Seoul, Korea, July 2009.

[KDF08] L. Keller, E. Drinea, and C. Fragouli. Online broadcasting with network coding. In *Proceedings of the 4th Workshop on Network Coding, Theory and Applications*, pages 1 –6, Hong Kong, China, January 2008.

[KM03] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, October 2003.

[KMC$^+$00] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M.F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18:263–297, August 2000.

[KMFR06] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: maximizing sensor network data persistence. *ACM SIGCOMM Computer Communication Review*, 36(4):255–266, August 2006.

[KRH$^+$08] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, June 2008.

[Lar08] P. Larsson. Multicast multiuser ARQ. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, pages 1985 –1990, Las Vegas, USA, April 2008.

[LF04] J. Lacan and J. Fimes. Systematic mds erasure codes based on vandermonde matrice. *IEEE Communications Letters*, 8(9):570–572, September 2004.

[LHL$^+$08] D.J. Love, R.W. Heath, V.K.N. Lau, D. Gesbert, B.D. Rao, and M. Andrews. An overview of limited feedback in wireless communication systems. *IEEE Journal on Selected Areas in Communications*, 26(8):1341–1365, October 2008.

[LMS12] D.E. Lucani, M. Médard, and M. Stojanovic. On coding for delay - network coding for time-division duplexing. *IEEE Transactions on Information Theory*, 58(4):2330–2348, April 2012.

[LYC03] S.Y.R. Li, R.W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, 1977.

[MWRZ07] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi. Network coding strategies for data persistence in static and mobile sensor networks. In *Proceedings of the 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops*, pages 1–8, Limassol, Cyprus, April 2007.

[MWRZ08] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi. Resilient coding algorithms for sensor network data persistence. In *Proceedings of the 5th European conference on Wireless sensor networks*, EWSN'08, pages 156–170, Bologna, Italy, 2008. Springer-Verlag.

[MYV$^+$04] P. Marti, J. Yepez, M. Velasco, R. Villa, and J.M. Fuertes. Managing quality-of-control in network-based control systems by controller and message scheduling co-design. *IEEE Transactions on Industrial Electronics*, 51(6):1159–1167, December 2004.

[NLC$^+$11] M. Nistor, D.E. Lucani, R.A. Costa, T.T.V. Vinhoza, and J. Barros. On the delay distribution of random linear network coding. *IEEE Journal on Selected Areas in Communications*, 29:1084–1093, May 2011.

[NNY07] D. Nguyen, T. Nguyen, and X. Yang. Multimedia wireless transmission with network coding. In *Proceedings of the IEEE Packet Video*, pages 326–335, Lausanne, Switzerland, November 2007.

[NTNB09] D. Nguyen, T. Tran, T. Nguyen, and B. Bose. Wireless broadcast using network coding. *IEEE Transactions on Vehicular Technology*, 58(2):914 –925, February 2009.

[OLV$^+$12] P. Oliveira, L. Lima, T. Vinhoza, J. Barros, and M. Medard. Coding for trusted storage in untrusted networks. *To appear in IEEE Transactions on Information Forensics and Security*, 7, December 2012.

[PPV10] Y. Polyanskiy, H.V. Poor, and S. Verdu. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, May 2010.

[San07] S. Sanghavi. Intermediate performance of rateless codes. In *Proceedings of the IEEE Information Theory Workshop*, pages 478–482, Bergen, Norway, September 2007.

[She11] I. Shevtsova. On the absolute constants in the Berry-Esseen type inequalities for identically distributed summands. `arXiv:1111.6554v1 [math.PR]`, November 2011.

[Sho06] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551 –2567, June 2006.

[SHW$^+$03] M. Sun, L. Huang, S. Wang, A. Arora, and T. Lai. Reliable MAC layer multicast in IEEE 802.11 wireless networks. *Wireless Communications and Mobile Computing*, 3(4):439–453, 2003.

[SJ08] R.N. Swamy and T. Javidi. Delay analysis of block coding over a noisy channel with limited feedback. In *Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1431 –1435, California, United States, October 2008.

[SM09] H. Seferoglu and A. Markopoulou. Video-aware opportunistic network coding over wireless networks. *IEEE Journal on Selected Areas in Communications*, 27(5):713 –728, June 2009.

[SSM08] J.K. Sundararajan, D. Shah, and M. Medard. ARQ for network coding. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 1651–1655, Toronto, Canada, July 2008.

[SSM$^+$11] J.K. Sundararajan., D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros. Network coding meets tcp: Theory and implementation. *Proceedings of the IEEE*, 99(3):490–512, March 2011.

[TG00] K. Tang and M. Gerla. Random access mac for efficient broadcast support in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 1, pages 454–459, Chicago, USA, 2000.

[TV10] T. Tirronen and J. Virtamo. Fountain-inspired erasure coding for real-time traffic. *Telecommunication Systems*, 48:219–232, May 2010.

[VSS$^+$09] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong. Scalable video multicast using expanding window fountain codes. *IEEE Transactions on Multimedia*, 11(6):1094 –1104, October 2009.

[ZX10] C. Zhan and Y. Xu. Broadcast scheduling based on network coding in time critical wireless networks. In *Proceedings of the IEEE International Symposium on Network Coding*, pages 1 –6, Toronto, Canada, June 2010.