Real-time Delay with Network Coding and Feedback

Eleni Drinea[a], Lorenzo Keller[a,*], Christina Fragouli[a]

[a] *École Polytechnique Fédérale de Lausanne (EPFL), CH-1015, Lausanne, Switzerland*

Abstract

We consider the problem of minimizing delay when broadcasting over erasure channels with feedback. A sender wishes to communicate the same set of μ messages to several receivers. The sender can broadcast a single message or a combination of messages at each timestep, through separate erasure channels. Receivers provide feedback as to whether the transmission was received. If, at some time step, a receiver cannot identify a new message, delay is incurred. Our notion of delay is motivated by database replication. Our setup is novel in that it combines coding techniques with feedback information to the end of minimizing delay. We show that it allows $\Theta(\mu)$ benefits as compared to previous approaches for offline algorithms, while feedback allows online algorithms to achieve smaller delay compared to online algorithms without feedback. Our main complexity result is that the offline minimization problem is NP-hard both under scheduling and coding algorithms. However we show that coding does offer delay and complexity gains over scheduling. We also discuss online heuristics and evaluate their performance through simulations.

Keywords: Network Coding, Broadcast, Feedback, Complexity, Delay

1. Introduction

Current and emerging applications, such as traffic-aware navigation, vehicle fleet management or real-time parking availability information, could greatly benefit from a database replication mechanism that allows to quickly and reliably disseminate information from a central host over possibly unknown channels. In practical networks, transmissions are subject to errors: packets get dropped due to congested links, wireless fading and interference, expired timestamps, etc. Such losses are perceived as packet erasures at higher layers, and are often modeled using independent erasure channels.

To cope with unknown channels, feedback information is often available at the broadcasting source. Thus the source, when deciding what to transmit next, knows which subset of receivers successfully received each of its past transmissions. Feedback can be efficiently employed in a wireless environment: the source might acquire such information by taking advantage of the symmetry of wireless links, or by collecting acknowledgment packets explicitly using specifically designed control traffic [1], or implicitly, by overhearing transmissions from the receiver nodes [2]. In satellite transmissions, a satellite might learn when a receiver goes in a deep fade (e.g., enters a tunnel), in which case it loses a sequence of packets. A similar approach to explicitly collect acknowledgments in wired networks, when the source multicasts the same content over a distribution tree in an overlay network appears in [3].We will assume in this paper that the source has perfect feedback information.

[☆] This work was partially funded by ERC Project NOWIRE (ERC-2009-StG-240317)

[*] Corresponding author

Email addresses: edrinea@gmail.com (Eleni Drinea), lorenzo.keller@epfl.ch (Lorenzo Keller), christina.fragouli@epfl.ch (Christina Fragouli)

In this paper, we consider the problem of combining coding techniques and feedback information over broadcasting channels to offer reliable database replication under delay guarantees. In our setup the database is created at a central location and then broadcasted to a set of devices that want to use it. The database is divided in μ partitions, that are the messages to be transmitted. Devices are interested in receiving the messages as soon as possible to be able to start using the contained data. Notice that messages are useful even if not all of them have already been received yet. For instance traffic data for a set of roads can be used to adapt the current route as soon as it is received.

An alternative motivation for our setup, not discussed in this paper, is when some data that has been encoded with a generic multiple description code must be delivered[4]. In this setup it is also useful to decode as soon as possible new descriptions. Notice however that if it is possible to select the multiple description code our analysis is not necessary [5].

We assume that every message is drawn uniformly and independently therefore there is no gain in performing source coding. If this is not the case for the data being transmitted then there will be some rate penalty with the presented architecture, in that case a judicious usage of source coding could lead to significant improvements. For instance simply compressing the different database partitions independently could already bring significant benefits while retaining the same delay characteristics of the approach being studied in this paper.

When communicating towards a single receiver, simple sequential transmission of the messages suffices: using automatic repeat request (ARQ) it is possible to efficiently transmit the messages and, since they are not coded, as soon as they are received they can be used.

The problem becomes much more challenging when the database needs to be broadcasted to a number of receivers, each of which receives information over its own erasure channel. The sender may use a *scheduling* algorithm to decide which partition to broadcast next. In this paper, we propose instead to use a *coding* algorithm, that encodes the partitions we need to transmit to the receivers. Both in the case of scheduling and coding, the algorithm may decide on the current transmission by using the feedback information it has collected, i.e., which receivers received the previous transmissions. Note that our proposed coding is not a source coding scheme: the partitions are transmitted as they are. Instead, it falls in the area of network coding (see [6, 7] and [8] for introductory tutorials), as its main purpose is to better share the network resources among the contending receivers.

In order to reconstruct the whole database, every time a receiver receives, it wants to learn some missing piece of information, namely *any* partition it does not know yet. This motivates us to increment the delay d_j of receiver r_j by one every time r_j successfully receives a transmission of the following type: (i) a partition r_j already knows, or (ii) an encoding of partitions which, when combined with r_j's successful receptions so far, does not allow r_j to extract some partition it does not know yet. This definition allows us to disengage delay from the erasure frequency as we only count delay when a transmission is successful. It also allows us to capture two causes of delay: delay due to useless received packets, namely packets that bring duplicate information to their receiver, and delay due to packets that, although useful, do not allow their receiver to decode some unknown message upon their reception. Finally, our definition of delay is the simplest instantiation possible, as it does not take into account any ordering. We thus hope that a good understanding of this problem can serve as a first step towards more combinatorially demanding delay definitions.

The main questions we consider in this paper are (i) whether coding offers benefits in terms of delay, and (ii) how to design coding schemes that minimize average and maximum delay, and what is the complexity of this task. We both consider the offline problem, i.e. when all channel realizations are known in advance and its online counterpart, where each channel realization is known only after the channel has been used. Our primary motivation for investigating the offline scenario is that it could provide a lower bound for the online

scenario: if the complexity of the offline problem is tractable, then the performance of the optimal offline algorithm gives a (efficiently computable) lower bound on the performance of any online algorithm. This fact is obviously relevant to our application: prior to our offline complexity analysis, one could hope to design such a polynomial time algorithm and obtain such optimal lower bounds. Our work shows that this is not the case.

We focus on the case where all receivers are interested in the same content and believe that this simple model will provide insight for variations where receivers may demand different subsets of the messages or request the messages in a specific order.

Notice that in our setup receivers are interested in consuming exact reconstructions of the partitions of the database, and not approximations, such as successively refined versions of the database. For this reason the scenario studied in this paper doesn't lend itself to the usage of multiple description codes[9]. It is also worth noting that the popular solution of employing rate-less erasure correcting codes at the source such as LT or Raptor codes [10, 11] for reliable broadcasting over erasure channels, performs very poorly in terms of delay (see also Subsection 1.1).

Our contributions include the following. Concerning the complexity of the offline problem, we show that minimizing the average and maximum delay when the source uses scheduling is NP-hard. We then examine the complexity of the problem when coding is allowed and show that, although specific classes of erasure instances become trivial, the general problem remains NP-hard. We examine classes of erasure instances where coding offers significant benefits in terms of delay, and give a simple inapproximability result for maximum offline delay. Finally, we discuss heuristic online algorithms where the erasures of different receivers are independent and i.i.d. distributed. We evaluate the performance of our heuristics through simulations. The latter verify our observation that coding can significantly reduce delay compared to scheduling.

The importance of our work lies perhaps in that, to the best of our knowledge, it was the first to examine the complexity and algorithmic aspects of the joint use of coding and feedback information for delay-optimal content delivery. Erasures are inherent in many realistic networks, and we believe that the trade-off between rate and delay that arises in our setting is worth exploring further.

The remainder of this paper is organized as follows. Section 2 introduces our model and notation. Section 3 examines the complexity of offline broadcasting with scheduling, while Section 4 examines the complexity when coding at the source is allowed. Section 5 discusses online results. Section 6 concludes the paper.

1.1. Related Work

A significant body of work has investigated the problem of scheduling user requests over a broadcast medium to maximize the per-user received rate and minimize the response time; see for example [12, 13, 14, 15]. In this setup, users typically arrive at different time instances, and ask for possibly different content. No coding is employed and no errors are assumed. The difficulty of the problem, which was recently shown to be NP-hard [12], arises from having to share the common medium over the contenting requests. It is worth mentioning that if receivers were to ask for the *same* data items (e.g., satellite images) in *any* order, which is the case for our broadcasting scenario, then even if the requests arrive at different time instances, a periodic (circular) transmission of the data items would suffice.

In uncoded transmissions, maximizing the per user throughput naturally minimizes the delay: the faster information is received, the better. However, in the presence of erasures, uncoded transmission leads to repetitive reception and cannot achieve rates close to the optimal, and thus neither can it achieve the optimal delay. On the other hand, when coding is employed, delay and rate may become conflicting requirements. For rate-less codes for

example, to operate close to capacity and avoid duplicate receptions, we need to encode at the source across μ packets, for large values of μ [10, 11]. A receiver needs to wait to collect $\Theta(\mu)$ coded packets to be able to decode, which implies a delay of $\Theta(\mu)$. Indeed, in the presence of erasures, satisfying requests even for the same content becomes a challenging problem [16]. In [17] use of MDS codes has also been proposed, but their performance is inferior to Raptor codes both in terms of complexity and adaptability to unknown channel conditions.

Our work can also be viewed as an instantiation of network coding with feedback. Recent work has looked at use of acknowledgments and coding to optimize the achievable rate, under the condition that each received packet is either useless or can be immediately decoded by the destination [2]. The impact on delay of such schemes has been extensively studied [18, 19, 20, 21]. In our work we relax the constraint on immediate decodability and we let the source send packets that are not decodable by all receivers. This type of coding has been studied in other work but using a different delay metric, i.e. the interval between the moment a message is first combined in a packet and the moment it is decoded. [22, 23] study the problem in a time division duplex setup and [24, 25] study the network case. This delay metric is important for in-order delivery of messages but is not well suited to the scenario discussed in this paper.

Another line of work has looked into use of coding and feedback to minimize the queue size at the sender [26]. This performance metric is quite different from delay. Also, [27] examines use of feedback over broadcast erasure channels to optimize rate and achieve zero probability of error.

A related broadcasting scenario, called Index Coding, was introduced in [28] and examined in a line of works [29, 30, 31]. In the more general setting of [31], there is a source with μ messages and there are ρ receivers. It is assumed that by time t each receiver knows some subset of the μ messages (its *side information*), no erasures occur after time t, and each receiver wants exactly one of the blocks it is still missing. The goal is to find the minimum length of the codeword whose transmission will allow all receivers to simultaneously recover their missing blocks. In our setting, the assumptions above do not hold. First, there is no guarantee that after some time t every source–receiver channel is perfect. Also, the source may transmit linear combinations at any time, thus the side information of a receiver might not just compose of single messages. Further, optimizing for the objective in [31] does not necessarily minimize ours.

As far as we know, this is the first work that examines jointly optimizing coding and use of feedback information for delay-optimal content delivery. This paper builds on our previous conference papers on the subject [32, 4].

2. The Model

Consider a source that wants to convey μ messages to ρ receivers using broadcast transmissions. Time is slotted, and at the beginning of each time slot $t \geq 1$, the source is allowed to transmit (linear or non-linear) combinations of the messages, which we call packets. We denote the packet transmitted at the beginning of time slot $t \geq 1$ by $p(t)$. We call a broadcasting scheme that schedules single (uncoded) messages at every time step a *scheduling* scheme. If the scheme is allowed to use coding operations (combinations) on the messages, we call it a *coding* scheme. Each receiver receives information from the source through an erasure channel, which might range from only deep fadings to i.i.d. erasures and may depend on the erasure channels of other receivers. We denote by $K_j^t \in \{0, 1\}$ the realization of receiver r_j's channel at time t with $K_j^t = 1$ if and only if r_j receives $p(t)$. In a worst case model these realizations could have given values, while in a probabilistic setting they would be random variables.

Depending on whether the transmitted packet successfully reached a receiver or not, the receiver sends an ACK or a NACK to the source respectively (we assume that the feedback channels are perfect). We assume that K_j^t is received by the source at the end of time slot t. Therefore the source can use this information for generating the packet $p(t+1)$ transmitted at the next time slot $t+1$. We assume that the duration of the time slot is sufficient for the receivers to receive the packet and decode it (if possible) using the packets they have already received. A receiver who has decoded all μ messages is no longer interested in the source transmissions. The source transmits a packet at the beginning of every time slot until all receivers have decoded all messages.

We can think of the μ source messages as defining a μ-dimensional space over a finite field $\mathbb{F}_q$, where each message corresponds to one of the orthonormal basis vectors $\{e_1, e_2, \ldots, e_\mu\}$. We will denote by $p(t)$ the linear packet [1] the source transmits at time t. Linear packets are of the form (c, x) where $c \in \mathbf{F}_q^\mu$ and $x = \sum_j c_j e_j$; the choice of the coefficient vector c determines x, so we leave x implied in what follows. Operations over a finite field of size say $q = 2^\ell$ in practice means that we divide the binary packets the source produces into contiguous sets of ℓ bits, and treat each such set as a symbol of $\mathbf{F}_q$. Linear combining of the packets occurs symbol-wise.

Let Π_j^t be the subspace spanned by the packets collected by r_j *at the end* of time slot t and E_j^t the set of basis vectors $e_\ell \in \Pi_j^t$. We say that a received vector (packet) brings *novel information* to a receiver r_j if it increases the dimension of Π_j by one. A class of schemes that will play an important role henceforth are schemes where every successfully received packet brings novel information to its receiver. We call these schemes *rate-optimal*. In a rate-optimal scheme, a receiver r_j that has received ℓ packets, has collected an ℓ-dimensional subspace Π_j of the μ-dimensional space. For $\ell = \mu$, the receiver can successfully decode all source messages. The following properties of rate-optimal schemes are straightforward:

1. A receiver r_j can decode the source message e_i if and only if $e_i \in \Pi_j$; and
2. With slight abuse of notation, let Π_1 denote the subspace spanned by the vectors $< p(i_1), \ldots, p(i_{t_1}) >$ receiver r_j has collected by time t_1 (recall that r_j may have some erasures during the t_1 time steps, hence $i_{t_1} \leq t_1$), and Π_2 the subspace spanned by the vectors $< p(i_{t_1}+1), \ldots, p(i_{t_2}) >$ the same receiver r_j collects between times t_1+1 and t_2. If E_1 is the set of vectors $e_\ell \in \Pi_1$ and E_2 the set of vectors $e_\ell \in \Pi_2$, then $E_1 \cap E_2 = \emptyset$, for all j, $t_1 < \mu$ and $t_2 \leq \mu$.

Let $\mathbf{1}(\ \cdot\ )$ denote the indicator function and $|S|$ the size of set S.

Definition 1. *The delay d_j^T that receiver r_j experiences when the source uses transmission scheme T is the number of packets that, although successfully received, did not allow r_j to immediately decode a new message.* [2] *In symbols,*

$$d_j^T \triangleq 1 + \sum_{t:|E_j^t|<\mu} \mathbf{1}(\ E_j^t = E_j^{t-1}\ )\ \cdot\ K_j^t$$

We call $\sum_{1\leq j\leq\rho} d_j^T$ the total *delay of the scheme.*

Let $\mathcal{D}_a^T$ and $\mathcal{D}_w^T$ denote the average and worst case delay of transmission scheme T respectively, given by

$$\mathcal{D}_a^T \triangleq \frac{1}{\rho} \sum_{1\leq j\leq\rho} d_j^T, \quad \text{and} \quad \mathcal{D}_w^T \triangleq \max_{1\leq j\leq\rho} d_j^T.$$

[1] It is not difficult to generalize this discussion to packets that consist of nonlinear combinations of messages.

[2] We introduce the +1 in the delay for technical reasons –we can interpret this as setup time: e.g., the time slot $t = 0$ is used by the source to identify the number of receivers in the system.

Different schemes may result in different values for $\mathcal{D}_a^T$ and $\mathcal{D}_w^T$. Our goal is to find among all transmission schemes the (possibly different, e.g. , see [32]) transmission schemes under which the average and worst-case delay are minimized.

To better illustrate the definitions of delay and rate-optimality above, consider a broadcasting scheme that achieves the minimum delay of one for a certain broadcasting instance. Given our definition of delay, such a scheme must be rate-optimal: at every time step, every receiver who successfully received, obtained some novel information. On the other hand, consider a scheme that incurred delay greater than one. Clearly, such a scheme may be one that is not rate-optimal, implying that some receivers did not receive novel information during certain time steps. However, given our definition of delay, a scheme incurring delay greater than one may very well be a rate-optimal scheme, as follows. At every time step, novel information was brought to every successfully receiving receiver, however some receiver(s) was not able to extract any (yet) unknown message during some time step(s). Therefore delay occurred.

The discussion above implies that further insight into the online problem can be gained by examining its offline version. In addition to the inputs ρ and μ to the online problem, an *offline broadcasting instance* has two other inputs: (a) an integer τ that stands for the number of source transmissions by which all receivers have received all messages, and (b) a $\tau \times \rho$ symbolic matrix P whose entries take values from $\{\surd,\text{x}\}$. P is defined by the successful receptions and the erasures of the ρ receivers during the τ time slots, which are known in the offline scenario: entry $P(t,j) = \surd$ if and only if receiver r_j successfully received the packet $p(t)$ transmitted at time slot t. [3] We shall henceforth denote an offline broadcasting instance by the quadruple (μ, ρ, τ, P), and refer to P as the erasure matrix (or erasure pattern) of the instance.

We say that a broadcasting scheme for the source *completes* the offline instance (μ, ρ, τ, P) if by time τ, all receivers have decoded all messages. Observe that a necessary and sufficient condition for a scheme to complete the offline instance is that every receiver has at least μ successful receptions by time τ. For example, any rate-optimal scheme requires exactly μ successful receptions per receiver to guarantee that all receivers can decode all messages, regardless of the delay.

As mentioned in the Introduction, offline analysis is useful because it can be used as a benchmark for the online algorithms: the performance of the optimal offline algorithm lower bounds the performance of *any* online algorithm. Moreover, offline problems can be particularly interesting and challenging on their own, as the works on Index Coding [28, 29, 30, 31] show.

3. Minimizing Delay with Scheduling Schemes is *NP*-hard

Given an offline broadcasting instance (μ, ρ, τ, P), the *scheduling* problem we are considering is to minimize the average (maximum) delay under any scheduling scheme that completes the instance. Observe that a priori this appears to be an easier problem than the one studied in [12] since our notion of delay is relaxed as all receivers need all messages and not specific subsets of messages, and further, the order in which the messages are received does not matter.

The decision version of the optimization problem above has as an extra input an integer $d \geq 1$, and answers "yes" if and only if there is a *scheduling* scheme that completes (μ, ρ, τ, P) with total (maximum) delay at most d [4] . An algorithm that solves the minimization problem

[3] We introduce $P(t,j)$ for the offline scenario while we use K_j^t in the online scenario.

[4] Minimizing total delay is equivalent to minimizing average delay when μ is independent of ρ.

for total (maximum) delay should be able to answer the decision problem for every value of d. Since the minimum possible value for both average and maximum delay is one (the delay corresponding to the initialization phase), and since average delay of one implies maximum delay of one, it suffices to prove that it is hard to decide if the average delay is one, in order to prove that both minimization problems are NP-hard. This is the main result of this section and it is summarized in the following theorem.

Theorem 1. *Minimizing average and maximum delay in offline broadcasting in the presence of erasures and when the source uses scheduling schemes is NP-hard.*

In the rest of this section we will prove Theorem 1 by reducing 3SAT to average delay of one in offline broadcasting. For brevity, we shall henceforth refer to average delay simply as delay.

Given a formula ϕ in CNF on n variables $x_1, \ldots, x_n$, and m clauses $c_1, \ldots, c_m$, where each clause consists of disjunctions of exactly 3 literals, we must decide if there is an assignment of truth values to the variables such that all clauses are satisfied.

We will construct an offline broadcasting instance $B(\phi) = (\mu, \rho, \tau, P(\phi))$ such that ϕ is satisfiable if and only if there is a scheduling scheme that completes $B(\phi)$ with delay one. In our instance, the source has $\mu = 2n$ messages, there are $\rho = n+2m$ receivers, and $\tau = 4n+5m$ time slots. Our construction guarantees that each receiver has exactly μ successful receptions by time τ. Notice that this suffices to decide if there is a delay-one scheme for our instance: any such scheme has to be rate-optimal and therefore must deliver all μ messages to every receiver during the first μ successful receptions of the receiver starting at $t = 1$.

In more detail, our construction works as follows. For every variable x_i, $1 \leq i \leq n$ we introduce 2 messages, e_i and $\overline{e}_i$. One receiver D^i is introduced for every variable x_i (their role will be discussed after the construction of $P(\phi)$ is complete). Also, two receivers, C_1^j and C_2^j are introduced for every clause c_j, $1 \leq j \leq m$. This results in the erasure matrix $P(\phi)$ having $\rho = n + 2m$ columns.

We now move on to discussing the number of rows in $P(\phi)$. For every variable x_i, we introduce 4 consecutive time slots, which we call the *variable period* β_i; β_i starts at time slot $4(i-1)+1$, and ends at time slot $4i$. Following the n-th variable period, we introduce m consecutive *clause* periods: the j-th clause period, denoted by γ_j, consists of 5 time slots, starts at time slot $4n+5(j-1)+1$, and ends at time slot $4n+5j$. Hence $P(\phi)$ has $\tau = 4n+5m$ rows.

To complete our construction, we need to assign values to the $\tau \cdot \rho$ entries of $P(\phi)$. We will do this sequentially in time, i.e., by first considering the *variable* periods and then the *clause* periods.

Time slot	C_1^j	C_2^j
$4(i-1)+1$	$\surd$	x
$4(i-1)+2$	x	x
$4(i-1)+3$	x	$\surd$
$4i$	x	x

Time slot	C_1^j	C_2^j
$4(i-1)+1$	$\surd$	x
$2(i-1)+2$	x	x
$4(i-1)+3$	x	x
$4i$	x	$\surd$

Time slot	C_1^j	C_2^j
$4(i-1)+1$	$\surd$	$\surd$
$4(i-1)+2$	$\surd$	$\surd$
$4(i-1)+3$	x	x
$4i$	x	x

Table 1: Erasure patterns for receivers C_1^j, C_2^j during β_i. If clause c_j contains x_i, they receive as in the left table; if c_j contains $\overline{x}_i$, they receive as in the middle table; else (c_j does not contain x_i or $\overline{x}_i$), they receive as in the right table.

During variable period β_i, for all $1 \leq j \leq m$, receivers C_1^j, C_2^j corresponding to clause c_j receive as shown in Table 1 depending on whether x_i, $\overline{x}_i$ or none of them appears in c_j. Also, during β_i, receivers D^ℓ for $1 \leq \ell \leq n$, receive as shown in Table 2: for $\ell \neq i$, D^ℓ receives during the first two time slots of β_i, while D^i receive during the last two time slots.

Time slot	D^1	...	D^{i-1}	D^i	D^{i+1}	...	D^n
$4(i-1)+1$	$\surd$	...	$\surd$	x	$\surd$	...	$\surd$
$4(i-1)+2$	$\surd$	...	$\surd$	x	$\surd$	...	$\surd$
$4(i-1)+3$	x	...	x	$\surd$	x	...	x
$4i$	x	...	x	$\surd$	x	...	x

Time slot	C_1^j	C_2^j
$4n+5(j-1)+1$	$\surd$	$\surd$
$4n+5(j-1)+2$	x	$\surd$
$4n+5(j-1)+3$	x	$\surd$
$4n+5(j-1)+4$	$\surd$	x
$4n+5j$	$\surd$	x

Table 2: The left table shows receptions of $D^1, \ldots, D^n$ during β_i. The right table shows receptions of C_1^j, C_2^j during clause period γ_j (all other receivers experience erasures during γ_j).

During clause period γ_j, receivers C_1^j, C_2^j corresponding to clause c_j receive as shown in the right table of Table 2. All other receivers experience erasures during γ_j.

The above completes our construction. Table 4 in the Appendix shows $P(\phi)$ for the example formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$ for which $B(\phi) = (8, 10, 31, P(\phi))$.

Some remarks are appropriate at this point. First, it is easy to check that every receiver has exactly μ successful receptions. So a priori there could be a scheduling scheme completing $B(\phi)$ with delay one. The role of the receivers D^i is to guarantee that exactly 2 messages are sent during each β_i, with the two messages sent during the first two time slots being rescheduled during the last two time slots, in any order (see Proposition 2 for a proof). In effect, this flexibility in the scheduling of the messages during the last two time slots of each β_i is our choice gadget. Our consistency gadget is that during β_i, C_2^j receives a different message from C_2^ℓ when x_i appears in clause c_j and $\overline{x}_i$ in c_ℓ. Finally our clause constraint gadget is the simultaneous reception of the two receivers corresponding to clause c_j during the first time slot of γ_j.

We now move to showing that ϕ is satisfiable if and only if $B(\phi)$ admits delay one. Before, we introduce the following two schedulings which will prove useful for our arguments.
Scheduling 1 for variable period β_i: the ordered sequence of messages $e_i, \overline{e}_i, e_i, \overline{e}_i$.
Scheduling 2 for variable period β_i: the ordered sequence of messages $e_i, \overline{e}_i, \overline{e}_i, e_i$.

Proposition 1. *If ϕ is satisfiable, then there is a scheduling scheme T_S that satisfies the offline broadcasting instance $B(\phi) = (2n, 2m+n, 4n+5m, P(\phi))$ with delay one.*

Proof. Consider a satisfying truth assignment for ϕ. For $1 \leq i \leq n$, if x_i is true, T_S applies Scheduling 1 for β_i during β_i, else if x_i is false, T_S applies Scheduling 2 for β_i during β_i. Then the first $4n$ transmissions of T_S incur delay one, and $D^1, \ldots, D^n$ obtain all messages.

Since ϕ is satisfiable, every clause has at least one literal that is true. W.l.o.g., let $c_j = (\ell_i \vee \ell_a \vee \ell_b)$ be any clause, where ℓ_y is either x_y or $\overline{x}_y$, and suppose that ℓ_i is (one of) the satisfying literal(s) for this clause, i.e., x_i is set to true if and only if $\ell_i = x_i$. We now show how T_S completes the clause periods so that the clause receivers obtain all messages without delay.

By time $4n$, receivers C_1^j, C_2^j know $2n-6$ messages, i.e., all messages corresponding to the variables that do not appear in clause c_j. Further (see Table 1), C_1^j knows $\{e_i, e_a, e_b\}$, and C_2^j knows exactly one from $\{e_a, \overline{e}_a\}$ and one from $\{e_b, \overline{e}_b\}$. C_2^j also knows e_i which he received at the third time slot of β_i if x_i appears in c_j (in which case, since ℓ_i is the satisfying literal of c_j, x_i was set to true, and T_S applied Scheduling 1 for β_i during β_i), or at the fourth time slot of β_i if $\overline{x}_i$ appears in c_j (in which case, x_i was set to false and T_S applied Scheduling 2). Then during the first time slot of γ_j (see Table 2), T_S sends $\overline{e}_i$. Next, T_S sends e_a or $\overline{e}_a$, depending on which one C_2^j missed during β_a; similarly for e_b during the third time slot. Finally T_s schedules $\overline{e}_a$ and $\overline{e}_b$ during the last two slots of γ_j. Since these transmissions result

in C_1^j, C_2^j obtaining all messages without delay, T_S satisfies $B(\phi)$ with minimum delay one. □

Conversely, let T'_S be any scheduling scheme that satisfies $B(\phi)$ with delay one. We will exhibit a satisfying truth assignment for ϕ. Intuitively, to complete the proof of the reverse direction, we will exhibit two key properties of any such T'_S. The first property stems from the fact that T'_S introduces zero delay during the variable periods. We will show that this can only happen if there is a certain structure in the scheduling scheme T'_S during the variable periods. In effect, this structure will allow us to extract unambiguous truth values for the variables in the formula ϕ, which will further be consistent among clauses, as forced by our consistency gadgets.

The second property stems from the fact that T'_S introduces zero delay during the clause periods. We will show that this can only happen if certain messages have been received by certain receivers by the beginning of the clause periods. In effect, using the truth assignment derived from the variable periods, our clause constraint gadgets will ensure that reception of these messages implies at least one true literal in every clause.

We formalize the intuition above in the following propositions. We first introduce some notation: for $1 \leq t_1 \leq t_T \leq 4$, we define $E_i^{t_1 \ldots t_T}$ to be the (simple) set of the messages scheduled at the discrete time steps $t_1, \ldots, t_T$ of β_i. For example, E_i^{123} is the simple set of messages sent during the first 3 slots of β_i. We can now show a technical but useful lemma concerning properties of T'_S during the variable periods.

Proposition 2. *Consider any scheduling scheme T'_S that satisfies $B(\phi)$ with delay one. For $1 \leq i \leq n$, T'_S schedules exactly two new messages during β_i, with the messages sent during the first two time slots of β_i being resent (in some order) during the last two time slots. In symbols, for all $1 \leq i \neq j \leq n$, $E_i^{12} = E_i^{34}$ and $E_i^{12} \cap E_j^{12} = \emptyset$.*

Proof. Trivially, T'_S is rate-optimal for every D^i since it satisfies $B(\phi)$ with delay one. Thus $|E_i^{12} \cup E_i^{34}| \geq 2$ for all i. Now suppose that there is a k such that $E_k^{12} \neq E_k^{34}$. It follows that $|E_k^{12} \cup E_k^{34}| \geq 3$, so by the pigeonhole principle, there is a message m that was scheduled during both β_k and some β_j with $j \neq k$. During β_k, m was either received by all D^ℓ with $\ell \neq k$ or by D^k or both. During β_j, m was either received by all D^i with $i \neq j$ or by D^j or both. If m was received by all D^ℓ and all D^i, all D^y with $y \neq k, j$ receive m twice (in β_k and in β_j). If m was received by D^k and all D^i, then D^k received m twice. If m was received by all D^ℓ and D^j, then D^j received m twice. Since T'_S does not introduce delay, we conclude that none of these scenarios may have happened and it must be that m was received by D^k in β_k and by D^j in β_j. Then $m \in E_k^{34} \cap E_j^{34}$. Since T'_S is rate-optimal for D^k and D^j, it follows that $|E_1^{12} \cup \ldots \cup E_{k-1}^{12} \cup E_k^{34} \cup E_{k+1}^{12} \cup \ldots \cup E_n^{12}| = 2n$, and $|E_1^{12} \cup \ldots \cup E_{j-1}^{12} \cup E_j^{34} \cup E_{j+1}^{12} \cup \ldots \cup E_n^{12}| = 2n$, hence $E_j^{12} \cup E_k^{34} = E_j^{34} \cup E_k^{12}$. Since T'_S is also rate-optimal for every D^i with $i \neq j, k$, we have that $E_j^{12} \cap E_k^{12} = \emptyset$ (D^i receives E_j^{12} in β_j and E_k^{12} in β_k). We arrive at a contradiction: if $m \in E_k^{34} \cap E_j^{34}$, it is impossible that $E_j^{12} \cup E_k^{34} = E_j^{34} \cup E_k^{12}$ when $E_j^{12} \cap E_k^{12} = \emptyset$. Hence m cannot be in $E_k^{34} \cap E_j^{34}$, implying that for all k, $E_k^{12} = E_k^{34}$. □

W.l.o.g., assume that T'_S schedules the two messages e_{x_i}, e_{y_i} during the first two time slots of β_i, in this order. By Proposition 2, these messages will not be rescheduled before time $4n$, so for the sake of clarity, we may relabel them as $e_i, \overline{e}_i$ respectively. We define the following truth assignment. For $1 \leq i \leq n$, if T'_S applied Scheduling 1 for β_i during β_i, x_i is set to true, else if T'_S used Scheduling 2 for β_i during β_i, x_i is set to false. Notice that Proposition 2 guarantees that any T'_S indeed applied one of these two schedulings during β_i.

We are now ready to conclude the converse direction of our reduction after stating one more proposition whose proof will appear shortly.

Proposition 3. *Let $c_j = (\ell_i \vee \ell_a \vee \ell_b)$ be* any *clause. Any T'_S that satisfies $B(\phi)$ with delay one is such that C_2^j has received at least one of e_i, e_a, e_b by time $4n$.*

Corollary 1. *If T'_S is a scheduling scheme that satisfies $B(\phi)$ with delay one then ϕ is satisfiable.*

Proof. Consider any clause $c_j = (\ell_i \vee \ell_a \vee \ell_b)$. By Proposition 3, any T'_S is such that C_2^j has received at least one of e_i, e_a, e_b by time $4n$. W.l.o.g., assume C_2^j received e_i. If C_2^j received this message at time $4(i-1)+3$, then x_i appears in c_j and T'_S used Scheduling 1 for β_i. Hence our truth assignment set x_i to true. Otherwise, if C_2^j received e_i at time $4i$, then $\overline{x}_i$ appears in c_j and T'_S used Scheduling 2 for β_i. Hence our truth assignment set x_i to false. In either case, our truth assignment for x_i satisfies c_j. Since Proposition 3 applies to all C_2^j for $1 \leq j \leq m$, there is (at least) one literal in every clause that is set to true by our truth assignment. Hence ϕ is satisfiable. □

We now give the proof of Proposition 3.

Proof. Consider the 2 receivers C_1^j, C_2^j corresponding to c_j. By Proposition 2, under any T'_S, at the beginning of γ_j, each of C_1^j, C_2^j knows the $2n-6$ messages that correspond to the $n-3$ variables that do not appear in c_j. Further, C_1^j knows $\{e_i, e_a, e_b\}$, and C_2^j knows exactly one of $\{e_i, \overline{e}_i\}$, one of $\{e_a, \overline{e}_a\}$, and one of $\{e_b, \overline{e}_b\}$.

Suppose that C_2^j received $\{\overline{e}_i, \overline{e}_a, \overline{e}_b\}$. Then at the first slot of γ_j where he receives simultaneously with C_1^j, there is no way to avoid delay since every message needed by one receiver incurs delay to the other. However if C_2^j had received at least one of e_i, e_a, e_b, say e_i, then at the first time slot of γ_j, T'_S could schedule $\overline{e}_i$ which does not delay any receiver. □

4. Benefits and Limits of Coding in Reducing Complexity

We here start by attempting to understand what are structural properties of instances where use of offline scheduling results in delay greater than one. We will then show that use of coding across messages can offer two benefits:

1. Reduces the delay. For example, we will see that there are instances where with coding we can have minimum delay one, while with scheduling we cannot.
2. Reduces the complexity of solving the problem for several cases. For example, for the erasure pattern in Section 3, we can trivially achieve average and maximum delay one: during β_i, with $1 \leq i \leq n$, send $e_i, \overline{e}_i, e_i, \overline{e}_i$, while during γ_j for clause $c_j = (\ell_i \cap \ell_a \cap \ell_b)$ as before, with $1 \leq j \leq m$, send $e_i + \overline{e}_i$, then whatever is missing from C_2^j, and finally $\overline{e}_a, \overline{e}_b$.

The main purpose of this section is to examine whether and how much use of coding can help.

We will use the following notation. Let B_t denote the set of messages the source has transmitted up to time t and $\overline{B}_t$ the set of remaining messages. For receiver r_j, let E_j^t denote the set of messages from B_t that r_j has received, and $\overline{E}_j^t$ the messages from B_t it has not. That is, $B_t = E_j^t \cup \overline{E}_j^t$ for all r_j.

For the case of one receiver, trivially, scheduling achieves delay one. For the case of two receivers, we can use the following simple algorithm to ensure delay one: if at time time t (i) both r_1 and r_2 receive, transmit a message from $\overline{B}_t$ (ii) only r_j receives, if $\bar{E}_j^t \neq \emptyset$ transmit a message from $\overline{E}_j^t$, otherwise a message from $\overline{B}_t$. This scheme ensures that at each time t either $\overline{E}_1^t = \emptyset$ or $\overline{E}_2^t = \emptyset$; moreover, $\overline{B}_t = \emptyset$ only when at least one of the two receivers has received all messages.

For the case of three receivers, offline scheduling can result in worst case delay of $\mathcal{O}(\mu)$. Indeed, note that for scheduling, delay is introduced only when the transmission scheme cannot be rate optimal. For the erasure pattern in Table 3, assume that each line is repeated for $\mu/2$ time slots. Rate optimality for r_3 implies that at $t = \mu + 1$, $\overline{E}_1^t \cap \overline{E}_2^t = \emptyset$ and thus, the transmissions at time-slots $t = \mu + 1, \ldots, 3\mu/2$ will incur total delay $\mu/2$ for r_1 and r_2. The existence of receiver r_3 is necessary to ensure that $\overline{E}_1^t \cap \overline{E}_2^t = \emptyset$ occurs in offline.

time-slots	r_1	r_2	r_3
$t = 1, \ldots, \mu/2$	$\surd$	x	$\surd$
$t = \mu/2 + 1, \ldots, \mu$	x	$\surd$	$\surd$
$t = \mu + 1, \ldots, 3\mu/2$	$\surd$	$\surd$	x

time-slots	r_1	r_2	r_3	r_4
$t = 1, \ldots, \mu/2$	$\surd$	x	$\surd$	x
$t = \mu/2 + 1, \ldots, \mu$	x	$\surd$	$\surd$	x
$t = \mu + 1, \ldots, 3\mu/2$	$\surd$	$\surd$	x	$\surd$

Table 3: The left table gives an erasure pattern where scheduling incurs delay $\mathcal{O}(\mu)$ but coding achieves delay one; the right table gives an erasure pattern where coding as well incurs delay $\mathcal{O}(\mu)$.

The following straightforward proposition formalizes this observation.

Proposition 4. *If at time t there exist receivers r_i and r_j such that $\overline{E}_i^t \cap \overline{E}_j^t = \emptyset$, and following time t, for the next D timeslots that r_i successfully receives so does r_j, with $D \triangleq \min\{|\overline{E}_i^t|, \quad |\overline{E}_j^t|\}$, then offline scheduling results in delay $\Omega(D)$.*

Use of coding allows to make the source transmissions rate optimal, for all possible erasure patterns. For example, for the pattern in the left Table 3, it is sufficient at time-slots $t = \mu + 1, \ldots, 3\mu/2$ to transmit $\mu/2$ messages from $\overline{E}_1^t + \overline{E}_2^t$. However, in this case delay is introduced, if a receiver cannot decode a received linear combination. This is the case for the pattern in the right Table 3 (see also [32]). It is easy to see that, at time $t = \mu + 1$, $\overline{E}_1^t \cap \overline{E}_2^t = \emptyset$, and additionally, $\overline{E}_1^t \subset \overline{E}_4^t$, $\overline{E}_2^t \subset \overline{E}_4^t$. To be rate optimal with respect to r_1 and r_2 we need, like before, to transmit from $\overline{E}_1^t + \overline{E}_2^t$. However, these transmissions cannot be decoded by r_4. Thus similarly to before we now have:

Proposition 5. *If at time t there exist receivers r_i, r_j and r_k such that $\overline{E}_i^t \cap \overline{E}_j^t = \emptyset$, $\overline{E}_i^t \subset \overline{E}_k^t$, $\overline{E}_j^t \subset \overline{E}_k^t$, and following time t, for the next D timeslots that r_i successfully receives so does r_j and r_k, with $D \triangleq \min\{|\overline{E}_i^t|, \quad |\overline{E}_j^t|\}$, then offline coding results in delay $\Omega(D)$.*

Clearly, coding allows to achieve delay one for a larger set of erasure patterns than scheduling. Some additional such patterns are described in the following proposition.

Proposition 6. *With coding we can achieve delay one when we have an arbitrary number of receivers ρ and:*

1. *Erasure patterns where each broadcast transmission is successfully received by at most two receivers (this corresponds to high erasure probability).*
2. *Erasure patterns where each broadcast transmission is not received by at most one receiver (this corresponds to low erasure probability).*

Proof. For two sets A, B, the notation $A + B$ means $\{a + b : a \in A, b \in B\}$.
1. If two receivers r_i and r_j receive, transmit a message from $\overline{E}_i^t \cap \overline{E}_j^t$ if not empty, otherwise from $\overline{B}_t$. If only r_i receives, if $\bar{E}_i^t \neq \emptyset$ transmit from $\bar{E}_i^t$, otherwise a message from $\overline{B}_t$.
2. Use systematic transmission (see 5 for definition) for μ time slots. After μ timeslots there exists at most one receiver who has not received each message. Continue by transmitting messages from $+_i \overline{E}_i^t$, where in the summation we include all receivers i that receive the transmission (and have not completed reception). $\square$

4.1. Minimizing Delay with Coding Schemes is NP-hard

In Section 4, we exhibited instances where the problem of minimizing delay becomes simpler when the source is allowed to use coding schemes, as opposed to just scheduling schemes. The question that arises now is whether this is true for all instances, in other words, whether the problem of minimizing delay using coding schemes becomes polynomial time, or remains NP-hard. Note that the problem of maximizing the throughput when multicasting over graphs becomes polynomial time if coding at intermediate network nodes is allowed [33], while, if coding is not allowed, it is NP-hard. However the theorem below shows that this is not the case in our problem.

Theorem 2. *Minimizing average and maximum delay in offline broadcasting in the presence of erasures and when the source uses (linear or nonlinear) coding is NP-hard.*

The proof of this theorem appears in the Appendix. It builds on and extends the ideas in the proof of Theorem 1. Some major differences and similarities between the two proofs are summarized below.

The proof consists again of a reduction from 3SAT to average delay of one in offline broadcasting when coding is allowed. Our construction is now more involved and requires more receivers, more messages and a longer completion time. The new reduction gadgets are more sophisticated as well to deal with the extra power coding offers over scheduling. However the main idea of the scheduling proof, which is the extraction of the truth assignment from the scheduling of the messages during the variable periods, is still present.

We conclude this section with an inapproximability result following the definitions in [34]

Proposition 7. *Unless $P = NP$, there is no ϵ-factor approximation (coding) algorithm for maximum delay in offline broadcasting with erasures for $\epsilon < 1/2$.*

Proof. Let OPT be the optimal algorithm for offline broadcasting in the presence of erasures, and $OPT(x)$ the maximum delay incurred by OPT on instance $x = (\mu, \rho, \tau, P)$. Let T_C be an ϵ-factor approximation algorithm for maximum delay in offline broadcasting with erasures for some $\epsilon < 1/2$. Then for all inputs x to T_C, $T_C(x) \leq \frac{1}{1-\epsilon} \cdot OPT(x)$.

Given a formula ϕ in 3CNF, we construct $B(\phi)$ in polynomial time as described in the proof of Theorem 2 and run T_C on input $B(\phi)$. There are two cases: if $T_C(B(\phi)) = 1$, then we know that ϕ is satisfiable. Otherwise, if $T_C(B(\phi)) \geq 2$, we conclude that

$$2 \leq T_C(B(\phi)) \leq \frac{1}{1-\epsilon} \cdot OPT(B(\phi)) \Rightarrow$$

$$2(1-\epsilon) \leq OPT(B(\phi)) \Rightarrow 1 < OPT(B(\phi)).$$

Since the maximum delay of $B(\phi)$ is one if and only if ϕ is satisfiable, we conclude that ϕ is not satisfiable. Since $B(\phi)$ can be constructed in polynomial time, and T_C runs in polynomial time, we can decide whether ϕ is satisfiable in polynomial time. □

5. Online Algorithms

In this section, we discuss the competitive ratio of a natural class of systematic rate-optimal online algorithms for minimizing average (maximum) delay in the cases of arbitrary and i.i.d. erasures. We also suggest an online heuristic that improves significantly on the performance of the other schemes.

A systematic coding algorithm uses the first μ transmissions to send all messages once uncoded and then starts sending combinations of messages. Rate optimal systematic algorithms

have smaller average delay than their non-systematic variants where linear combinations of all messages are used for every transmission. The competitive ratio of systematic rate-optimal algorithms in the presence of a deterministic adversary who only knows that the source is using such an algorithm is given by the following proposition. The adversary is allowed to incur an erasure at any channel during any time step but is not allowed to observe any channel.

Proposition 8. *For $\mu < \rho$ and arbitrary erasures, a systematic rate-optimal online algorithm is $(\mu - O(1))$-competitive for minimizing average delay and $(\mu - 1)$-competitive for minimizing maximum delay.*

Proof. During the first μ transmissions, the adversary allows receiver r_j to obtain transmission j and causes erasures to all other receivers. After the μ-th transmission the adversary allows all ρ receivers to successfully receive all transmitted packets.

Since exactly one receiver has obtained each message by time μ, the source must transmit linear combinations of all messages starting at time $\mu+1$ in order to be rate optimal for every receiver. This incurs a total delay of

$$\rho + (\mu - 2) \cdot \rho + (\rho - \mu)$$

during the following μ transmissions. The first term comes from $t = 0$, the second term from time $\mu + 1$ up to time $2\mu - 2$ where all receivers delay, and the last term from time $2\mu - 1$ where receivers $r_{\mu+1}, \ldots, r_\rho$ delay (at this point, receivers $r_1, \ldots, r_\mu$ have successfully received μ packets hence can decode for the μ messages and do not delay). It follows that the average delay is given by

$$\mu - \frac{\mu}{\rho} = \mu - O(1),$$

while the maximum delay is $\mu - 1$ (the delay of any receiver r_j with $j > \mu$).

On the other hand, the optimal offline algorithm incurs average and maximum delay of one: it only transmits e_1 during the first μ transmissions, followed by $e_2, \ldots, e_\mu, e_1$ during the following μ transmissions. The proposition follows. □

Proposition 8 motivates us to look at algorithms that are not rate-optimal in the online scenario. Specifically, we examine the case where all ρ channels experience i.i.d. erasures with common constant erasure probability q. Figure 1 compares the performance of scheduling and three heuristics based on coding:

1. **Scheduling (sorted)**: At every time slot, the message that is not yet received by most receivers is transmitted.
2. **Systematic Forward Error Correction (FEC)**: During the first μ transmissions, all μ messages are transmitted once. Then packets with random linear combinations of the messages are transmitted so that every transmitted packet is rate optimal for every receiver with high probability. With this algorithm nodes rarely receive useless packets but cannot decode most of them until they receive μ of them.
3. **Sorted opportunistic [19]**: Packets are created by combining the messages that are still missing on the largest number of receivers while ensuring that they are decodable by all receivers. The procedure is described in detail in Algorithm 1. For every block b, the function $Missing(b)$ in the algorithm returns the set of receivers that have not decoded it yet. The set S contains all the receivers that will not experience delay if they receive the packet. This algorithm ensures that no receiver ever delays because of undecodable packets but this causes non-innovative packets to be transmitted.
4. **Cost driven**: Unlike the opportunistic algorithm, this algorithm may send non-decodable packets, if it computes a (local) benefit in estimated delay, as explained below.

In this algorithm, described in detail in Algorithm 2, an initial packet p is generated as in the sorted opportunistic algorithm. Subsequently, every message m not contained in p is sequentially considered for inclusion in p. If the estimated delay upon transmission of p decreases by inclusion of m, then m is combined with the existing messages in p. The estimated delay is computed as follows. Let p be the packet generated so far. Let S_1 be the set of receivers that have already decoded all messages combined in p. Then these receivers will experience delay if p is transmitted. Let S_2 be the set of receivers that have not decoded exactly one of the messages contained in p. Upon receiving p, these receivers can decode one additional message. Finally, let S_3 be the set of receivers that have not decoded more than one of the messages combined in p. These receivers are likely not to be able to decode the packet, unless they have received enough linear combinations to do so. The estimated delay generated by the packet p is $|S_1 \cup S_3|$. Notice that some of the receivers in S_3 could potentially decode the packet p if they have already received enough linear combinations.

Algorithm 1 Sorted Opportunistic Algorithm

$(p, S) \leftarrow (0, \emptyset)$
$Q \leftarrow \{e \in e_1, \ldots, e_\mu | Missing(e) \neq \emptyset\}$
while $|Q| > 0$ **do**
    $b \leftarrow \text{argmax}_{e \in Q} |Missing(e)|$
    $Q \leftarrow Q - \{b\}$
    if $Missing(b) \cap S = \emptyset$ **then**
        $p \leftarrow p + b$
        $S \leftarrow S \cup Missing(b)$
    end if
end while
return p

Figure 1 compares the performance of the cost driven algorithm with scheduling, opportunistic and systematic FEC. As mentioned in Section 1, use of coding is critical to achieve lower delay as simple scheduling performs very poorly. Although our new heuristic is clearly suboptimal, it can improve by even 50% the performance of systematic FEC (which achieves the expected delay of $\mu\mathbb{P}(erasure)$) as $\mathbb{P}(erasure)$, ρ and μ increases (shown in Figure 1). It also improves significantly the performance of opportunistic when the number of receivers is large. Further, it achieves more than 78% of the maximum rate for each point in the graphs.

We believe that the interdependence between delay from useless packets and delay from non instantly decodable packets must be further exploited in order to improve the performance of our online algorithm.

Acknowledgements

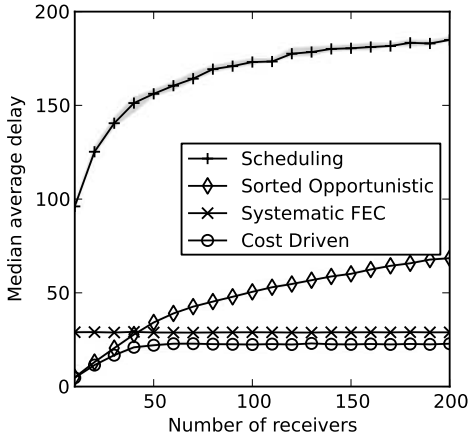
We would like to thank the anonymous reviewers for their insightful comments that significantly helped improve this paper.

6. Conclusions

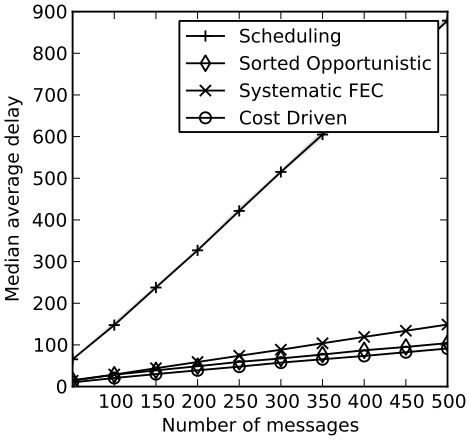
We address the question of finding the optimal offline algorithm for broadcast scheduling or coding that minimizes average and maximum delay when feedback is available. We show that the general offline problem is NP-hard under scheduling schemes, and remains NP-hard even under (linear or non-linear) coding schemes. However we show that in the offline scenario

Algorithm 2 Cost driven Algorithm

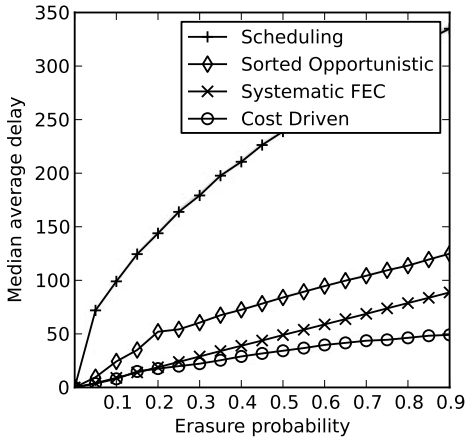
$p \leftarrow 0$
$(S_1, S_2, S_3) \leftarrow (\{r_1, \ldots, r_\rho\}, \emptyset, \emptyset)$
$Q \leftarrow \{e \in e_1, \ldots, e_\mu | Missing(e) \neq \emptyset\}$
while $|Q| > 0$ **do** ▷ Create an instantly decodable packet
    $b \leftarrow \text{argmax}_{e \in Q} |Missing(e)|$
    $Q \leftarrow Q - \{b\}$
    if $Missing(b) \cap B = \emptyset$ **then**
        $S_2 \leftarrow S_2 \cup Missing(b)$
        $S_1 \leftarrow S_1 - Missing(b)$
        $p \leftarrow p + Random(1, q-1) \cdot b$
    end if
end while
$Q \leftarrow \{b \in e_1, \ldots, e_\mu | Missing(b) \neq \emptyset\}$
while Q is not empty **do** ▷ Add messages if estimated delay is smaller
    $b \leftarrow \text{argmax}_{e \in Q} |Missing(e)|$
    $Q \leftarrow Q - \{b\}$
    $S_1' \leftarrow S_1 - Missing(b)$
    $S_2' \leftarrow (S_2 \cup (S_1 \cap Missing(b))) - (S_2 \cap Missing(b))$
    $S_3' \leftarrow S_3 \cup (S_2 \cap Missing(b))$
    if $|B'| > |B|$ **then** ▷ Compare estimated delays
        $(S_1, S_2, S_3) \leftarrow (S_1', S_2', S_3')$
        $p \leftarrow p + Random(1, q-1) \cdot b$
    end if
end while
return p



(a) $\mu = 100$, $\mathbb{P}(\text{erasure}) = 0.3$



(b) $\rho = 40$, $\mathbb{P}(\text{erasure}) = 0.3$



(c) $\mu = 100$, $\rho = 150$

Figure 1: Comparison of the delay of different online algorithms. Every data point is the median of 50 simulations, the gray area corresponds to the 95% confidence interval. Computations are performed using $\mathbf{F}_{2^8}$.

coding offers gains both in terms of delay and complexity. Also, our online heuristic algorithms show that feedback information allows online algorithms designed for delay-sensitive applications to outperform scheduling schemes, as well as standard FEC schemes which do not use feedback.

References

[1] M. Durvy, C. Fragouli, P. Thiran, On feedback for network coding, in: IEEE International Symposium on Information Theory (ISIT), IEEE, 2007.

[2] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, J. Crowcroft, Xors in the air: Practical wireless network coding, in: ACM SIGCOMM, ACM, 2006.

[3] C. Fragouli, D. Lun, M. Medard, P. Pakzad, On feedback for network coding, in: Conference on Information Sciences and Systems, IEEE, 2006.

[4] E. Drinea, C. Fragouli, L. Keller, Delay with network coding and feedback, in: IEEE International Symposium on Information Theory (ISIT), IEEE, 2009.

[5] D. Silva, F. R. Kschischang, Rank-metric codes for priority encoding transmission with network coding, in: Proc. Canadian Workshop on Information Theory, 2007.

[6] R. Ahlswede, N. Cai, S.-Y. R. Li, R. W. Yeung, Network information flow, IEEE Transactions on Information Theory (2000) 1204–1216.

[7] S.-Y. R. Li, R. W. Yeung, N. Cai, Linear network coding, IEEE Transactions on Information Theory 49 (2003).

[8] C. Fragouli, E. Soljanin, Network coding: Fundamentals and Applications, volume 1 of *Foundations and Trends in Networking*, Now Publisher, 2007.

[9] V. K. Goyal, Multiple description coding: Compression meets the network, IEEE Signal Processing Magazine 18 (2001) 74–93.

[10] M. Luby, Lt codes, in: ACM Symposium on Theory of Computing (STOC), ACM, 2002.

[11] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory 52 (2006) 2551–2567.

[12] J. Chang, T. Erlebach, R. Gailis, S. Khuller, Broadcast scheduling: algorithms and complexity, in: ACM Symposium on Discrete Algorithms (SODA), ACM, 2008.

[13] N. Bansal, D. Coppersmith, M. Sviridenko, Improved approximation algorithms for broadcast scheduling, in: ACM Symposium on Discrete Algorithms (SODA), ACM, 2006.

[14] M. Charikar, N. Bansal, S. Khanna, S. Naor, Approximating the average response time in broadcast scheduling, in: ACM Symposium on Discrete Algorithms (SODA), ACM, 2005.

[15] W. Mao, Competitive analysis of on-line algorithms for on-demand data broadcast scheduling, in: International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN), IEEE, 2000.

[16] K. Foltz, L. Xu, J. Bruck, Coding and scheduling for efficient loss-resilient data broadcasting, in: IEEE International Symposium on Information Theory (ISIT), IEEE, 2003.

[17] C. Su, L. Tassiulas, V. J. Tsotras, Broadcast scheduling for information distribution, Wireless Networks 5 (2004).

[18] D. Traskov, M. Medard, P. Sadeghi, R. Koetter, Joint scheduling and instantaneously decodable network coding, in: IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2009.

[19] P. Sadeghi, D. Traskov, R. Koetter, Adaptive network coding for broadcast channels, in: Workshop on Network Coding, Theory, and Applications (NetCod), IEEE, 2009.

[20] J. Sundararajan, P. Sadeghi, M. Medard, A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay, in: Workshop on Network Coding, Theory, and Applications (NetCod), IEEE, 2009.

[21] P. Sadeghi, R. Shams, D. Traskov, An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels, EURASIP Journal on Wireless Communications and Networking 2010 (2010).

[22] D. Lucani, M. Stojanovic, M. Medard, Random linear network coding for time division duplexing: When to stop talking and start listening, in: IEEE INFOCOM, IEEE, 2009.

[23] D. Lucani, M. Me anddard, M. Stojanovic, Online network coding for time-division duplexing, in: IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2010.

[24] J. Barros, R. Costa, D. Munaretto, J. Widmer, Effective delay control in online network coding, in: IEEE INFOCOM, IEEE, 2009.

[25] R. Costa, D. Munaretto, J. Widmer, J. Barros, Informed network coding for minimum decoding delay, in: Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on, IEEE, 2008.

[26] J. Sundararajan, D. Shah, M. Medard, On queueing in coded networks –queue size follows degrees of freedom, in: IEEE Information Theory Workshop (ITW), IEEE, 2007.

[27] J. Massey, P. Massey, Zero error, in: IEEE Information Theory Winter School, IEEE, 2007.

[28] Y. Birk, T. Kol, Informed-source coding-on-demand (ISCOD) over broadcast channels, in: IEEE INFOCOM, IEEE, 1998.

[29] Z. Bar-Yossef, Y. Birk, T. S. Jayram, T. Kol, Index coding with side information, in: IEEE Symposium on the Foundations of Computer Science (FOCS), IEEE, 2006.

[30] E. Lubetzky, U. Stav, Non-linear index coding outperforming the linear optimum, in: IEEE Symposium on the Foundations of Computer Science (FOCS), IEEE, 2007.

[31] N. Alon, A. Hassidim, E. Lubetzky, U. Stav, A. Weinstein, Broadcasting with side information, in: IEEE Symposium on the Foundations of Computer Science (FOCS), IEEE, 2008.

[32] L. Keller, E. Drinea, C. Fragouli, Online broadcasting with network coding, in: Workshop on Network Coding, Theory, and Applications (NetCod), IEEE, 2008.

[33] D. Lun, M. Medard, T. Ho, R. Koetter, Network coding with a cost criterion, in: IEEE International Symposium on Information Theory (ISIT), IEEE, 2004.

[34] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

Eleni Drinea received the B.S. degree with distinction in computer engineering in 1999 from the University of Patras, Patras, Greece. She continued her studies at Harvard University, Cambridge, MA, where she received the M.S. and the Ph.D. degrees in 2005, both in computer science. She joined the New England Complex Systems Institute in 2006 as a postdoctoral fellow working on information theoretic tools for the analysis of medical data. From 2007 to 2009 she was a research associate with the school of computer and communication sciences in EPFL, Switzerland, where her interests centered around quality of service in realistic wireless settings using network coding algorithms.

Lorenzo Keller is pursuing a Ph.D. in the School of Computer and Communication Sciences, EPFL, Switzerland. He received his bachelor and master degrees in Communication Systems from the same institution in 2005 and 2007 respectively. His research is on theoretical and practical aspects of wireless networks and, in particular, on improving the performance, reliability and energy efficiency of sensor networks.

Christina Fragouli is an Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, in 1996, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1998 and 2000, respectively. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. Her research interests are in network information flow theory and algorithms, network coding, and connections between communications and computer science.

Appendix

Time slot	C_1^1	C_2^1	C_1^2	C_2^2	C_1^3	C_2^3	D^1	D^2	D^3	D^4
1	√	x	√	x	√	√	x	√	√	√
2	x	x	x	x	√	√	x	√	√	√
3	x	√	x	x	x	x	√	x	x	x
4	x	x	x	√	x	x	√	x	x	x
5	√	x	√	x	√	x	√	x	√	√
6	x	x	x	x	x	x	√	x	√	√
7	x	√	x	x	x	x	x	√	x	x
8	x	x	x	√	x	√	x	√	x	x
9	√	x	√	x	√	x	√	√	x	√
10	x	x	x	x	x	x	√	√	x	√
11	x	√	x	√	x	x	x	x	√	x
12	x	x	x	x	x	√	x	x	√	x
13	√	√	√	√	√	x	√	√	√	x
14	√	√	√	√	x	x	√	√	√	x
15	x	x	x	x	x	√	x	x	x	√
16	x	x	x	x	x	x	x	x	x	√
17	√	√	x	x	x	x	x	x	x	x
18	x	√	x	x	x	x	x	x	x	x
19	x	√	x	x	x	x	x	x	x	x
20	√	x	x	x	x	x	x	x	x	x
21	√	x	x	x	x	x	x	x	x	x
22	x	x	√	√	x	x	x	x	x	x
23	x	x	x	√	x	x	x	x	x	x
24	x	x	x	√	x	x	x	x	x	x
25	x	x	√	x	x	x	x	x	x	x
26	x	x	√	x	x	x	x	x	x	x
27	x	x	x	x	√	√	x	x	x	x
28	x	x	x	x	x	√	x	x	x	x
29	x	x	x	x	x	√	x	x	x	x
30	x	x	x	x	√	x	x	x	x	x
31	x	x	x	x	√	x	x	x	x	x

Table 4: $P(\phi)$ for the example formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$ from Section 3.

Minimizing Delay with Coding Schemes is NP-hard

In this section we present the proof of Theorem 2. Given an offline instance $B(m, n, h, P)$, the *coding problem* we are considering is to minimize the delay under any coding scheme that satisfies the instance. Again, delay here stands for average delay.

We will construct an offline broadcasting instance $B(\phi) = (\mu, \rho, \tau, P(\phi))$ such that ϕ is satisfiable if and only if there is a coding scheme that satisfies $B(\phi)$ with delay one. In our instance, the source has $\mu = 3n$ messages, there are $\rho = 5n + 3m$ receivers, and $\tau = 10n + 6m - 3$ time slots. Our construction guarantees that each receiver has $\mu = 3n$ successful receptions within these τ time slots.

Our construction works as follows. For every variable x_i, we introduce 3 messages, e_i, $\overline{e}_i$ and e'_i. For every clause c_j, $1 \leq j \leq m$, we introduce three receivers, denoted by C_1^j, C_2^j and C_3^j. For every variable x_i, $1 \leq i \leq n$, we introduce five receivers, denoted by D_k^i, for $1 \leq k \leq 5$, whose role will be discussed after completing the construction of the erasure pattern.

Time slot	C_1^j	C_2^j	C_3^j
$5(i-1)+1$	$\surd$	$\surd$	x
$5(i-1)+2$	$\surd$	x	x
$5(i-1)+3$	x	$\surd$	x
$5(i-1)+4$	x	x	$\surd$
$5i$	x	x	x

Time slot	C_1^j	C_2^j	C_3^j
$5(i-1)+1$	$\surd$	$\surd$	x
$5(i-1)+2$	$\surd$	x	x
$5(i-1)+3$	x	$\surd$	x
$5(i-1)+4$	x	x	x
$5i$	x	x	$\surd$

Time slot	C_1^j	C_2^j	C_3^j
$5(i-1)+1$	$\surd$	$\surd$	$\surd$
$5(i-1)+2$	$\surd$	$\surd$	$\surd$
$5(i-1)+3$	$\surd$	$\surd$	$\surd$
$5(i-1)+4$	x	x	x
$5i$	x	x	x

Table 5: Erasure patterns for receivers C_1^j, C_2^j, C_3^j during β_i. If clause c_j contains x_i, they receive as in the left table; if c_j contains $\overline{x}_i$, they receive as in the middle table; else (c_j does not contain x_i or $\overline{x}_i$), they receive as in the right table.

Time slot	D_1^ℓ	D_2^ℓ	D_3^ℓ	D_4^ℓ	D_5^ℓ
$5(i-1)+1$	$\surd$	$\surd$	x	$\surd$	$\surd$
$5(i-1)+2$	$\surd$	$\surd$	x	$\surd$	$\surd$
$5(i-1)+3$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$5(i-1)+4$	x	x	$\surd$	x	x
$5i$	x	x	$\surd$	x	x

Time slot	D_1^i	D_2^i	D_3^i	D_4^i	D_5^i
$5(i-1)+1$	$\surd$	x	x	x	x
$5(i-1)+2$	$\surd$	$\surd$	x	x	x
$5(i-1)+3$	$\surd$	$\surd$	$\surd$	x	x
$5(i-1)+4$	x	x	$\surd$	$\surd$	x
$5i$	x	x	$\surd$	$\surd$	$\surd$

Table 6: Let $1 \leq k \leq 5$. During β_i, D_k^ℓ receive as shown in the left table if $\ell < i$; the right table shows receptions for D_k^i.

Also, for every variable x_i, we introduce 5 consecutive time slots, which we call the *variable period* β_i; β_i starts at time slot $5(i-1)+1$, and ends at time slot $5i$. Following the n-th variable period, we introduce m consecutive *clause* periods: the j-th clause period, denoted by γ_j, consists of 6 time slots, starts at time slot $5n+6(j-1)+1$, and ends at time slot $5n+6j$. Finally, following the m clause periods, we introduce $5n-3$ time slots, which we call "patching" time slots, because their role is simply to provide sufficient time for the receivers D_k^i to obtain all messages.

We now proceed to giving values to the $\tau \cdot \rho$ entries of the erasure matrix P. We will do this sequentially in time, i.e., by first considering the *variable* periods, then the *clause* periods, and finally the "patching" time slots.

During variable period β_i, for all $1 \leq j \leq n$, receivers C_1^j, C_2^j, C_3^j corresponding to clause c_j receive as shown in Table 5 depending on whether x_i, $\overline{x}_i$ or none of them appears in c_j. Also, during β_i, receivers D_k^ℓ with $1 \leq k \leq 5$ receive as shown in Table 6 if $\ell \leq i$; otherwise (if $\ell > i$), they all experience erasures.

During clause period γ_j, receivers C_1^j, C_2^j, C_3^j corresponding to clause c_j receive as shown in Table 7. All other receivers experience erasures during γ_j.

Finally, consider the "patching" time slots. We can think of them as being grouped into $n-1$ periods of 5 time slots, and one last period of only two time slots. For $1 \leq i \leq n$, at time $5n+6m+5(i-1)+1$, receivers D_4^i, D_5^i receive. At the next time slot $5n+6m+5(i-1)+2$, receivers D_2^i, D_5^i receive. For $1 \leq i \leq n-1$, during the last three time slots of patching period i, all receivers D_k^ℓ with $\ell > i$, $1 \leq k \leq 5$ receive.

The above completes our construction. Table 8 shows $P(\phi)$ for the example formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$ for which $B(\phi) = (12, 29, 55, P(\phi))$.

Again it is easy to check that the reduction can be carried out by a deterministic Turing machine in logarithmic space and that all receivers have exactly $3n$ successful receptions by time τ, hence any rate-optimal scheme would satisfy $B(\phi)$. Here the role of the receivers D_k^i is twofold (see Lemma 1 for a proof): they guarantee that (a) a single message is scheduled during every time slot of every β_i, and (b) exactly 3 messages are sent during each β_i, with the two messages sent during the first two time slots being rescheduled during the last two time

Time slot	C_1^j	C_2^j	C_3^j
$5n+6(j-1)+1$	$\surd$	$\surd$	$\surd$
$5n+6(j-1)+2$	x	x	$\surd$
$5n+6(j-1)+3$	x	$\surd$	$\surd$
$5n+6(j-1)+4$	$\surd$	x	$\surd$
$5n+6(j-1)+5$	x	$\surd$	$\surd$
$5n+6j$	$\surd$	x	$\surd$

Table 7: Receiving pattern of C_1^j, C_2^j, C_3^j during clause period γ_j.

slots, in any order. Similarly to Section 3, this flexibility in the scheduling of the messages during the last two time slots of each β_i is our choice gadget. Our consistency gadget is that during β_i, C_3^j receives a different message from C_3^ℓ when x_i appears in clause c_j and $\overline{x}_i$ in c_ℓ. Finally our clause constraint gadget is the simultaneous reception of the three receivers corresponding to clause c_j during the first time slot of γ_j.

We now move to showing that ϕ is satisfiable if and only if $B(\phi)$ admits delay one. Before, we introduce the following two schedulings that will prove useful for our arguments.
Scheduling 1 for β_i: the ordered sequence of messages $e_i, \overline{e}_i, e'_i, \overline{e}_i, e_i$.
Scheduling 2 for β_i: the ordered sequence of messages $e_i, \overline{e}_i, e'_i, e_i, \overline{e}_i$.

Proposition 9. *If ϕ is satisfiable, then there is a coding scheme T that satisfies the offline broadcasting instance $B(\phi) = (3n, 3m+5n, 10n+6m-3, P(\phi))$ with delay one.*

Proof. Consider a satisfying truth assignment for ϕ. For $1 \leq i \leq n$, if x_i is true, the coding scheme T applies Scheduling 1 for β_i during variable period β_i. Otherwise, if x_i is false, it applies Scheduling 2 for β_i during β_i. Then the first $3n$ transmissions of T incur delay one, and D_1^1, D_3^1 obtain all messages.

Since ϕ is satisfiable, every clause has at least one literal that is true. W.l.o.g., let $c_j = (\ell_i \vee \ell_a \vee \ell_b)$ be any clause, where ℓ_y is either x_y or $\overline{x}_y$, and suppose that ℓ_i is (one of) the satisfying literal(s) for this clause, i.e., x_i is set to true if and only $\ell_i = x_i$. We now show how T completes the clause periods so that the clause receivers obtain all messages without delay.

By time $5n$, receivers C_1^j, C_2^j, C_3^j know $3n-9$ messages, i.e., all messages corresponding to the variables that do not appear in clause c_j. Further (see Table 5), C_1^j knows $\{e_i, \overline{e}_i, e_a, \overline{e}_a, e_b, \overline{e}_b\}$, C_2^j knows $\{e_i, e'_i, e_a, e'_a, e_b, e'_b\}$, and C_3^j knows exactly one from $\{e_a, \overline{e}_a\}$ and one from $\{e_b, \overline{e}_b\}$. C_3^j also knows $\overline{e}_i$ which he received at the fourth time slot of β_i if $\ell_i = x_i$ (in which case, x_i was set to true, and T applied Scheduling 1 for β_i during β_i), or at the fifth time slot of β_i if $\ell_i = \overline{x}_i$ (in which case, x_i was set to false and T applied Scheduling 2).

During γ_j (see Table 7), T sends the following packets: first it sends $\overline{e}_i + e'_i$. This results in C_1^j obtaining e'_i, C_2^j obtaining $\overline{e}_i$, and C_3^j obtaining e'_i. Next it sends e_i. During the third and the fourth slots, if C_3^j received e_a during β_a, it sends $e_a + \overline{e}_a$, and e'_a respectively. Otherwise, if C_3^j received $\overline{e}_a$ during β_a, it sends $\overline{e}_a + e'_a$ and $e_a + e'_a$ respectively. The situation for ℓ_b is identical. Hence by time $5n+6j$, C_1^j, C_2^j and C_3^j have obtained all messages without delay.

At time $5n+6m+1$, every receiver D_k^i except for D_1^1 and D_3^1 still needs some messages. For $1 \leq i \leq n$, at the first time slot of the i-th patching period, T sends e'_i, which both D_4^i and D_5^i need (see Table 6). At the second slot, it sends $e_i + \overline{e}_i$ if x_i was set to true, or e_i if x_i was set to false; this satisfies both D_2^i and D_5^i. At this point, D_2^i, D_4^i, D_5^i have received

all messages. For $1 \leq i \leq n-1$ [5] , during the last three slots of patching period i, T sends e_i, $\overline{e}_i$, e'_i, which receivers D_k^ℓ with $\ell > i$ had missed during β_i. These transmissions result in D_1^{i+1}, D_3^{i+1} obtaining all messages. This explains why the n-th patching period consists of only two slots: the only receivers still requiring some messages are D_2^n, D_4^n, D_5^n. We conclude that by the end of the n-th patching period all receivers know all messages. Hence T satisfies $B(\phi)$ with delay one. □

Conversely, consider any coding scheme T' that satisfies $B(\phi)$ with delay one. We will exhibit a satisfying truth assignment for ϕ. Similarly to the proof of Theorem 1, in order to prove the reverse direction, we will again rely on two key properties of any such T'. The first property stems from the fact that T' introduces zero delay during the variable periods. We will show that any such T' must employ scheduling (and not coding) during the variable periods, and that this scheduling has to be structured in a certain way to guarantee that T' introduces no delay. In effect, the structure of the scheduling will allow us to extract unambiguous truth values for the variables in the formula ϕ. Moreover these values will be consistent among clauses, as forced by our consistency gadgets.

The second property stems from the fact that T' introduces zero delay during the clause periods. We will show that this can only happen if (a) coding is used during the clause periods, and (b) certain messages are received by certain receivers by the beginning of the clause periods. In effect, using the truth assignment derived from the variable periods, our choice gadgets and our clause constraint gadgets will translate reception of these messages into at least one true literal in every clause.

We formalize the intuition above in the following propositions. We first extend our previous notation of $E_i^{t_1 \ldots t_T}$ to include all time steps $1 \leq t_1 \leq t_T \leq 5$. We now show a technical but useful lemma concerning properties of T' during the variable periods.

Lemma 1. *Any coding scheme T' that satisfies $B(\phi)$ with delay one is rate-optimal and sends a single message at every time step j for $1 \leq j \leq 5n$. Moreover T' sends exactly 3 messages during every β_i, with the messages sent during the first two time slots of β_i being resent (in some order) during the last two time slots of β_i. After β_i, these messages will not be sent again before (potentially) time $5n+1$.*

Proof. Trivially, T' must be rate-optimal. Suppose that at time step $5(i-1)+k$, for any $1 \leq i \leq n$ and $1 \leq k \leq 5$, a (linear or nonlinear) function of at least two messages is sent. Then receiver D_k^i will delay, as he has received nothing so far.

For the second part of the lemma, we need to show that $E_i^{12345} \cap E_j^{12345} = \emptyset$ for all $1 \leq i \neq j \leq n$. Also, we must show that $E_i^{12} = E_i^{45}$.

Since D_1^1 receives during the first three time slots of every β_i, and the scheme is rate-optimal, $E_i^{123} \cap E_j^{123} = \emptyset$ for all $1 \leq i < j \leq n$. Similarly, since D_3^1 receives during the last three time slots of all β_i, and T' is rate-optimal, $E_i^{345} \cap E_j^{345} = \emptyset$ for all $1 \leq i < j \leq n$. Therefore we need show that $E_i^{12} \cap E_j^{45} = \emptyset$ for all $1 \leq i \neq j \leq n$. Equivalently we just need to show that $E_i^{12} = E_i^{45}$.

Observe that, for all i, D_4^i receives $E_i^{45} \cup E_{i+1}^{123} \cup \ldots \cup E_n^{123}$, and T' is rate-optimal. Hence $E_i^{45} \cap E_j^{123} = \emptyset$ for all $j > i$.

Moreover, for example for $i = 1$, $|E_1^{45} \cup E_2^{123} \cup \ldots \cup E_n^{123}| = 3n-1$. Hence D_4^1 knows $3n-1$ messages by time $5n$. Since T' satisfies $B(\phi)$ and D_1^1 does not receive after time $5n$, D_1^1 must know all $3n$ messages by time $5n$. Since D_1^1 and D_4^1 know the same $3n-3$ messages corresponding to $E_2^{123} \cup \ldots \cup E_n^{123}$, it follows that $E_1^{45} \subset E_1^{123}$. Since $E_1^3 \not\subset E_1^{12}$ as D_1^1 receives

[5] Recall that the last patching period consists of only two time slots.

during all three time steps 1,2,3, and $E_1^3 \not\subset E_1^{45}$ as D_3^1 receives during all three time steps 3,4,5, we conclude that $E_1^{12} = E_1^{45}$. The same argument holds for all $i > 1$.

□

W.l.o.g., assume that T' schedules the three messages $e_{x_i}, e_{y_i}, e_{z_i}$ during the first three time slots of β_i, in this order. By Lemma 1, these messages will not be rescheduled before time $5n$, so for the sake of clarity, we may relabel them as $e_i, \overline{e}_i, e'_i$ respectively. We define the following truth assignment. For $1 \leq i \leq n$, if T' applied Scheduling 1 for β_i during β_i, x_i is set to true, else if T' used Scheduling 2 for β_i during β_i, x_i is set to false. Notice that Lemma 1 guarantees that any T' applied one of these two schedulings indeed during β_i.

Similarly to Lemma 1, the following proposition presents necessary and sufficient conditions for the clause periods to be completed with delay one.

Proposition 10. *Let $c_j = (\ell_i \vee \ell_a \vee \ell_b)$ be* any *clause. In any scheme T' that satisfies $B(\phi)$ with delay one, C_3^j has received at least one of $\overline{e}_i, \overline{e}_a, \overline{e}_b$ by time $5n$.*

Before we give the proof of Proposition 10, we show how it concludes the second direction of our reduction.

Corollary 2. *If T' is a coding scheme that satisfies $B(\phi)$ with delay one then ϕ is satisfiable.*

Proof. Consider any clause $c_j = (\ell_i \vee \ell_a \vee \ell_b)$. By Proposition 10, T' is such that C_3^j has received at least one of $\overline{e}_i, \overline{e}_a, \overline{e}_b$ by time $5n$. W.l.o.g., assume C_3^j received $\overline{e}_i$. If C_3^j received this message at time $5(i-1)+4$, then x_i appears in c_j and T' used Scheduling 1 for β_i. Hence our truth assignment set x_i to true. Otherwise, if C_3^j received $\overline{e}_i$ at time $5i$, then $\overline{x}_i$ appears in c_j and T' used Scheduling 2 for β_i. Hence our truth assignment set x_i to false. In either case, our truth assignment for x_i satisfies c_j. Since Proposition 10 applies to all C_3^j for $1 \leq j \leq m$, there is (at least) one literal for every clause that is set to true by our truth assignment. Hence ϕ is satisfiable. □

We now give the proof of Proposition 10.

Proof. Consider the 3 receivers C_1^j, C_2^j, C_3^j corresponding to c_j. By Lemma 1, under any T', at the beginning of γ_j, each of C_1^j, C_2^j, C_3^j knows the $3n-9$ messages that correspond to the $n-3$ variables that do not appear in c_j. Further, C_1^j knows $\{e_i, \overline{e}_i, e_a, \overline{e}_a, e_b, \overline{e}_b\}$, C_2^j knows $\{e_i, e'_i, e_a, e'_a, e_b, e'_b\}$ and C_3^j knows exactly one of $e_i, \overline{e}_i$, one of $e_a, \overline{e}_a$, and one of $e_b, \overline{e}_b$.

Suppose that C_3^j did not receive any of $\overline{e}_i, \overline{e}_a, \overline{e}_b$. Then he received e_i, e_a, e_b. Now at the first slot of γ_j where C_1^j, C_2^j, C_3^j receive simultaneously, there is no way to avoid delay: C_3^j needs exactly one single message either from $\{\overline{e}_i, \overline{e}_a, \overline{e}_b\}$, or one from $\{e'_i, e'_a, e'_b\}$, while C_1^j needs no single message from the first set and C_2^j needs no single message from the second set.

On the other hand suppose that C_3^j received at least one of $\overline{e}_i, \overline{e}_a, \overline{e}_b$, w.l.o.g. $\overline{e}_i$. Then at time $5n+6(j-1)+1$, a function of $\overline{e}_i$ and exactly one from $\{e'_i, e'_a, e'_b\}$ would suffice to incur zero delay to all 3 receivers. □

t	C_1^1	C_2^1	C_3^1	C_1^2	C_2^2	C_3^2	C_1^3	C_2^3	C_3^3	D_1^1	D_2^1	D_3^1	D_4^1	D_5^1	D_1^2	D_2^2	D_3^2	D_4^2	D_5^2	D_1^3	D_2^3	D_3^3	D_4^3	D_5^3	D_1^4	D_2^4	D_3^4	D_4^4	D_5^4
1	✓	✓	x	✓	✓	x	✓	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2	✓	x	x	✓	x	x	✓	✓	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3	x	✓	x	x	✓	x	✓	✓	✓	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4	x	x	✓	x	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	✓	x	x	x	x	x	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6	✓	✓	x	✓	✓	x	✓	✓	x	✓	✓	x	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x
7	✓	x	x	✓	x	x	✓	x	x	✓	✓	x	✓	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x
8	x	✓	x	x	✓	x	x	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x
9	x	x	✓	x	x	x	x	x	x	x	x	✓	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x
10	x	x	x	x	x	✓	x	x	✓	x	x	✓	x	x	x	x	✓	✓	✓	x	x	x	x	x	x	x	x	x	x
11	✓	✓	x	✓	✓	x	✓	✓	x	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	x	x	x	x	x	x	x	x	x
12	✓	x	x	✓	x	x	✓	x	x	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	✓	x	x	x	x	x	x	x	x
13	x	✓	x	x	✓	x	x	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	x	x	x	x	x	x
14	x	x	✓	x	x	✓	x	x	x	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	✓	x	x	x	x	x	x
15	x	x	x	x	x	x	x	x	✓	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	✓	✓	x	x	x	x	x
16	✓	✓	✓	✓	✓	✓	✓	✓	x	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	x	x	x	x
17	✓	✓	✓	✓	✓	✓	✓	x	x	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	✓	x	✓	✓	✓	✓	x	x	x
18	✓	✓	✓	✓	✓	✓	x	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	x
19	x	x	x	x	x	x	x	x	✓	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	✓	x
20	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	x	x	x	x	✓	✓	✓
21	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
22	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
23	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
24	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
25	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
26	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
27	x	x	x	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
28	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
29	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
30	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
31	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
32	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
33	x	x	x	x	x	x	✓	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
34	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
35	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
36	x	x	x	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
37	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
38	x	x	x	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
39	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x	✓	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
41	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
42	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
43	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
44	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x	x	x	x	x	x
45	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	✓	x	x	x	x	x	x	x	x	x	x
46	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
47	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
48	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
49	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	x	x	x	x	x
50	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	✓	x	x	x	x	x
51	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓
52	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓
53	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓	✓	✓	✓
54	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	✓
55	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	✓

Table 8: $P(\phi)$ for $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$ from Section 4.