Linear coding with feedback for broadcast communications

Lorenzo Keller

Algorithmic Research in Network Information Group
School of Computer and Communication Sciences

Ecole Polytechnique Fédérale de Lausanne

Report Presented in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

October 2, 2007

Supervisor: Prof. Christina Fragouli
External expert: Prof. Leandros Tassiulas

Contents

Acknowledgments

I would like first to thank Prof. Christina Fragouli for the extensive advice and the many ideas and suggestions. I would like as well to thank Dr. Eleni Drinea for the collaboration that turned into in the theoretical part of this report. Her advice, as well, was precious during the analysis of simulation results. Finally I would like to thank Prof. Leandros Tassiulas for accepting of being part of the evaluation committee. This work was partly supported by the EU project NetReFound FP6 - IST-034413.

Chapter 1

Introduction

In this thesis we analyze the problem of reliably broadcasting a set of messages from a single sender to a set of receivers that are connected through a shared medium. Efficiently broadcasting information in presence of communication errors is not a trivial task because each receiver may receive a different subset of information from the source, while transmissions have to be planned jointly for all receivers.

The problem of reliable broadcasting of information recurs very often in real life engineering applications. The typical setting where such problem arise are wireless applications. When using the wireless medium all receivers near a sender can receive at the same time the data sent by it. Sometimes we are interested in disseminating some information reliably from such sender to all neighbors. There exist other widely used communication media, that are by nature broadcast, where algorithms presented in this report can be run, for example cable television systems or communications over powerline. Internet itself can be used as a broadcast medium if IP multicast is available.

The question we are asking in this thesis is, what are the potential benefits we can gain in this problem from the use of feedback. Such benefits would include operational complexity, reliability and delay. In this report in particular we will focus on reliable transfer and under this constraint optimize for delay. Reliable transfer is required when all the data has to be received. An example is the case of dissemination of firmware updates of networked devices, shipping of high quality video or mirroring of data for backup purposes. Such an approach is in contrast with for instance real time video or audio communications where the fact that there are errors in the communication is tolerated.

Broadcasting has been studied in the literature, mainly along the following three directions:

1. Work in scheduling examines offline and online algorithms, where feedback is available from the destinations to the source, but the source does not employ coding. Performance criteria include achievable rate

and delay, see for example [7].

2. For applications such as satellite communications, where use of feedback might be impractical, elegant forward error correction schemes have been proposed, such as Raptor codes [3] and LT codes [4]. Under the constraint of not using feedback, these codes achieve rate optimality, rate adaptability, and minimize encoding and decoding complexity.

3. Theoretical work has examined use of feedback to achieve the optimal rate and zero probability of error [8] over broadcast erasure channels.

In this report we present results, both theoretical and experimental, on the performance that can be achieved using linear coding techniques that exploit feedback. In the theoretical part we analyze optimal algorithms: some results already known are summarized and some new one are presented. In the experimental part we present some heuristic algorithms and we extensively analyze their performance using simulations. An article with results presented in this report has been submitted to the NetCod '08 workshop.

In Chapter 2 we formally introduce the problem and the model; in the same chapter we present the linear coding approach that will be used in our algorithms, we show how it can be used in our context and we discuss the difference between online and offline algorithms. In Chapter 3 we show some results about optimal codes. Finally in Chapter 4 we propose algorithms and we analyze their performance using extensive simulations.

Chapter 2

Problem statement

2.1 Model

A source S has M messages $m_1, \ldots, m_M$ of k bits and wants to transmit them reliably to N receivers. By reliable transmission we mean that each message is received by each receiver with zero error probability. The messages are randomly chosen from the set of all possible k bits messages and therefore the receivers don't have a priori any information on them.

Whenever the source sends a packet the receivers observe it through N independent erasure channels, one for each receiver. For simplicity we are going to assume these channels are identical. A receiver gets the packet that has been sent with probability $1 - p$, otherwise it receives an erasure symbol. The event of having an erasure on a specific receiver and packet is independent of the event of having it on others and its probability p is time invariant.

Definition 2.1.1. *An erasure pattern E is a realization of the N erasure channels.*

Each receiver has an error-free feedback channel of infinite capacity that connects it to the source. Such type of channels is not realistic, but practical protocols presented in this report can work by just sending one bit of feedback acknowledging each packet transmission, as this is the only information not known at the sender side. A more verbose feedback can be useful when the protocol offloads some computations to the receivers.

Definition 2.1.2. *A broadcast schedule is a list of packets that are sent consecutively by the source.*

In this report we always consider that the time is slotted. At each time step the source can send one message and all the receivers can send feedback based on whether they received the message. The variable t indicates the current time slot, $t = 1$ being the time slot when the first message is sent from the source.

2.2 Linear coding approach

In this report we present different ways to construct broadcast schedules that send linear combinations of messages. Messages can be represented by elements of $\mathbb{F}_{2^k}$. Each packet transmitted contains a linear combination of messages, called $\bar{v}$, built using coefficients from a field $\mathbb{F}_q$ with $q = 2^m$ where m divides k. The field size q is called *alphabet size* [1]. Each possible $\bar{v}$ can be mapped to a *coding vector* v, an element of $\mathbb{F}_q^M$ containing the coefficients used to build it. A packet is a couple $(v, \bar{v})$. We denote Π_i the subspace of $\mathbb{F}_q^M$ spanned by the coding vectors collected by receiver i.

In order to recover the initial set of messages the receiver has to solve a set of linear equations. Let $p_1 = (v_1, \bar{v}_1), \dots, p_t = (v_t, \bar{v}_t)$ the packets received up to time t by some receiver. The receiver can recover the initial messages $m_1, \dots, m_M$ by solving the following equation set:

$$\begin{pmatrix} v'_1 \\ \dots \\ v'_t \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ \dots \\ m_M \end{pmatrix} = \begin{pmatrix} \bar{v}_1 \\ \dots \\ \bar{v}_t \end{pmatrix} \tag{2.1}$$

This system of equations can be solved only if the rank of the matrix of coefficients is M. It is possible however that for l messages $m_{i_1}, .., m_{i_l}$ and for all $0 < j \leq t$ all coefficient $(v'_j)_{i_k}$ with $0 < k \leq l$ are zero. In this case if the rank of the matrix of coefficients is $M - l$ we can partially solve the system to find $M - l$ messages.

The complexity of decoding linear codes depends mainly on two factors. One source of complexity is the alphabet size: operations over a field $\mathbb{F}_q$ of size $q = 2^n$ require with typical algorithms $O(n^2)$ binary operations [1]. Another source of complexity is the kind of linear combinations that are received. If the set of messages that are being decoded is small the number of columns of the decoding matrix is small and therefore inverting it requires less operations. A coding scheme that wants to minimize this complexity tries to make sure that at each packet reception it is possible to decode a new message. This is because when a message has been decoded it can be removed from an incoming packet with a simple subtraction. Therefore the decoding matrix coefficients on the column corresponding to such message can be set to zero thus simplifying the problem. In Appendix A we show an algorithm that allows to decode incrementally the received packet and has a worst case complexity of $O(M^3)$ operations. Its instantaneous complexity depends on how many messages are actively decoded at the same time.

We now present a definition and a lemma useful in the analysis of our algorithms.

Definition 2.2.1. *A packet innovative for receiver R_i is a packet that contains a coding vector linearly independent from the coding vectors previously received by R_i.*

The concept of innovative packet is important when we try to maximize the rate of algorithms: if an algorithm can always send innovative packets for all receivers it achieves the capacity of the erasure channel.

The following lemma helps understand when a receiver can decode a message:

Lemma 2.2.1. *A node R_i can decode the message m_j if and only if $e_j \in \Pi_i$ with Π_i the space spanned by the coding vectors received by R_i and e_j the j-th vector of the canonical basis.*

Proof. If $e_j \in \Pi_i$, e_j can be expressed as a linear combination of coding vectors received by R_i, by using the same coefficient it is possible to reconstruct m_j from the content of the packets R_i has received. $\square$

2.3 Performance metrics

The performance of scheduling algorithms can be analyzed from different points of view. In this report we concentrate mainly on two aspects: delay and complexity. We are interested in finding algorithms that find schedules that have a rate as high as possible and that allow to decode messages at a constant rate. The factors that affect the performance of the algorithms are M, N and p.

This report analyzes various algorithms that create broadcasting schedules that use linear coding. After receiving a packet a receiver could not be able to decode a message for two reasons: the received packet is not innovative or the receiver has not collected enough packets to solve a new part of the equation system. We define formally delay as:

Definition 2.3.1. *The delay $\mathcal{D}_i$ that a receiver R_i experiences is defined as the number of packets that are successfully received by R_i but do not allow to decode messages that couldn't be decoded before receiving them.*

This definition of delay is based on the number of packets successfully received therefore even in the case of a high probability of erasure it could be possible to have a delay equal to zero.

This delay measure penalizes two undesirable behaviors of algorithms: sending many packets that are useless for most of the receivers and sending linear combinations of messages that cannot be decoded when received. These two are not the only undesirable behaviors; another important characteristic of schedules that could be considered is the order in which messages can be decoded. This is important if the data is consumed sequentially at the receiver. It is less important if the receiver has to collect all the messages before using them or in the case of a photo or a video where we can start partially reconstructing the image from any subset of messages. Another metric that could be optimized is the time between a message is sent for the first time in a linear combination and the time when such a message is decoded. This metric is similar to the previous one and is important for real time applications. Another characteristic that could be interesting for an algorithm is that all packets on all receivers can be decoded using other packets that have been recently received. An algorithm that ensures this property could support very a large or even infinite number of messages.

From this discussion on delay we can define two classifications of algorithms that try to optimize the delay metric:

Definition 2.3.2. *An algorithm is rate optimal if for any erasure pattern it produces a broadcast schedule in which a receiver that has received M packets can decode all M messages.*

Definition 2.3.3. *An algorithm is decoding delay optimal if for any erasure pattern it produces a broadcast schedule in which any receiver can from any received packet immediately decode at least one of the source messages.*

If an algorithm is both rate and decoding delay optimal then $\mathcal{D}_w = 0$.

In our theoretical analysis we concentrate on rate optimal algorithms. Under this constraint we attempt to find decoding delay optimal algorithms. We will see that it is not always possible to find an algorithm that is optimal accordingly to both criteria and therefore the optimal algorithm may not require to satisfy both of these constraints.

In order to evaluate the performance of the algorithm for all receivers we define four aggregated versions of the delay metric. The first is the *total delay* defined as:

$$\mathcal{D}_t \triangleq \sum_{i=1}^{N} \mathcal{D}_i$$

This metric directly relates to the average delay:

$$\bar{\mathcal{D}} \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}_i = \frac{1}{N} \mathcal{D}_t$$

The third aggregate metric is the *worst case delay.* This metric is defined as:

$$\mathcal{D}_w \triangleq \max_i \mathcal{D}_i.$$

This metric can indicate how the delay is distributed among the different receivers. If this metric has a value very close to $\mathcal{D}_t$ then the delay is being concentrated on a single receiver; if it is very close to $\bar{\mathcal{D}}$ then the delay is equally distributed to all receivers.

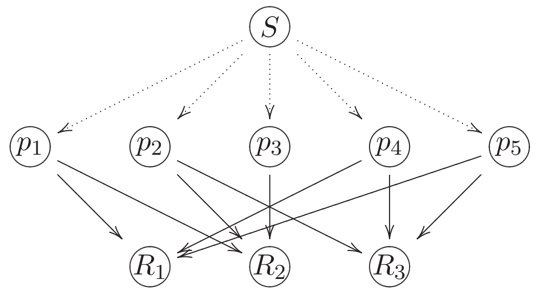

Figure 2.1: Graph representing an erasure pattern for five time slots

In the simulations we use the *median delay*. We use this aggregated metric because it is possible to derive the confidence intervals without knowing the underlying distribution. The median delay is defined as:

$$\mathcal{D}_m \triangleq \text{median}_i\ \mathcal{D}_i.$$

Different algorithms may result in different values for $\mathcal{D}_t$ (respectively $\mathcal{D}_w$); we say that a particular value of $\mathcal{D}_t$ or $\mathcal{D}_w$ is *optimal for a given erasure pattern*, if this is the minimal value achievable by all possible *rate optimal* transmission schemes.

2.4 Taxonomy of scheduling algorithms

An erasure pattern can be represented, as described in [1], by a graph. Such graph contains one vertex that represents the source and N vertices that represent the receivers. Source and receivers are connected through vertices corresponding to time slots. In each time slot the source sends a packet. An edge between a time slot and a receiver exists if and only if the receiver gets the packet sent during such time slot. Figure 2.1 represent the graph for a specific erasure pattern.

Depending on the feedback information required by algorithms to build a broadcast schedule we can classify them in three classes:

- *Forward Error Correction (FEC)*: They do not require access to any feedback information. They can achieve rate optimality asymptotically in the number of messages M [3][4]. They have however some drawbacks. The first is that to achieve rate optimality many messages have to be linearly encoded therefore the decoding delay is usually large. Another issue is that usually decoding is much more complex than using other kind of algorithms that use feedback.

- *Online Algorithms*: They use the feedback information produced by the receivers until the reception of the last packet. Such algorithms try to maximize the expected performance.
- *Offline Algorithms*: They use perfect knowledge of the whole erasure pattern. Such information is not available in our model but they can be very useful to evaluate a bound for performance of any algorithm. This kind of algorithms can be employed if the erasures can be forecast, for instance in a network where traffic is scheduled offline.

2.5 Broadcast scheduling as network coding problem

In [1] it has been shown that it is possible to map the problem of finding a *rate optimal* broadcast schedule to a problem of network coding, i.e., finding a scheme to multicast M streams from a source to N receivers through a network where nodes can linearly recode streams. The graph representing the erasure pattern can be seen as a fixed network with links of unit capacity. Each message can be viewed as a stream of information. The source of the streams is the node corresponding to the sender. The constraint imposing that the receivers want to receive all M streams directly maps to the fact that the receivers want to receive all M messages. The linear combination received in each packet from the receivers corresponds to the linear combination of streams received through the incoming link from each time slot node to each receiver node. Since they receive only one stream, nodes that corresponds to a time slot are forced to send the same information to all receivers: this stream corresponds to the contents of the packet. The following theorem assures that if the min-cut from the source to each receiver is at least M it is possible to deliver the M streams at the same time to the N receivers.

Theorem 2.5.1. *(Main network coding theorem) Consider a directed acyclic graph $G = (V, E)$ with unit capacity edges, h unit rate sources located on the same vertex of the graph and N receivers. Assume that the value of the min-cut to each receiver is h. Then there exists a multicast transmission scheme over a large enough finite field $\mathbb{F}_q$, in which intermediate network nodes linearly combine their incoming information symbols over $\mathbb{F}_q$, that delivers the information from the sources simultaneously to each receiver at a rate equal to h. [2]*

The theorem however doesn't state what is the complexity of finding such multicast scheme and doesn't clearly indicate the minimal field size over which coding operations have to be done. Answers to some of these questions are presented in the theoretical part of this report for our specific network.

Chapter 3

Theoretical analysis

In this chapter we present our theoretical results on optimal codes. In the first section we present a result from [1] that discusses the required field size for rate optimal algorithms. The field size, as mentioned earlier, affects the decoding complexity. Then we introduce two online algorithms described in [6] and [5] that, when $N < 3$ are both rate and delay optimal. In the successive section we show that for $N = 3$ an algorithm that achieves both rate and decoding delay optimality requires some knowledge about the future. In the last section we prove that when $N > 3$ there exist some erasures patterns where the total delay is bounded away from zero even using offline algorithms.

3.1 Finite field size required for rate optimal algorithms

The following theorem presented in [1] shows that online rate optimal linear code algorithms that operate with finite fields exist provided that the alphabet size is large enough. Moreover such algorithms have a reasonable complexity and scale well with the number of receivers.

Theorem 3.1.1. *Assuming perfect feedback and no CSI, a rate optimal code may require encoding operations over a finite field of size N, where N is the number of receivers.*

Proof. Assume that we have two messages m_1 and m_2 to transmit to N receivers. Packet coefficients will be chosen from a space $\mathbb{F}_q^2$. We are interested in determining a lower bound to the alphabet size that has to be used in oder to ensure rate optimality. We can consider only vectors in the set $\mathcal{A} = \{[01], [10], [1\alpha^i] \text{ for } 0 < i \leq q-1\}$ with α a primitive element of $\mathbb{F}_q$ because any other vector is linearly dependent on one of these. At transmission t we only know the erasures up to $t-1$. We will show that in this case the optimal code may need to use an alphabet of size N. Indeed consider the following sequence of realizations:

- The sender uses the coding vector [01]. Only the receiver R_1 receives it successfully.
- The sender now needs to use a different vector in the set $\mathcal{A}$, to ensure that R_1 receives an innovative packet. Assume that only R_2 receives this second packet.
- Continuing along these lines, at transmission k, with $3 \leq k \leq N+1$ the sender needs to use a different vector from $\mathcal{A}$ from the $k-1$ previously employed, to ensure that receivers $R_1, \ldots, R_{k-1}$ do not receive the same packet twice. Thus the set $\mathcal{A}$ needs to have size $q+1 \geqslant N+1$

□

Theorem 3.1.2. *Assuming perfect feedback and no CSI, there exist polynomial time algorithms to construct codes that use an alphabet size of size N.*

Proof. We can simply use an adaptation of the linear flow algorithm proposed in [9]. This algorithm maintains for each receiver R_j a set of received coding vectors, that is updated as the algorithm evolves. To assign a coding vector at transmission t, we can assume conservatively that all receivers (that have not yet received the min-cut value, i.e. that have not yet finished) are going to successfully receive the transmitted packet. Using the feedback information, we can then simply update the sets of the receivers that indeed received this transmission. □

3.2 Optimal algorithm for $N < 3$

In the literature it has already been shown that for $N < 3$ there exist online algorithms that achieve both rate and decoding delay optimality:

- For $N = 1$ the problem reduces to coding for a single erasure channel. The well known ARQ (Automatic Repeat reQuest) algorithm that repeats each packet until is received achieves both rate and decoding delay optimality.
- For $N = 2$ Algorithm 1 can be used to achieve optimality [5]. It works as follows: the sender maintains for each receiver a list of messages S_i that has been received only by that receiver. When it has to choose a new packet if possible it chooses a linear combination of one message from each set. Such packet is instantly decodable by both receivers because both have already received one of the two messages. In the

case one of the two sets S_1 or S_2 is empty the sender sends a message that has never been sent before. It is possible to find such message until one of the two receivers has received all the messages, at that point the transmissions for the remaining receiver can be scheduled as in the case of $N = 1$.

Algorithm 1 Optimal algorithm for $N = 2$

```
procedure BROADCAST($\{m_i\}_{i=1..M}$)
    $U \leftarrow \{m_i\}_{i=1..M}$
    $S_1 \leftarrow \emptyset$
    $S_2 \leftarrow \emptyset$
    while $R_1$ and $R_2$ have not received all messages do
        if $S_1 = \emptyset \vee S_2 = \emptyset$ then
            pick a message $m_i$ from $U$
            send the packet $p = (e_i, m_i)$
            if $R_1$ has received $p$ then
                $S_1 \leftarrow S_1 \cup \{m_i\}$
            end if
            if $R_2$ has received $p$ then
                $S_2 \leftarrow S_2 \cup \{m_i\}$
            end if
        else
            pick a message $m_i$ from $S_1$
            pick a message $m_j$ from $S_2$
            send the packet $p = (e_i + e_j, m_i + m_j)$
            if $R_1$ has received $p$ then
                $S_1 \leftarrow S_1 \cup \{m_j\}$
            end if
            if $R_2$ has received $p$ then
                $S_2 \leftarrow S_2 \cup \{m_i\}$
            end if
        end if
        $S_1 \leftarrow S_1 - (S_1 \cap S_2)$
        $S_2 \leftarrow S_2 - (S_1 \cap S_2)$
    end while
    for $m_i \in S_j$ where $S_j$ is the receiver that has finished do
        send the packet $p = (e_i, m_i)$ until $R_j$ receivers it
    end for
end procedure
```

3.3 Optimal algorithm for $N = 3$

The next theorem proves that for $N = 3$ there exists an offline algorithm that can achieve both rate and decoding delay optimality, thus results in delay 0 for every receiver.

Theorem 3.3.1. *There exists a polynomial time offline algorithm that can find an optimal schedule achieving $\mathcal{D}_w = 0$ for $N = 3$. This algorithm needs to know only the erasures that will affect the next packet, and uses binary operations.*

Proof. Let $p(t)$ denote the packet the source sends at time t. Let $B_S(t)$ be the set of messages that the source has used to build the packets up to time $t-1$, and let $S_i(t) \subseteq B_S(t)$ be the subset of the $B_S(t)$ messages that node R_i has *not* decoded, before the transmission of packet $p(t)$. Moreover, define $S_{ij}(t) \triangleq S_i(t) \cap S_j(t)$. For simplicity of notation, we may omit the time index t when there is no confusion. At $t = 1$, before transmissions begin, $B_S = S_1 = S_2 = S_3 = \emptyset$. We create the packet $p(t)$ to be transmitted at time t as follows. We distinguish three cases:

1. Assume that only one node R_i receives $p(t)$. Then

   a) If $S_i(t) = \emptyset$, then we select $p(t) = (e_{new}, m_{new})$ with $m_{new} \notin B_S(t)$. We update $B_S(t+1) = B_S(t) \cup \{m_{new}\}$, $S_j(t+1) = S_j(t) \cup \{m_{new}\}$ for $j \neq i$, and $S_{jk}(t+1) = S_{jk}(t) \cup \{m_{new}\}$ for $j, k \neq i$.

   b) Otherwise, transmit $p(t) = (e_{old}, m_{old})$ with $m_{old} \in S_i(t)$, and update $S_i(t+1) = S_i(t) \setminus \{m_{old}\}$. Also, if $m_{old} \in Sij(t)$, update $Sij(t+1) = S_{ij} \setminus \{m_{old}\}$ for $j \neq i$.

2. Assume two nodes R_i and R_j receive $p(t)$. Then

   a) If $S_i(t) = \emptyset$ or $S_j(t) = \emptyset$, transmit $p(t) = (e_{new}, m_{new})$ with $m_{new} \notin B_S(t)$, and update $B_S(t+1) = B_S(t) \cup \{m_{new}\}$, $S_k(t+1) = S_k(t) \cup \{m_{new}\}$ for $k \neq i, j$.

   b) Otherwise, if $S_{ij}(t) \neq \emptyset$, transmit packet $p = (e_{old}, m_{old})$ with $m_{old} \in S_{ij}$, while if $S_{ij}(t) = \emptyset$ transmit $p(t) = (e_i + e_j, m_i + m_j)$ with $m_i \in S_i$ and $m_j \in S_j$.

3. Assume all three nodes successfully receive the transmitted packet. We then send $p(t) = (e_{new}, m_{new})$ with $m_{new} \notin B_S(t)$. Sending a new message is the only choice, because this algorithm belongs to the class of algorithms described by Proposition 3.4.1 at the end of the next section, and thus there exists at least one receiver R_i with $S_i = \emptyset$.

Once one of the three receivers has received all M packets, we can use the optimal algorithm for $N = 2$ receivers. Note that the most computationally intensive task at each step of the described algorithm, is the search of whether $m \in S_{ij}(t)$. Since $S_{ij}(t)$ has size at most M, the search can be performed in $\mathcal{O}(M \log M)$ operations. □

Unfortunately it is not possible to build an online algorithm that is at the same time rate and decoding delay optimal. The following lemma proves this result:

Lemma 3.3.1. *There exist no online algorithm that is rate and decoding delay optimal for $N > 2$ and $M \geq 2$*

Proof. A simple example can show that some patterns incur at least one delay. Consider the first two transmissions and suppose the first packet is received only by receiver R_1 and the second packet only by receiver R_2. Any rate optimal algorithm has to send two different packets at each transmission and any decoding delay optimal algorithm has to send one message per packet to make it decodable by all receivers. The next $M - 2$ packets are received by all receivers. Like the previous ones they have to contain only one new message. The $M + 1$ packet has to be a linear combination of messages in order to make the scheduling rate optimal for R_1 and R_2, but no linear combination can be decoded by all the receivers, so depending on the choice of the algorithm it is possible to find an erasure pattern that delivers the linear combination to the receiver that can't decode and therefore there will be at least a delay of 1 on one receiver. □

We show now that the greedy algorithm that has been presented for $N = 2$ has a worst case performance if used with $N = 3$ of the order of M. Thus it performs equally bad, with respect to this metric, as FEC schemes. Before we have to introduce a lemma describing a constraint that rate optimal linear coding algorithms have to satisfy. The notation $\Pi_1 \cup \Pi_2$ and $\Pi_1 \cap \Pi_2$ is used to denote the common span and the intersection of two subspaces Π_1 and Π_2, respectively.

Lemma 3.3.2. *Let Π_1 denote the subspace spanned by the coding vectors $< v_1 \dots v_{t_1} >$ receiver R_j has collected up to time t_1, and Π_2 the subspace spanned by the coding vectors $< v_{t_1+1} \dots v_{t_2} >$ receiver R_j collects between times $t_1 + 1$ and t_2. In a rate optimal scheme, if $e_i \in \Pi_1$, then $e_i \notin \Pi_2$, for all i, $t_1 < M$ and $t_2 \leq M$.*

Proof. Assume the condition does not hold. This implies that $\dim(\Pi_1 \cap \Pi_2) > 0$ and therefore $\dim(\Pi_1 \cup \Pi_2) < t_2$, which contradicts rate optimality. □

Theorem 3.3.2. *The greedy online algorithm with $N = 3$ has a worst case performance of at least $\frac{M}{2}$.*

Proof. Consider an erasure pattern in which the first $\frac{M}{2}$ packets are received only by R_1, the next $\frac{M}{2}$ packets are received only by R_2 and the following $\frac{M}{2}$ packets are received by all three nodes. After M packets have been sent, R_1 and R_2 have decoded exactly two disjoint sets of $\frac{M}{2}$ messages and R_3 has still not received any packet. According to Lemma 3.3.2 no message already decoded by some receiver can be decoded by R_3. Therefore he will not decode until $\frac{M}{2}$ packets have been sent to R_1 and R_2. The delay for R_3 is therefore $\frac{M}{2}$. □

3.4 Properties of the optimal algorithm when $N > 3$

As we have seen in the previous section for $N > 2$ the delay of online algorithms cannot be zero, while offline algorithms can still be rate and decoding delay optimal. In this section we show that for $N > 3$ this is not the case, more specifically we present some results that prove that when $N > 3$ even offline algorithms cannot have zero delay. We show that if we force rate optimality the delay for some erasure patters is even $O(M)$, i.e. the same that can be achieved using FEC codes.

In this section we always assume that $M = 2k$ for some positive integer $k \geq 2$. Consider the following erasure pattern. For $1 \leq j \leq 4$, let π_j be the time period corresponding to transmissions $(j-1)\cdot M/2+1$ up to $j\cdot M/2$. Each receiver receives every packet sent during the following periods and receives nothing during the other periods: R_1 receives in π_1 and π_2, R_2 in π_1 and π_3, R_3 in π_2 and π_3, and R_4 in π_3 and π_4. The resulting erasure pattern is depicted in Table 3.1 for $M = 4$; we will call this pattern E, the dependence on M is implied. Since we only consider rate optimal schemes, no receiver requires further transmissions after time $2M$ since each has already collected M packets by then.

The first result shows that there exits an erasure patter with $N = 4$ and M even where the worst case bound $O(M)$ is achieved.

Lemma 3.4.1. *Consider $N = 4$ receivers that want to receive messages $m_1, ..., m_M$ with $M = 2k$. For the erasure pattern E and any rate optimal transmission scheme*

$$\mathcal{D}_t \geq M/2. \tag{3.1}$$

A transmission scheme that achieves the equality above is depicted in Table 3.2 for $M = 4$ (generalizing for $M = 2k$ is straightforward).

Proof. Let $\Pi_1, \ldots, \Pi_4$ be the subspaces generated by the coding vector in the packets the source transmits during $\pi_1, \ldots, \pi_4$. Also, let $\alpha_1, \alpha_2 \geq 0$ be the delay R_1 experiences during π_1, π_2 respectively, $\beta_1, \beta_2 \geq 0$ the delay R_2 experiences during π_1, π_3 respectively, $\gamma_1, \gamma_2 \geq 0$ the delay R_3 experiences

Time slot	R_1	R_2	R_3	R_4
1	–	–	x	x
2	–	–	x	x
3	–	x	–	x
4	–	x	–	x
5	x	–	–	–
6	x	–	–	–
7	x	x	x	–
8	x	x	x	–

Table 3.1: Erasure pattern for $N = 4$, and $M = 4$, where "x" denotes erasure, and "–" successful reception.

during π_2, π_3 respectively, and $\delta_1, \delta_2 \geq 0$ the delay R_4 experiences during π_3, π_4 respectively. Then $\alpha_1 = \beta_1$ since R_1 and R_2 both receive Π_1 during π_1 and this is the first period they receive any packets. However, it could be $\gamma_1 > 0$ while $\alpha_2 = 0$ since R_1 has already received some packets in π_1 hence might be able to decode messages from Π_2 that R_3 can not. From rate optimality it holds that:

(i) $\Pi_1 \cup \Pi_2 = \mathbb{F}_q^M$, $\Pi_1 \cup \Pi_3 = \mathbb{F}_q^M$, and $\Pi_2 \cup \Pi_3 = \mathbb{F}_q^M$, and

(ii) $\Pi_i \cap \Pi_j = 0$.

Let $E_k \triangleq \{e_i\} \subseteq \Pi_k$ be the set of $\{e_i\}$ vectors contained in Π_k, $k = 1, \ldots, 3$. Then:

(iii) $E_i \cap E_j = 0$ from (ii),

(iv) $\alpha_1 = \beta_1 \geq \frac{M}{2} - |E_1|$, $\gamma_1 \geq \frac{M}{2} - |E_2|$, $\delta_1 \geq \frac{M}{2} - |E_3|$ (because R_1, R_2 observe only Π_1, R_3 only Π_2 and R_4 only Π_3), and

(v) $|E_1| + |E_2| + |E_3| \leq M$.

Since $\mathcal{D}_t = \alpha_1 + \beta_1 + \alpha_2 + \gamma_1 + \beta_2 + \gamma_2 + \delta_1 + \delta_2$, we have

$$\begin{aligned}
\mathcal{D}_t &\geq \alpha_1 + \beta_1 + \gamma_1 + \delta_1 \\
&\geq 2(\frac{M}{2} - |E_1|) + (\frac{M}{2} - |E_2|) + (\frac{M}{2} - |E_3|) \\
&= 2M - |E_1| - (|E_1| + |E_2| + |E_3|) \\
&\geq \frac{M}{2}.
\end{aligned} \tag{3.2}$$

where the first inequality follows from $\alpha_2, \beta_2, \gamma_2, \delta_2 \geq 0$, the second from (iv), and the last from $|E_1| \leq M/2$ and (v). $\square$

Equation (3.2) further yields the following lower bound for the total delay of R_2, R_3 and R_4.

Corollary 3.4.1. *The total delay of R_2, R_3 and R_4 is at least $M/2$ under* any *rate optimal transmission scheme for E.*

We conclude that for the transmissions schemes that achieve equality for (3.1), it must hold that $\alpha_1 = 0$. This implies $\beta_1 = 0$ as well; since achieving the lower bound in (3.1) also requires $\alpha_2 = \beta_2 = \gamma_2 = \delta_2 = 0$, we obtain the following corollary.

Corollary 3.4.2. *Any rate optimal transmission scheme that optimizes $\mathcal{D}_t$ for the erasure pattern E satisfies*

$$\gamma_1 + \delta_1 = M/2.$$

Intuitively this corollary states that delay introduced in π_1 is more costly because they delay two receivers (R_1 and R_2).

Corollary 3.4.1 refines the simple lower bound of $M/8$ for $\mathcal{D}_w$ provided by Lemma 3.4.1 to $\lceil M/6 \rceil$. It is now clear from Corollary 3.4.2 that for the transmission schemes that achieve the optimal $\mathcal{D}_t$, $\mathcal{D}_w$ exceeds the lower bound of $\lceil M/6 \rceil$. However, Table 3.3 depicts a transmission scheme that achieves $D_w = \lceil M/6 \rceil$ for $M = 8$ (generalizing for any $M = 2k$ is straightforward). Hence we obtain the following corollary.

Corollary 3.4.3. *Consider $N = 4$ receivers that want to receive messages $m_1, ..m_M$ with $M = 2k$. For the erasure pattern E and any rate optimal transmission scheme*

$$\mathcal{D}_w \geq \lceil M/6 \rceil. \tag{3.3}$$

By Corollary 3.4.1, the schemes that achieve the optimal $\mathcal{D}_w$ satisfy the following property.

Corollary 3.4.4. *Any rate optimal transmission scheme that optimizes $\mathcal{D}_w$ for the erasure pattern E satisfies*

$$\lfloor M/6 \rfloor + \lceil M/6 \rceil \leq \beta_1 + \gamma_1 \leq 2 \cdot \lceil M/6 \rceil.$$

Observe that the total delay of any of the above schemes exceeds the total delay of any of the schemes described by Corollary 3.4.2 by at least $\lfloor M/6 \rfloor$. This implies that depending on the measure of delay we wish to minimize, different strategies should be considered.

Besides its interest for the study of offline algorithms, the erasure pattern E is also worth studying because it can model a dynamic scenario where new

Time slot	R_1	R_2	R_3	R_4
1	e_1	e_1	x	x
2	e_2	e_2	x	x
3	e_3	x	e_3	x
4	e_4	x	e_4	x
5	x	$e_1 \oplus e_3$	$e_1 \oplus e_3$	$e_1 \oplus e_3$
6	x	$e_2 \oplus e_4$	$e_2 \oplus e_4$	$e_2 \oplus e_4$
7	x	x	x	e_1
8	x	x	x	e_2

Table 3.2: Coding vectors of the optimal schedule for D_t for erasure pattern in Table 3.1.

Time slot	R_1	R_2	R_3	R_4
1	$e_1 \oplus e_2$	$e_1 \oplus e_2$	x	x
2	e_3	e_3	x	x
3	e_4	e_4	x	x
4	e_5	e_5	x	x
5	$e_3 \oplus e_6$	x	$e_3 \oplus e_6$	x
6	$e_4 \oplus e_7$	x	$e_4 \oplus e_7$	x
7	e_1	x	e_1	x
8	e_8	x	e_8	x
9	x	e_2	e_2	e_2
10	x	e_6	e_6	e_6
11	x	e_7	e_7	e_7
12	x	$e_5 \oplus e_8$	$e_5 \oplus e_8$	$e_5 \oplus e_8$
13	x	x	x	e_5
14	x	x	x	e_4
15	x	x	x	e_3
16	x	x	x	e_1

Table 3.3: Coding vectors of the optimal schedule for D_w for erasure pattern E and $M = 8$.

receivers join the system at different times and we do not want to penalize them with very long delay.

It is worth mentioning here that the algorithm described in the proof of Theorem 1 does not readily generalize to the optimal offline solution for

$N > 3$ (e.g., see Tables 3.4 and 3.5). It is interesting to understand why this algorithm succeeds for $N = 3$ but fails for $N = 4$. The "problem" occurs when three receivers successfully receive the transmitted packet. In this case, we can always find a packet such that at least two receivers can decode, but we cannot always find a packet so that all three receivers decode (e.g., see Table 3.1).

Time slot	Packet sent	R_1	R_2	R_3	R_4
1	e_1	e_1	e_1	x	e_1
2	e_2	e_1,e_2	x	x	e_1,e_2
3	e_3	x	e_1,e_3	x	e_1,e_2,e_3
4	e_1	x	x	e_1	x
5	$e_2 \oplus e_3$	e_1,e_2,e_3	e_1,e_2,e_3	e_1,$e_2 \oplus e_3$	x
6	e_2	x	x	e_1,e_2, e_3	x

Table 3.4: Coding vectors selected by greedy (delay 1).

Time slot	Packet sent	R_1	R_2	R_3	R_4
1	e_1	e_1	e_1	x	e_1
2	e_2	e_1,e_2	x	x	e_1,e_2
3	e_3	x	e_1,e_3	x	e_1,e_2,e_3
4	e_2	x	x	e_2	x
5	$e_2 \oplus e_3$	e_1,e_2,e_3	e_1,e_2,e_3	e_2, e_3	x
6	e_1	x	x	e_1,e_2, e_3	x

Table 3.5: Coding vectors of the optimal schedule (delay 0).

We conclude this section with a proposition that shows that for any N, we can have a receiver that experiences zero delay.

Proposition 3.4.1. *There exists a rate optimal offline algorithm where at least one receiver has received all the information the source has transmitted.*

Proof. Let $p(t)$ denote the packet the source sends at time t. Let $\Pi_S(t)$ be the subspace spanned by the coding vectors contained in the packets that the source S has transmitted, and $\Pi_i(t) \subseteq \Pi_S(t)$ the subspace generated by the coding vectors received by destination R_i, before the transmission at time t. We will prove that, provided $\dim(\Pi_S(t)) < M$, there exists at least one receiver such that $\Pi_S(t) = \Pi_i(t)$. In other words, the set $R^*(t) = \{R_i \in \text{receivers} \,|\Pi_i(t) = \Pi_S(t)\}$ is not empty. Our proof uses induction. For $t = 1$, $\Pi_S(t) = \Pi_i(t) = \{0\}$, and the condition is satisfied. Assume the condition holds at time t. Then we distinguish two cases:

- There exists a receiver $R_i \in R^*(t)$ that receives $p(t)$. Clearly a rate optimal code will then need to send a packet $p(t) = (e_k, m_k)$ with $m_k \notin \Pi_S(t)$. We will then have that

  $$\Pi_S(t+1) = \Pi_S(t) \cup m_k = \Pi_i(t) \cup m_k = \Pi_i(t+1)$$

  and R_i will still belong in $R^*(t+1)$.

- There does not exist a receiver $R_i \in R^*(t)$ that receives $p(t)$. Then, we can select a packet $p(t) = (e_k, m_k)$ with $m_k \in \Pi_S(t)$ that brings innovative information to all receivers. Note that $\Pi_S(t) = \Pi_S(t+1)$, and as a result, $R^*(t) \subseteq R^*(t+1)$.

$\square$

Chapter 4

Empirical analysis of heuristic algorithms

In this chapter we first present two variations of the random network coding algorithm, a well known FEC algorithm, that are used as benchmarks in our performance evaluation. Then we present three decoding delay optimal algorithms, one that doesn't use any linear coding, one that opportunistically sends instantly decodable linear combinations chosen randomly and one that chooses the messages to be encoded such that receivers that are experiencing more delay are helped. We also present a rate optimal algorithm that tries to choose messages to be encoded such that the packets are instantly decodable by as many receivers as possible. Since we define the delay at a receiver to be the sum of decoding delay and delay due to useless packets, it is not surprising that none of the previous algorithms, i.e. neither the rate optimal nor the decoding delay optimal, perform consistently better for all ranges of parameters of our setup, as we show in the section devoted to simulation results.

In order to find an algorithm that takes into account both these sources of delay, we propose a scheme that is neither rate optimal nor decoding delay optimal but tries instead to build packets that generate delay on as few receivers as possible. It does this by adding a message in the linear combination only if the cost, measured in terms of delay introduced, is less than the gain, measured in delay removed.

The algorithms presented here incur very different decoding complexity to the receivers. In particular, the decoding delay optimal algorithms impose low complexity as the packets can be immediately decoded at the receivers, hence do not involve solving systems of equations. On the other hand, the rate optimal and the cost driven algorithms could be more computationally challenging but no worse than the systematic random network coding algorithm.

Finally, it should be clear that the heuristics presented here are still perfectible. The cost driven approach clearly shows that choosing algorithms that do not emphatically force rate and decoding delay optimality can be rewarding.

4.1 Standard and systematic random network coding

These two algorithms use the standard random network coding approach: at each time step they send a random linear combination of all messages. When the underlying field is large enough any M random vectors are linearly independent w.h.p., hence this transmission scheme is rate optimal w.h.p. Feedback information can be used to ensure rate optimality with smaller fields.

For the standard random network coding the expected delay of each receiver is approximately $M-1$: rarely messages can be decoded before the last packet is received.

The systematic random network coding algorithm differs from the standard one because it first sends all M messages uncoded. Thus each receiver receives an average of $(1-p)M$ uncoded messages, hence only needs to delay (i.e., is not able to decode) during approximately $pM-1$ time slots.

4.2 Decoding delay optimal algorithms

We present three different decoding-delay optimal algorithms, with increasing complexity:

Simple repetition algorithm This algorithm uses an approach similar to ARQ (Automatic Repeat reQuest) and operates in rounds. At the beginning of each round, the algorithm uses the feedback information to compute a set of messages Q that have not yet been received by at least one node; these messages are ordered according to some criterion (in our simulations we just send a random permutation of the messages) and are then sent uncoded during the round. The round will last $|Q|$ time slots. The algorithm ends when the computed Q is empty at the beginning of some round. This algorithm is not rate optimal for $N > 1$.

Random opportunistic algorithm This algorithm uses an opportunistic approach to improve the performance of the previous algorithm. Like COPE [10] it sends packets that can be immediately decoded by all receivers. Thus it never incurs decoding delay. However, it is not necessarily rate optimal since it might not always be possible to select a packet that is both innovative for all receivers and instantly decodable. Hence it might incur delay due to reception of useless packets.

The algorithm works in rounds and uses coefficients from $\mathbb{F}_2$. Like the simple repetition algorithm, at the beginning of each round it builds the set Q of messages that have not yet been received by at least one node. The algorithm then chooses each packet c to be sent as follows. First, it removes an element m_q from Q at random and sets $c = (e_q, m_q)$. Then it goes (in some order) over every message m_r that is still needed by some node. In particular, m_r might be a message that was already transmitted and removed from Q during the round but one of the nodes that needed it experienced an erasure during the transmission. Let v the coding vector associated to c and $\bar{v}$ the linear combination of messages associated to c. The algorithm sets $c = (e_r + v, \bar{v} + m_r)$ if $\bar{v} + m_r$ is decodable by all nodes that can decode c. Each time a packet is added to c the algorithm removes it from Q if it is present.

Highest delay first (HDF) opportunistic algorithm This algorithm operates similarly to the previous one but instead of choosing randomly the packet to be sent it always starts by choosing a message that is useful for the receiver that is experiencing the worst delay. To decide among the different messages that satisfy this condition it chooses the one that is required by most receivers. Then it selects the receiver with the highest delay that will not receive something useful and it chooses a message that can be added to the packet being sent without making it non decodable by some receiver. This procedure is repeated until no messages can be added to the packet. All ties are broken arbitrarily. This algorithm by choosing to help the receiver with the worst delay tries to reduce the worst case delay and by selecting the messages that are useful for the bigger number of receivers tries to minimize the number of receivers that will experience delay, thus reducing the median delay.

4.3 Rate optimal algorithm

This algorithm mimics the opportunistic algorithm but makes sure that each packet it sends is innovative for all receivers: after selecting c as in the random opportunistic algorithm it adds more messages, one at a time, until the packet contains for each receiver at least a message that is still not decoded. This ensures that it is possible to find a set of coefficients that make the packet linearly independent from what has already been received. The messages are then multiplied by coefficients from an appropriately large field, so that the final transmitted packet is innovative for every receiver. The coefficients are chosen randomly. If the resulting packet is linearly dependent on what has already been received by a receiver another set of coefficients is drawn. This procedure is repeated until the packet is linearly independent.

4.4 Cost driven algorithm

The cost driven algorithm is neither rate nor decoding delay optimal, but as we will see it has a median delay that is better than all other proposed algorithms. It works by first randomly selecting a set of messages that build a packet instantly decodable by all the receivers. It adds then more messages if adding them will create a packet that generate delay on less receivers. For instance assume that R_1 has received message m_1, R_2 has received m_2 and R_3 has received m_1 and m_2. The algorithm could select the packet $p_2 = ((110), m_1 + m_2)$ in the first phase, since it is instantly decodable by all receivers. This packet is not innovative for R_3 but the algorithm does not add m_3 because a packet $p_2 = ((111), m_1 + m_2 + m_3)$ would generate a delay on two receivers, which is worse than p_1. If however there would be two other receivers R_4 and R_5 that are in the same situation as R_3 the algorithm would send p_2 because in this case the overall delay would go from 3 to 2. The complexity of this algorithm is less than the one of an exhaustive search of the packet with the lowest delay because the initial instantly decodable packet is chosen at random and the successive messages are visited in a random order.

4.5 Empirical performance evaluation

In this section we discuss the performance of the algorithms presented previously as a function of M, N and p. Our graphs show the median delay $\mathcal{D}_m$ across all receivers. More specifically, to generate each graph, we run our algorithms many times (depending on the experiment from 10 to 30 runs). The outcome of each run is the median of the delay of the N receivers. Our plots show the median of these medians. We do not simulate the two random network coding algorithms; instead we use the theoretical approximations for their performance.

4.5.1 Linearity of delay as a function of M

Figure 4.1 and 4.2 show the performance of a selection of algorithms. The different lines show, in Figure 4.1 the performance with different values of N when $p = 0.5$, and in Figure 4.2 the performance with different values of p when $N = 100$. The same behavior is present with other values of $N < 100$ and $p > 0.1$ not illustrated in the graphs. The delay of the two random network coding algorithms is not illustrated but, for fixed p and N, it is approximately $\mathcal{D}_m(M) = M - 1$ for the standard version and $\mathcal{D}_m(M) = pM - 1$ for the systematic version. The plots clearly show that the delay of the algorithms is linear in the number of messages. The solid line indicates a least squares approximation of data. Most points are very close to this approximation. These experiments suggest that, for the selected

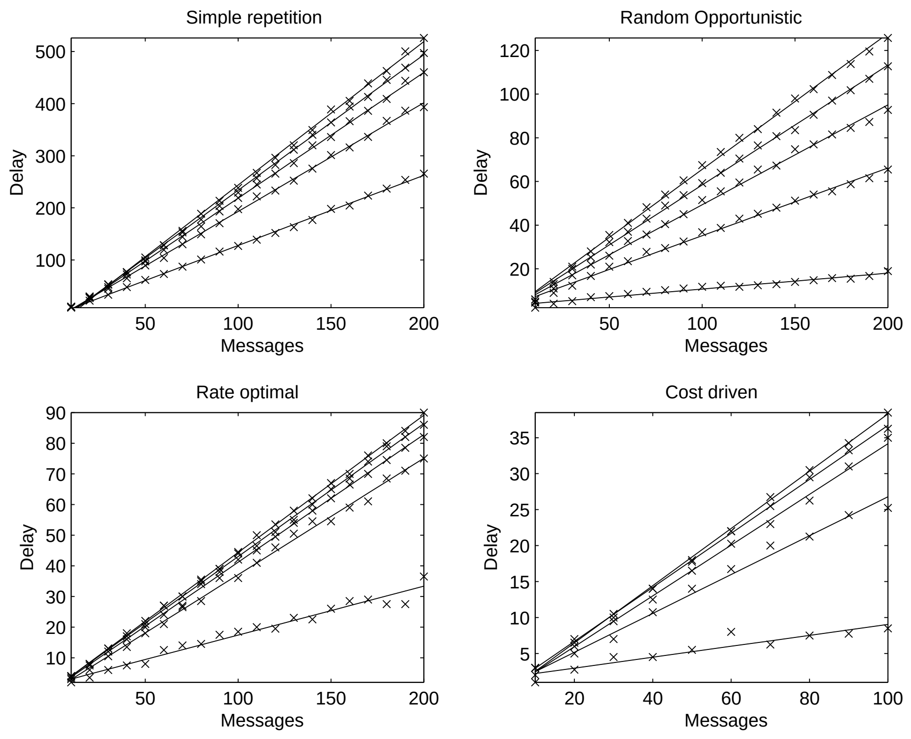

Figure 4.1: Delay as a function of M ($p = 0.5$). The lines are the least squares linear approximations, the crosses are the actual data points. The plots show the function for $N = 10, 30, 50, 70, 90$ (higher N is the line with steeper slope)

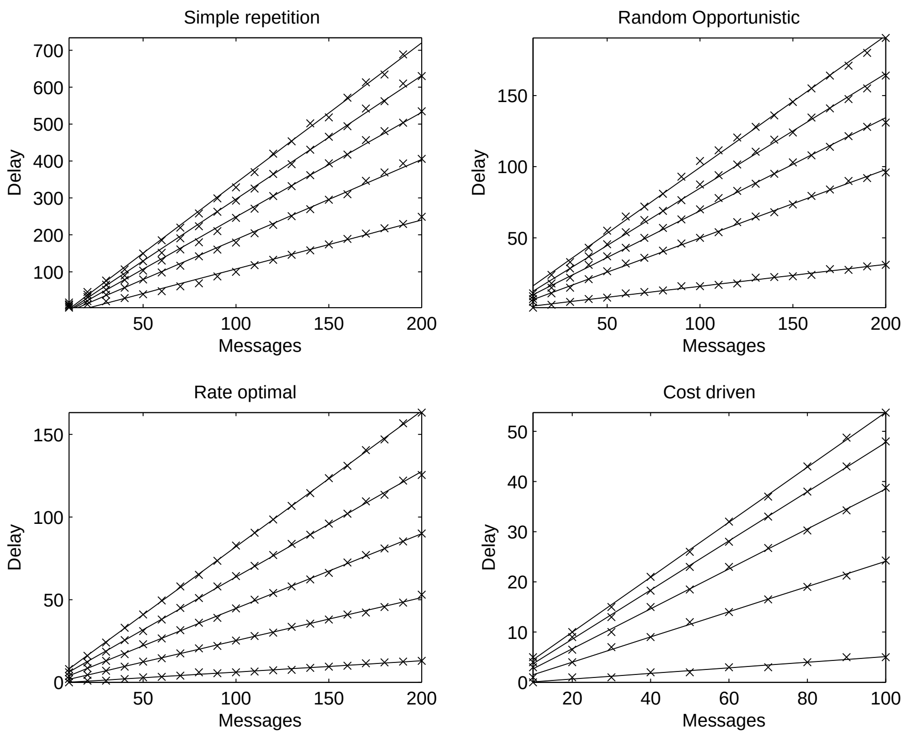

Figure 4.2: Delay as a function of M $(N = 100)$. The lines are the least squares linear approximations, the crosses are the actual data points. The plots show the function for $p = 0.1, 0.3, 0.5, 0.7, 0.9$ (higher p is the line with steeper slope)

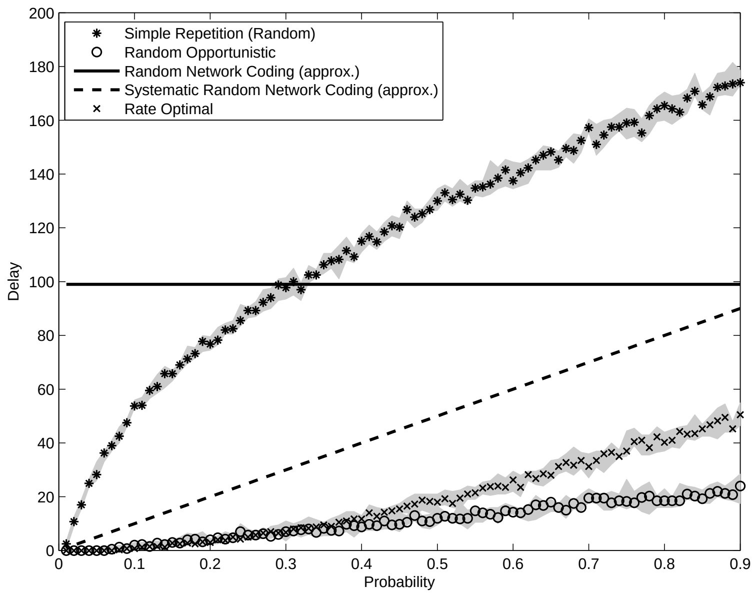

Figure 4.3: Median delay with $N = 10$ (gray area is the 95% confidence interval, $M = 100$)

algorithms, there is a fixed per-message delay independent on the number of messages. The fact that the number of message don't affect the relative performance of the different algorithms may imply that what is the optimal algorithm depends only on p and N.

4.5.2 Delay as a function of p

Figures 4.3, 4.4 and 4.5 show the performance of some of the proposed algorithms as a function of the probability of erasure p. We can see that all algorithms, except the standard random network coding, have a delay that tends to zero when the probability of erasure tends to zero. The performance of the standard random network coding is constant and the performance of the systematic version is linear. Surprisingly for the random opportunistic algorithm the delay is not always increasing.

When the number of receivers is small (Figure 4.3) the performance of both rate optimal and random opportunistic algorithms is better than the performance of the systematic random network coding algorithm. The best algorithm is the random opportunistic algorithm. This indicates that when the number of receivers is small it appears it is better to send non-innovative packets that are instantly decodable than ensuring rate optimality.

When the number of receivers increases (Figure 4.4) the performance of

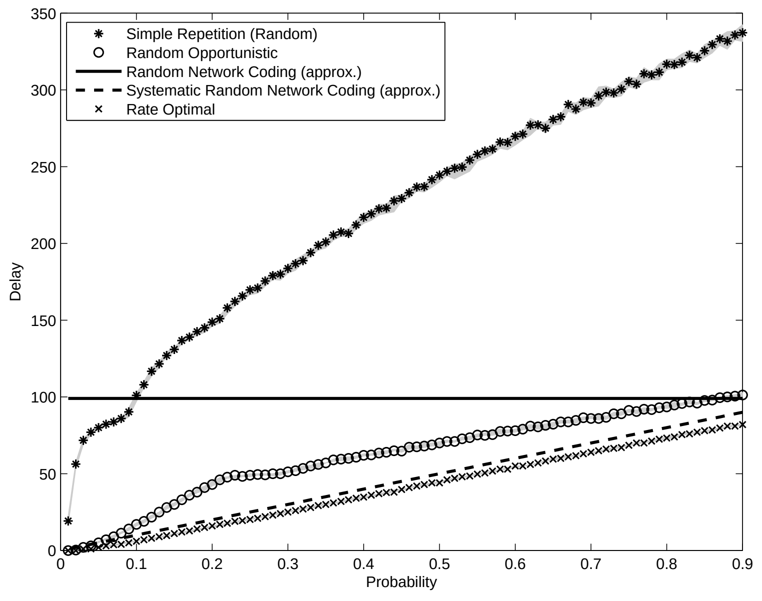

Figure 4.4: Median delay with $N = 100$ (gray area is the 95% confidence interval, $M = 100$)

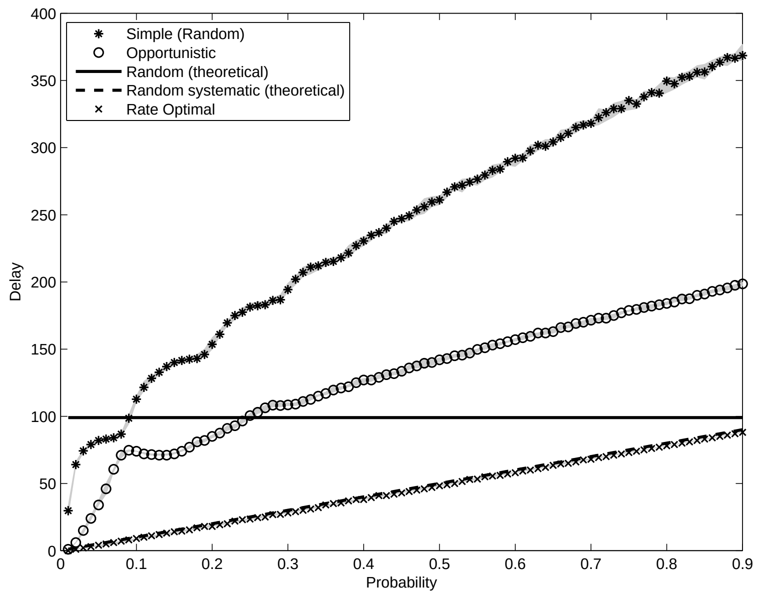

Figure 4.5: Median delay with $N = 1000$ (gray area is the 95% confidence interval, $M = 100$)

the random opportunistic algorithm becomes worse than the performance of the systematic random network coding algorithm. The best algorithm in this case is the rate optimal one. Sending non innovative packets seems to generate delay on too many receivers, so it's better to send rate optimal packets. It is interesting to see that around $p = 0.2$ there is a change in the slope of the performance of the random opportunistic algorithm. The simple repetition algorithm shows two rapid changes in slope before $p = 0.1$. Such behaviors are even more apparent in the next plot.

For $N = 1000$ (Figure 4.5) the performance of the rate optimal algorithm and the systematic random network coding are indistinguishable, the rate optimal algorithm sends linear combinations that are not decodable by most of the receivers, exactly like the other algorithm. When p is small the delay of the simple decoding delay optimal algorithm increases rapidly and then has an almost a constant value, such behavior is repeated some times with decreasing intensity. The random opportunistic algorithm has a delay that is no more always increasing: when $p \in [0.1; 0.15]$ it is decreasing.

The peculiar form of the performance of the simple decoding delay optimal algorithm is probably due to to the fact that when the number of receivers is large the algorithm is sending in the second round (or in the successive ones) all the messages because it is very likely that at least one receiver has not received one of them. Each receiver is waiting for some messages (on average Mp in the second round) therefore it receives many useless messages, being p still quite small.

The shape of the random opportunistic algorithm performance is probably explained by similar reasons. From our observations we discovered that most of the packets containing more than one message are sent during the last rounds. Depending on p and N there can be more or less receivers in these rounds. When $p = 0.1$ and $N = 1000$ conditions are such that there are so few opportunities of linear encoding that the algorithm performs similarly to the simple version. It would be interesting to further study the inner working of the algorithm and precisely understand what happens.

4.5.3 Delay as a function of N

Figures 4.6, 4.7 and 4.8 show the performance of a selection of algorithms as a function of N. The different plots show the delay at some values of p. The performance of the random network coding algorithms is constant, they are FEC codes. The delay of all other algorithms is 0 when $N = 1$ and then increase with N. The random opportunistic algorithm achieves a big improvement over the simple repetition algorithm. The shape of all algorithms is not affected by p but the relative performance of the algorithms changes.

The delay of the simple repetition algorithm is already large at $N = 2$, this is explained by the fact that, even with two receivers, in each round

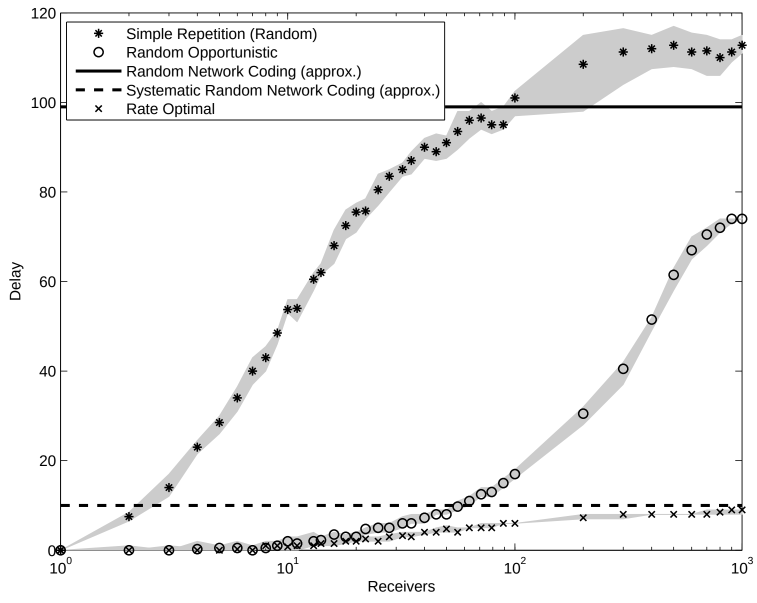

Figure 4.6: Median delay with $p = 0.1$ (gray area is the 95% confidence interval, $M = 100$)

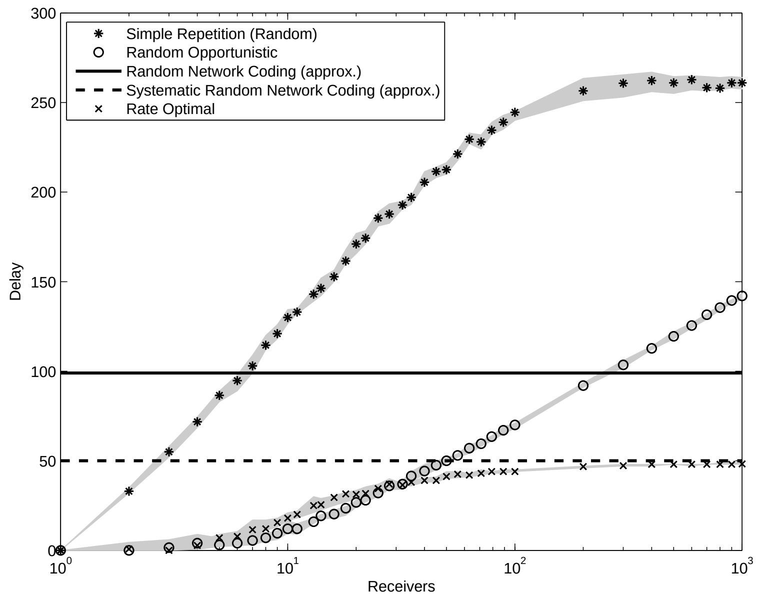

Figure 4.7: Median delay with $p = 0.5$ (gray area is the 95% confidence interval, $M = 100$)

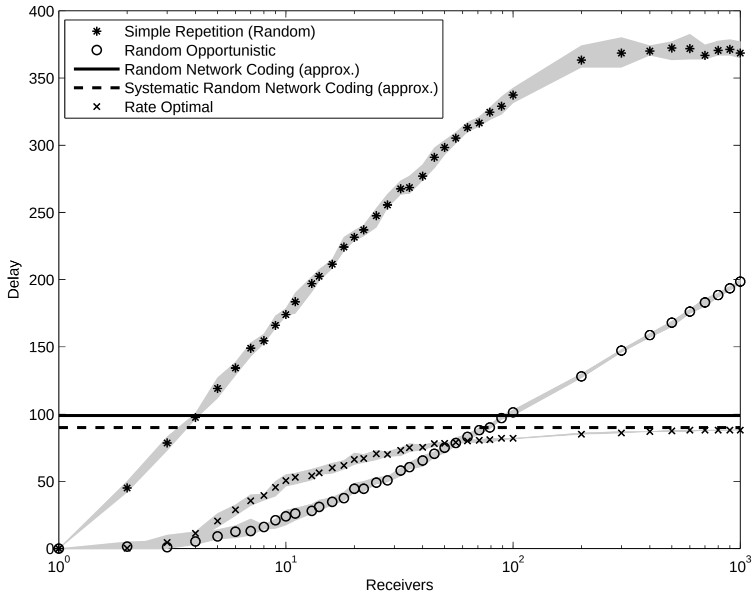

Figure 4.8: Median delay with $p = 0.9$ (gray area is the 95% confidence interval, $M = 100$)

many messages are useless for one of the receivers. The performance of the rate optimal and the random opportunistic algorithm are zero even for $R > 2$. If few packets are erased and there are few receivers it is easy to find innovative and instantly decodable packets. Both algorithms in this case send the same packets. As soon as the number of receivers increases the rate optimal one cannot send instantly decodable packets anymore, it then sends different packets from the other and their performance diverges.

When the number of receivers is large all algorithms reach an asymptote: since it's very likely that at least one sender has not received each of the messages decoding delay optimal algorithms often send all of them for many rounds. The number of rounds that a node has to listen doesn't depend on the number of receivers but on the probability of erasure. Therefore the overall performance is no more determined by the number of receivers. The opportunistic algorithm has a better performance because it generates less delay in the final phase where it is possible to find messages that can be linearly encoded. The rate optimal algorithm behaves like the systematic random network coding one because when the number of receivers is large it is forced to send packets that cannot be decoded until a receiver collects M of them.

In Figure 4.6 we see when p is small that the best algorithm is always the rate optimal one. In Figure 4.7 we see that with a higher probability of

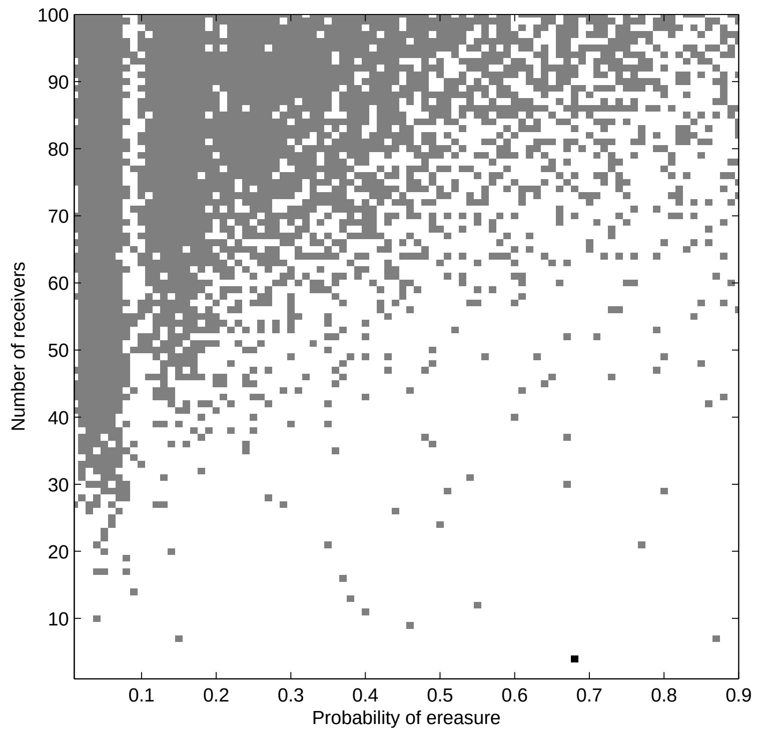

Figure 4.9: Comparison between random and heuristic simple ARQ (gray indicates heuristic is better at 90% confidence, white indicates that the performance is similar)

erasure when N is small it is better to use the opportunistic algorithm. Only when $N > 20$ the rate optimal is better. We can notice that the asymptote for the opportunistic algorithm is better than the random network coding when p is small, but as the erasure probability increases it is worse.

Our simulations show that as N grows, the proposed feedback algorithms are not the best choice: although the rate optimal performs the best, the complexity of keeping track of the state of the receivers is too high. Systematic network coding should be preferred since it performs comparably and does not use feedback. On the contrary, for small values of N, our proposed feedback algorithms constitute a viable solution.

4.5.4 Using an heuristic in the simple random ARQ

In this paragraph we look at how the performance of the simple repetition algorithm changes if we use a more sophisticated criterion to choose which message to send in each packet. We compare here the performance of our simple repetition algorithm with the performance of an algorithm that al-

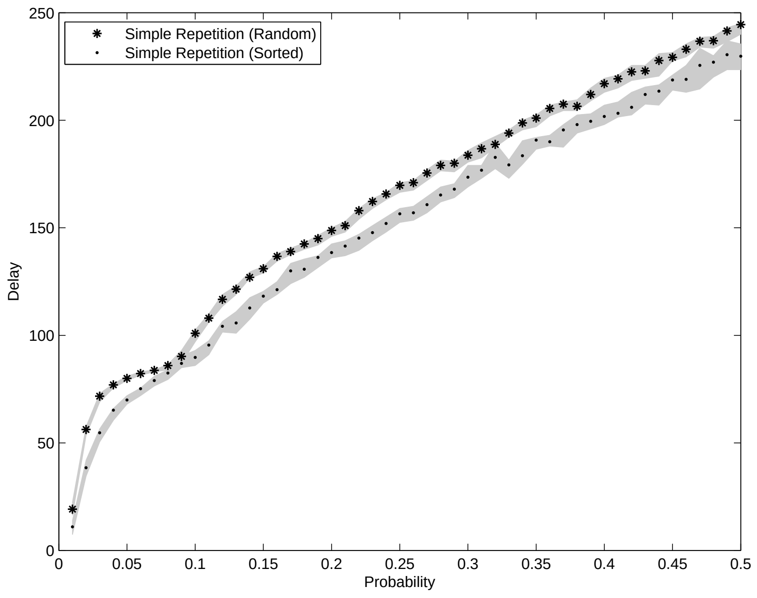

Figure 4.10: Median delay with $N = 100$ (gray area is the 95% confidence interval, $M = 100$)

ways sends the message required by more receivers. Figure 4.9 shows that the performance of the two algorithms is comparable when the probability of erasure is low and the number of receivers is small. In this region most of the messages that have to be retransmitted after the first round are required by the same number of receivers thus they are sent in a random order in both algorithms. When the number of receivers is large, there is more benefit in sending the messages in order, but as we see in Figure 4.10 that depicts the performance when $N = 100$ the difference in delay is marginal.

4.5.5 Comparison between random and HDF opportunistic algorithms

The relative performance of the two proposed opportunistic algorithms can be seen in Figure 4.11. We can notice that using a more sophisticated heuristic helps when the probability of erasure is low. On the contrary when the number of erasures and receivers is high the random algorithm performs better. The explanation of this behavior could be that when the probability of erasures is small it is possible to do a better job by choosing carefully which receiver to help since it is quite likely we will actually get the packet through to it. When instead the probability of erasures is high, the packet, which may have negative effects on many receivers, is likely not to

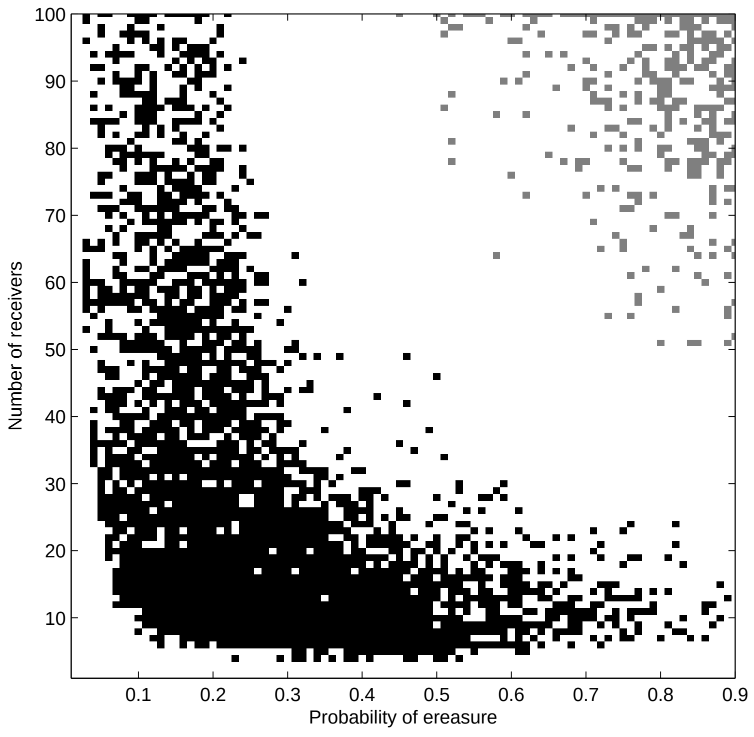

Figure 4.11: Comparison between random and HDF opportunistic algorithms: black indicates HDF is better, gray indicates random is better (at 90% confidence), white indicates that the performance is similar

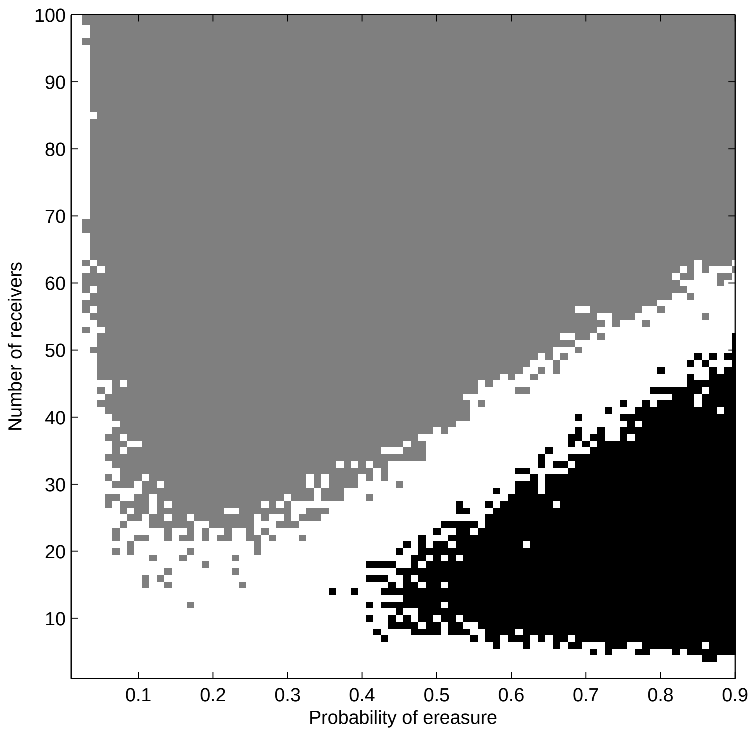

Figure 4.12: Comparison between random opportunistic and rate optimal algorithms: gray indicates rate optimal is better, black indicates opportunistic is better (at 90% confidence), white indicates that the performance is similar

get through to the receivers that has to be helped. Thus the median delay is actually worse than when choosing messages at random. Further study of this behavior could be interesting.

4.5.6 Comparison between random opportunistic and rate optimal algorithms

As we have already shown in section 4.5.2 and 4.5.3 depending on p and N the rate optimal or the random opportunistic algorithm have the best performance. In Figure 4.12 we can see precisely where these regions are. When the number of receivers is large rate optimal is better than opportunistic. Apparently in that situation the delay that is generated by non innovative packets in the opportunistic dominates the decoding delay of the systematic random network coding algorithm towards which the rate optimal algorithm converges. When the probability of receiving a packet and the number of receivers are low the opportunistic algorithm performs better, this is probably

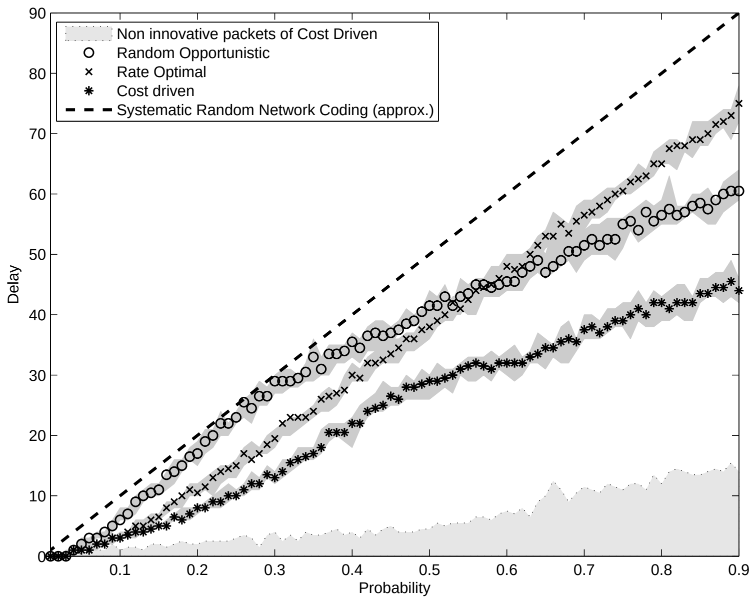

Figure 4.13: Median delay with $N = 35$ (gray area around data points is the 95% confidence interval, $M = 100$)

due to the fact that in such circumstances sending randomly chosen innovative packets creates delay on almost all successfully receiving receivers while with instantly decodable packets at least some receivers might not experience delay.

4.5.7 Performance of the cost based algorithm

In this last section we analyze the performance of the cost driven algorithm. Such algorithm is motivated by the fact that neither rate optimal nor decoding delay optimal algorithms performs consistently better. This algorithm adds messages to packets only if this doesn't increase the overall delay generated by the packet. We can see in Figure 4.13 that such approach is rewarding when $N = 35$. The same has been seen to hold as well for all other values $N < 100$. This algorithm outperforms both other approaches, thus is better than a simple combination of them.

Figure 4.14 shows that the performance of the cost driven algorithm is better than any of the other two algorithms with varying numbers of receivers. From the plot it is not possible to determine if the performance of the cost driven approach is better than the one of the FEC scheme because at $p = 0.5$ even the rate optimal algorithm doesn't reach its asymptote until $N = 1000$ (see Figure 4.7). The asymptotical behavior of this algorithm

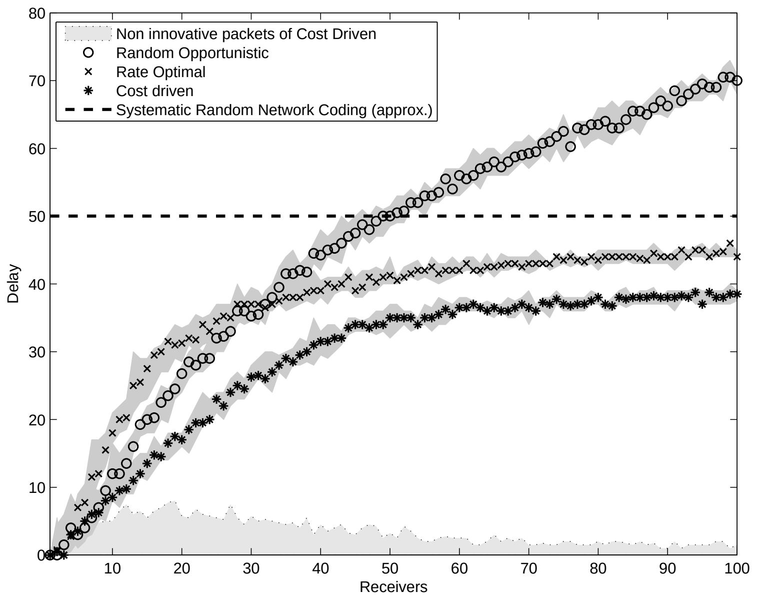

Figure 4.14: Median delay with $p = 0.5$ (gray area around data points is the 95% confidence interval $M = 100$)

should be further studied.

In both Figures 4.14 and 4.13 we show the part of delay due to non innovative packets for the cost driven algorithm. If such part is near zero it means that the algorithm is almost rate optimal. We can see that when the number of receivers increases the algorithm is operating near to rate optimality while when the probability of erasure is high it is operating well below rate optimality. This is consistent with our observations so far that rate optimality is not useful for small N and high p while it is useful for high N. This could give some insight into the behavior of the optimal algorithm. However we should keep in mind that all three algorithms choose packets among the ones that satisfy their criteria randomly. Therefore it could be possible, for example, that the optimal scheme is rate optimal, but our rate optimal algorithm is missing most of the time the best packets just because they are rare in the set of all rate optimal packets that can be sent.

Chapter 5

Conclusions

In this report we studied the problem of broadcasting M packets to N receivers over independent erasure channels with perfect feedback and source coding using rate optimal transmission schemes. We presented the optimal offline algorithm in the case of $N = 3$ receivers that achieves zero delay. In contrast, we showed that for $N = 4$, there exists an erasure pattern that imposes a total delay of at least $M/2$ to the receivers under any rate optimal transmission scheme. Then we investigated some online algorithms, and compared their performance to standard algorithms via simulations. Those algorithms can achieve significant improvements in delay compared to standard FEC codes. We found that the best performance among the collection of algorithms presented is achieved by a schema that is neither rate optimal nor decoding delay optimal.

The main theoretical questions that are still open are finding the optimal offline algorithm for the general case of N receivers (or proving that this problem is NP-hard). Other interesting challenges include the design and the analysis of more sophisticated online algorithms, and further comparison of rate optimal with non-rate optimal transmission schemes in order to understand the interplay between delay due to non-innovative packets and decoding delay. An interesting point would be to analyze the performance of the cost driven algorithm presented in this report when the number of receivers is large. It could be as well interesting to analyze the structure of the broadcast schedules generated by the algorithms, to better understand the performance results.

In this report the performance is optimized to have high rates. The algorithms presented cannot be used to do real time streaming of information because no care is taken of the order the messages are delivered. The first part of a message could be delivered as last on some receivers. Many applications don't require such a constraint: in many cases the information cannot be used until all messages are available to the receivers therefore the order in which they are sent is irrelevant. It could be however interesting to analyze algorithms that optimize some delay metric that embeds the concept of order of delivery.

Appendix A

Incremental decoding of linear combinations of messages

In this section we show some results that prove that solving incrementally a system of equations given by the incoming packets with the standard Gauss-Jordan elimination method decodes messages as soon as possible and the overall worst case complexity is not worse than solving the system with the same method but waiting until M packets are received.

Definition A.0.1. *Let p(t) the packet received at time t, let $p_B(t) = \{b_1, \ldots, b_m\}$ the messages linearly combined in $p(t)$ and $p_c(t) = c_1, \ldots, c_m$ the corresponding coefficients. Define $D(0)$ a 0×0 matrix, and D(t+1) with $t > 0$ the matrix created by the following algorithm where $b_D(0, \cdot) : \varnothing \to \varnothing$ and $b_D(t, \cdot)$ is a mapping from the set of columns of $D(t)$ to $\bigcup_{t=1}^{t-1} p_B(t)$:*

1. *$D = D(t)$*
2. *$B_{\text{new}} = \bigcup_{t=1}^{t-1} p_B(t) - p_B(t)$*
3. *Augment the matrix D with $|B_{\text{new}}|$ empty columns*
4. *Define $b(i)$ such that $b(i) = b(t, i)$ if column i existed in $D(t)$; otherwise set $b(i) = b$ with $b \in B_{\text{new}}$ such that if $b(u) = b(v) \Rightarrow u = v$.*
5. *Append to the matrix D a new empty line k*
6. *Set the $D_{k,i} = c_j$ for all $b_D(i) = b_j$*
7. *$\forall 0 \leq i < k$ $D_{k,\cdot} = D_{k,\cdot} - D_{k,i} \cdot D_{i,\cdot}$*
8. *If $\{j | D_{k,j} \neq 0\} = \varnothing$, delete row $D_{k,\cdot}$ and stop algorithm*
9. *Find $p = \min\{j | D_{k,j} \neq 0\}$, swap column $D_{\cdot,p}$ with $D_{\cdot,k}$ and $b_D(p)$ with $b_D(k)$*
10. *$D_{k,\cdot} = 1/D_{k,k} \cdot D_{k,\cdot}$*

11. $\forall 0 \leq i < k \; D_{i,\cdot} = D_{i,\cdot} - D_{i,k} \cdot D_{i,\cdot}$

12. $D(t+1) = D$, $b_D(t+1,\cdot) = b_D(\cdot)$

Lemma A.0.1. *Matrix* $D(t) = (I|R(t))$

Proof. Proof by induction: $D(0)$ satisfies the conditions. Matrix $D(t+1)$ is constructed by adding trailing columns (that don't change the property) and by adding a row. At step 7 we make sure that in the first $\text{rows}(D(t+1)) - 1$ columns the new line contains only zeros (using the standard gaussian elimination procedure); at steps 8 and 9 we make sure that $D(t+1)_{\text{rows}(t+1),\text{rows}(t+1)} = 1$, and at step 9 we make sure that the column $\text{rows}(D(t+1))$ is of the correct shape (using Gauss-Jordan elimination). □

Lemma A.0.2. *The space generated by the lines of matrix $D(t)$ is the same as the one generated by the coefficients of the packets received up to time $t-1$*

Proof. Lines of the matrix $D(t)$ are created by linearly combining coefficients of the packets. By construction the dimension of the space is preserved (because the line are either multiplied by a non zero term (step 10) or substituted with a linear combination of itself and another line (steps 7 and 11)). □

Lemma A.0.3. *No message in set* $\{b | \exists i > \text{rows}(D(t)), b(i,t) = b\}$ *can be decoded with packets* $p(0), \ldots p(t-1)$

Proof. Let assume a message b such that $b(i,t) = b$ with $i > \text{rows}(D(t))$ can be decoded. This would mean that $e_i \in \text{span}(P_C) = \text{span}(L)$ and therefore that would mean that e_i could be expressed as:

$$e_i = \sum_{l_u \in L} a_u \cdot l_u \Leftrightarrow e_i = \sum_{l_u \in L} c_u (e_u + (0_{1\times m} | R_{u,\cdot}) = \sum_{l_u \in L} c_u e_u + \sum_{l_u \in L} c_u (0_{1\times m} | R_{u,\cdot})$$

The first part of the expression forces all c_u to be 0 and the second part forces at least one c_u to be different than 0 that is a contradiction so no such message can be decode. □

Theorem A.0.1. *A message b can be decoded using packets $P = p(0), \ldots,$ $p(t)$ if and only if there exists a line j of matrix $D(t+1)$ such that $D(t+1)_{j,k} = 1$, $b(t+1,k) = b$ and $D(t+1)_{j,l} = 0 \forall l \neq k$.*

Proof. Let $P_C = p_C(0) \ldots p_C(t)$ the coefficients of the packets. We know that a message b can be decoded using P if and only if $e \in \text{span}(P_C)$ with e the unit vector associated to b. If there is a row $l_u = e_v$ then it's trivial to see that e_v is in the span of L and therefore that that message $b(u,t+1)$ can be decoded (by using the standard gauss-jordan method).

To show the converse we know from the previous lemma that $D(t+1)$ can be decomposed in $(I|R)$. Let $m = \text{rows}(D(t+1))$ and $n = \text{cols}(D(t+1))$. Let i such that $b(t+1,i) = b$. The unit vector corresponding to b expressed with the notation used in matrix $D(t+1)$ is e_i.

We know from the previous lemma that $i \leq m$. We will show that $l_i = e_i$ We know that $e_i \in \text{span}(L)$ therefore:

$$\begin{aligned}
e_i = \sum_{l_u \in L} a_u \cdot l_u \quad & \Rightarrow \quad l_i = \sum_{l_u \in L - \{l_i\}} c_u l_u + d \cdot e_i \\
\text{(lemma 1)} \quad & \Rightarrow \quad e_i + (0_{1\times m}|R_{u,\cdot}) = \sum_{l_u \in L - \{l_i\}} c_u (e_u + (0_{1\times m}|R_{u,\cdot})) + d \cdot e_i \\
& \Rightarrow \quad e_i = \sum_{l_u \in L - \{l_i\}} c_u e_u + d \cdot e_i \\
& \Rightarrow \quad c_u = 0 \forall u \\
& \Rightarrow \quad R_{u,\cdot} = \sum_{l_u \in L - \{l_i\}} c_u R_{u,\cdot} = 0
\end{aligned}$$

□

Theorem A.0.2. *If the received packets are always innovative the number of arithmetic operations required to build the matrix $D(t+1)$ is $O(tM)$ with M the number of existing messages.*

Proof. The number of rows of matrix $D(t)$ is $t-1$. The number of columns of the matrix is upper bound by M. At step 7 we do $O(tM)$ operations, at step 10 we do $O(M-t)$ operations, at step 11 we do $O(tM)$ operations. The overall complexity is therefore $O(tM)$. □

Corollary A.0.1. *The overall complexity of decoding in terms of arithmetic operations is $O(M^3)$*

Bibliography

[1] M. Durvy, C. Fragouli, P. Thiran, “Towards reliable broadcasting using ACKs”, ISIT 2007.

[2] C. Fragouli, E. Soljanin, “Network Coding Fundamentals (Foundations and Trends in Networking)”, Now Publishers Inc, 2007

[3] A. Shokrollahi, “Raptor Codes”, IEEE Trans. Inf. Theory, vol. 52, pp. 2551–2567, 2006.

[4] M. Luby, “LT Codes”, Proceedings of the 43rd Symposium on Foundations of Computer Science, p.271, 2002.

[5] Y. Wu, P. A. Chou, S.-Y. Kung, “Information exchange in wireless networks with network coding and physical-layer broadcast”, CISS 2005.

[6] Cullen, J., “Binary Synchronous Communications”, Communications, IEEE Transactions on (legacy, pre - 1988), vol.17, no.6, pp. 654–658, 1969.

[7] W. Mao, “Competitive analysis of on-line algorithms for on-demand data broadcast scheduling”, ISPAN 00.

[8] J. Massey and P. Massey, “Zero error with feedback”, online tutorial.

[9] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain and L. Tolhuizen, “Polynomial time algorithms for multicast network code construction”, IEEE Trans. Inform. Theory, vol. 51, iss. 6, pp. 1973–1982, 2005.

[10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard and J. Crowcroft, “XORs in the Air: Practical Wireless Network Coding”, Sigcomm 2006.