

# On Delay Performance Gains From Network Coding

(Invited Paper)

Atilla Eryilmaz

Laboratory for Information and Decision Systems  
Massachusetts Institute of Technology  
Cambridge, MA, 02139  
Email: eryilmaz@mit.edu

Asuman Ozdaglar and Muriel Médard

Electrical Engineering and Computer Science Department  
Massachusetts Institute of Technology  
Cambridge, MA, 02139  
Email: {asuman, medard}@mit.edu

**Abstract—** This paper analyzes the gains in delay performance resulting from network coding. We consider a model of file transmission to multiple receivers from a single base station. Using this model, we show that gains in delay performance from network coding with or without channel side information can be substantial compared to conventional scheduling methods for downlink transmission.

## I. INTRODUCTION

With the introduction of third-generation cellular systems over the last decade, there has been both a significant increase in the capacity of wireless networks and a growing use of wireless communication for data transmission. An essential feature of the newly emerging wireless networks is the transmission of files to multiple (potentially heterogeneous) receivers, as exemplified by transmission of video or music files.

While the most common approach to data transmission builds on the scheduling approach, where information is transmitted to one of multiple receivers as a function of their channel conditions, it has also been recognized that broadcasting to multiple receivers using *network coding* may be more efficient for utilizing the capacity of the network (c.f. [1, 8, 7, 9]). Although these throughput gains may appear to imply gains in delay through Little's law, this is not the case since coding is performed over large blocks and each packet in the block must await the completion of the whole block before it can be decoded. Despite considerable practical interest in the use of network coding in wireless communication systems, gains in delay performance resulting from network coding relative to traditional scheduling have not been analyzed or quantified. The best setting to investigate such gains is the *rateless* transmission scenario, where data of fixed length is to be communicated over the channel. In this context, the comparison in delay performance between traditional schemes and network coding is performed through the completion times of the *whole* data.

The objective of this paper is to develop a model to study delays in file downloads with network coding and quantify the gains resulting from network coding relative to the traditional scheduling methods. We consider downlink transmission of (multiple) files from a single base station to multiple receivers with varying channel conditions. The varying channel conditions are modeled as stochastic changes in ON/OFF state of the channel. We analyze the model both when Channel

Side Information (CSI) about the state of receiver channels is available to the base station and when transmission must be carried without such information.

Our analysis shows that, as well as the already well-understood capacity gains, network coding leads to significant improvement in delay performance both with and without CSI. This is potentially important, since depending on the application, delay performance may be critical to the satisfaction of the users. Equivalently, with network coding more users can be supported with the same delay performance of scheduling.

Our paper differs from existing work in this area by explicitly modeling delay performance in file downloads and allowing for transmission without CSI, and to the best of our knowledge, presents the first quantification of gains in delay performance resulting from network coding. Previous research has instead focused on either optimal scheduling with time-varying channel conditions (see [13, 14]), or on the capacity gains from network coding (see [11, 5, 6, 12, 10, 15]) under various different scenarios.

The rest of the paper is organized as follows: Section II describes the model for downloading (multiple) files to multiple receivers and introduces different transmission strategies considered. Section III focuses on broadcasting a single file and investigates the performance of network coding compared to the performance of delay optimal scheduling strategies. We show analytically and through simulations that using network coding results in significant gains in delay performance relative to delay optimal scheduling strategies both with or without CSI. In Section IV, we focus on the multiple unicast scenario. We show that in the presence of CSI, network coding achieves optimal performance when no coding is allowed across separate files. Moreover, it provides considerable delay gains compared to scheduling when CSI is not available. This last point of significant delay gains is particularly interesting given that network coding does not provide any capacity gains for the single hop unicast scenario. We complete with final remarks and ideas for future work in Section V.

## II. SYSTEM MODEL

In this work, we consider the cellular downlink scenario where the base station holds a set of files,  $\mathcal{F}$ . The set of receivers is denoted by  $\mathcal{N}$ . File  $f \in \mathcal{F}$  is demanded by the

set  $\mathcal{N}_f \subseteq \mathcal{N}$  of receivers<sup>1</sup>. We are interested in the minimum average time required to complete the download of all the files by all the interested receivers, where the transmissions have to be done over a time varying channel. We seek an answer to this problem with and without the availability of CSI at the base station and with and without the possibility of linear coding in the manner we will describe.

A given file  $f \in \mathcal{F}$  is composed of  $K_f$  packets, where Packet- $k$  of file  $f$  is referred to as  $\mathbf{P}_{f,k}$ , which is a vector<sup>2</sup> of length  $m$  over a finite field  $\mathbb{F}_q$ . It is assumed that transmissions occur in time slots, each of which is of duration just long enough to accommodate a single packet transmission. The channel between the base station and the  $i^{\text{th}}$  receiver is a randomly varying ON/OFF channel. We let  $C_i[t] \in \{0, 1\}$  denote the state of user  $i$ 's channel in slot  $t$ . We assume that Receiver- $i$  successfully receives the packet transmitted at slot  $t$  if  $C_i[t] = 1$ , and it cannot receive anything if  $C_i[t] = 0$ . We will take each  $C_i[t]$  to be a Bernoulli random variable with mean  $c_i$  that are independent across time and across receivers. The channels of different receivers can in general be asymmetric. However, in parts of the subsequent analysis we will restrict our attention to symmetric cases in order to have tractable formulations. The presence of CSI implies that the channel state vector,  $\mathbf{C}[t]$ , is known at the transmitter at the beginning of slot  $t$ . The system model is depicted in Figure 1.

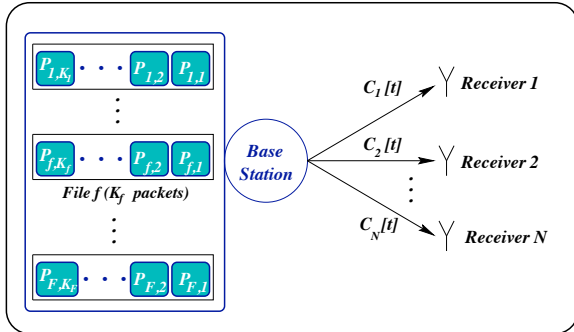


Fig. 1. System model

Let us use  $\mathbf{P}[t]$  to denote the packet chosen for transmission in slot  $t$ . If the base station is not allowed to code, then at any given slot it must transmit a single packet from one of the files. Thus, we have  $\mathbf{P}[t] \in \{\mathbf{P}_{f,k}\}_{\substack{k=1, \dots, K_f \\ f \in \mathcal{F}}}$ . This is the typical mode of transmission considered in literature. We will refer to this mode as the *Scheduling Mode* (or simply *Scheduling*).

If coding is allowed, then in a slot, say  $t$ , any linear combination of the packets can be transmitted. Specifically, we have

$$\mathbf{P}[t] = \sum_{f \in \mathcal{F}} \sum_{k=1}^{K_f} a_{f,k}[t] \mathbf{P}_{k,f},$$

where  $a_{f,k}[t] \in \mathbb{F}_q$  for each  $f \in \mathcal{F}$  and  $k \in \{1, \dots, K_f\}$ . The transmitter chooses the coefficients  $\{a_{f,k}[t]\}$  at every time slot

<sup>1</sup>We will occasionally use  $F, N$  and  $N_f$  to denote the cardinalities of the sets  $\mathcal{F}, \mathcal{N}$  and  $\mathcal{N}_f$ , respectively.

<sup>2</sup>We will consistently use **boldface** letters to denote vectors.

$t$ . This mode of transmission will be referred to as the *Coding Mode* (or simply *Coding*) henceforth.

Given the above model, we are interested in minimizing the amount of time necessary for all the files to be transmitted to all the interested receivers. We will refer to this metric as the *completion time*. In particular, we would like to find answers to the following questions:

- If we restrict ourselves to scheduling policies, then what is the policy that minimizes the average completion time?
- If coding is allowed, then what is the best policy and what is its delay performance?
- How do these two policies' performances compare?

We seek answers to these questions under various scenarios. In Section III, we focus on the case of a single file demanded by all the receivers (broadcast scenario). Then, in Section IV, we will consider the other extreme of each receiver demanding a unique file (multiple unicasts scenario). We will discuss extensions in Section V.

### III. BROADCASTING A SINGLE FILE

In this section, we are concerned with the transmission of a single file to all the receivers. Since  $|\mathcal{F}| = 1$ , we will drop the subscript  $f$  in our notation, and denote Packet- $k$  as  $\mathbf{P}_k$  and the size of the file as  $K$ . We will study the minimum mean completion time of the file using coding in Section III-A, where we will observe the asymptotic optimality of coding over all possible strategies. Then, we will characterize the optimal scheduling strategy with and without CSI in Section III-B. Comparison of the findings will be presented in Section III-C along with a discussion on the strengths and weaknesses.

#### A. Coding with and without CSI

It has been shown in the literature that linear coding is sufficient to achieve the maximum achievable rate for multicast networks [8]. Noticing that the broadcast scenario is a special instance of a multicast transmission, we consider the set of policies where the transmitted packet in slot  $t$  is given by

$$\mathbf{P}[t] = \sum_{k=1}^K a_k[t] \mathbf{P}_k, \text{ with } a_k[t] \in \mathbb{F}_q \text{ for each } k \in \{1, \dots, K\}.$$

We will consider the following randomized strategy (c.f. [4]).

**RANDOMIZED BROADCAST CODING (RBC):**  
 While (File is incomplete)  
     Pick  $a_k[t]$  uniformly at random from  $\mathbb{F}_q$  for each  $k$ ;  
     Transmit  $\mathbf{P}[t] = \sum_{k=1}^K a_k[t] \mathbf{P}_k$ ;  
      $t \leftarrow t + 1$ ;

Each receiver keeps the incoming packets that it could receive and then decodes all the packets  $\{\mathbf{P}_k\}_{k=1, \dots, K}$  as soon as  $K$  linearly independent combinations of the packets are collected (c.f. [4] and references therein). Random linear coding arguments imply that the expected number of slots before  $K$  linearly independent combinations can be collected with RBC is given by

$$\sum_{k=1}^K \frac{1}{(1 - (1/q)^k)}.$$

This expression can be upper-bounded by  $Kq/(q-1)$ , which in turn can be made close to  $K$  even with reasonably low values of  $q$ . Thus, for all practical purposes, for a large enough field size  $q$ , it is sufficient for each receiver to be active  $K$  slots on average before it can decode the whole file. Notice that information theoretically it is impossible to send the file with less than  $K$  transmissions, and so *RBC asymptotically (in  $q$ ) achieves the best possible performance over all strategies.*

Another important issue is the overhead related with this mode of transmission. Coding requires  $\lceil K \log_2 q \rceil$  bits of overhead to contain the coefficients of the associated linear combination, whereas the packet size is  $\lceil m \log_2 q \rceil$  bits. Thus, for  $m \gg K$ , the overhead is negligible. Henceforth, we will consider this scenario, and ignore the overhead.

Notice that RBC is not only easy to implement, but also requires no knowledge of the channel state vector, and asymptotically achieves the smallest mean completion time over all policies. We will see in Section III-B that the optimal scheduling policy is much more difficult to characterize, even for the symmetric channel conditions.

Next, we find the mean completion time expression for RBC. Let us define the random variable  $Y_i$  as the number of slots before Receiver- $i$ 's channel is ON  $K$  times, for  $i = 1, \dots, N$ . Then, we can claim that the mean completion time is equal to  $\mathbb{E} \left[ \max_{i \in \{1, \dots, N\}} Y_i \right]$ , which is given in the next proposition.

**Proposition 1.** *Let  $T_1$  denote the completion time of the optimal coding policy given above. Then, we have*

$$\mathbb{E}[T_1] = K + \sum_{t=K}^{\infty} \left[ 1 - \prod_{i=1}^N \left( \sum_{\tau=K}^t \binom{\tau-1}{K-1} \bar{c}_i^{(\tau-K)} c_i^K \right) \right],$$

where  $\binom{n}{m}$  gives the number of combinations of size  $m$  of  $n$  elements, and  $\bar{c}_i \triangleq (1 - c_i)$ .

*Proof:* The proof follows from combinatorial arguments and is omitted due to space constraints. ■

### B. Scheduling Mode

In this mode, unlike in the coding mode, the presence or lack of CSI affects the performance. Hence, these two cases will be studied separately. Throughout, we will assume symmetric channels for tractability.

1) *Scheduling without CSI:* To minimize the load of uplink transmission which is typically the bottleneck in cellular systems, we assume that the transmitter receives feedback from each receiver only at the time when it has just received the whole file. Notice that in this case, all packets have equal priority. Also, since the channels are independent and identically distributed (i.i.d.) over time and users, one of the optimal scheduling policies is *Round Robin* (RR), where Packet- $k$  is transmitted in time slots  $(mK + k)$  for  $m = 0, 1, \dots$  until all the receivers get the file.

To compute the mean completion time of the above RR scheduler, we define  $X_k^i$  to be the number of transmissions of

$\mathbf{P}_k$  before it is received by Receiver- $i$ . Then,

$$Y^i \triangleq \max_{k \in \{1, \dots, K\}} \{KX_k^i + k\}$$

gives the time slot when Receiver- $i$  receives the whole file. Finally,  $T_2 \triangleq \max_{i \in \{1, \dots, N\}} Y^i$  gives the completion time of the algorithm. Its mean is described in the next proposition.

**Proposition 2.** *Under symmetric channel conditions (i.e.  $c_i = c \in (0, 1)$  for all  $i$ ), we have*

$$\frac{\mathbb{E}[T_2]}{K} = \gamma + \sum_{t=1}^{\infty} \left[ 1 - (1 - (1 - c)^t)^{KN} \right],$$

for some  $\gamma \in (1/2, 1)$ .

*Proof:* The upper bound of 1 for  $\gamma$  is due to the fact that  $k \leq K$ . The lower bound of 1/2 follows from stochastic coupling arguments and heavily relies on the symmetry of the channel distributions. In particular, consider a sample path of the channel state process,  $\omega \triangleq (\mathbf{C}[1], \mathbf{C}[2], \dots)$ . We use  $i(\omega)$  to denote the receiver that was the last to complete the file, and  $k(\omega)$  to denote the index number of the last packet that Receiver- $i(\omega)$  received. With our earlier notation,  $Y(\omega)$  gives the completion time of the file at Receiver- $i(\omega)$  under the given sample path. Also, notice that we have  $Y(\omega) = X_{k(\omega)}^{i(\omega)} K + k$ , for some integer  $X_{k(\omega)}^{i(\omega)}$  that depends on  $\omega$ .

Next, for each sample path  $\omega$  that leads to  $k(\omega) \in \{1, \dots, \lfloor K/2 \rfloor\}$ , we will construct another sample path  $\tilde{\omega}$  that has the same probability of occurrence as  $\omega$ , but leads to  $Y(\tilde{\omega}) = X_{k(\omega)}^{i(\omega)} K + (K - k(\omega))$ . This implies that

$$\mathbb{E}[Y] \geq \frac{(K+1)}{2} + K \mathbb{E}[\max_{i,k} X_k^i]. \quad (1)$$

The construction of  $\tilde{\omega} = (\tilde{\mathbf{C}}[1], \tilde{\mathbf{C}}[2], \dots)$  follows the following rule:

$$\tilde{C}_j[rK + l] = \begin{cases} C_j[rK + (K - l)], & \text{if } r = X_{k(\omega)}^{i(\omega)}(\omega), \\ & j = i(\omega), \\ & l \in \{k(\omega), K - k(\omega)\}, \\ C_j[rK + l], & \text{otherwise.} \end{cases}$$

It is easy to see that under symmetric conditions this sample path has the properties listed above.

Next, we would like to find the second term in (1). Due to i.i.d. assumptions,  $X_k^i$  are also i.i.d. with distribution  $\mathbb{P}(X_k^i = m) = (1 - c)^{m-1} c$ ,  $m = 1, 2, \dots$ . Since this distribution is independent of  $i$  and  $k$ , we can compute

$$\mathbb{E}[\max_{i,k} X_k^i] = \sum_{t=1}^{\infty} \left[ 1 - (1 - (1 - c)^t)^{KN} \right]. \quad (2)$$

The proof is complete once (2) is substituted into (1). ■

2) *Scheduling with CSI:* Before we characterize the optimal scheduling rule with CSI, we demonstrate the suboptimality of scheduling compared to coding with the following simple example.

*Example 1:* Consider the case of  $K = 3$  and  $N = 3$ , i.e. three packets are to be broadcasted to three receivers.

Consider the channel realizations  $\mathbf{C}[1] = (0, 1, 1)$ ,  $\mathbf{C}[2] = (1, 0, 1)$ ,  $\mathbf{C}[3] = (1, 1, 0)$ , and  $\mathbf{C}[4] = (1, 1, 1)$ . Thus, in the first four slots, each receiver can hear the transmission three times. The optimal scheduling rule would transmit  $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$  in the first three slots, leaving Receiver- $i$  in demand for Packet- $i$  in the fourth slot. Clearly, no scheduling rule can ever complete the file download at all three receivers in the fourth slot. With coding, on the other hand, the following transmissions will complete the transmissions:  $(\mathbf{P}_1 + \mathbf{P}_2), (\mathbf{P}_2 + \mathbf{P}_3), (\mathbf{P}_1 + \mathbf{P}_3), (\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3)$  (see Table I). It is not difficult to see that coding will never require more slots than is necessary for scheduling for all other realizations. Hence, we achieve strictly better completion times with coding.

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
$R_1$	–	$\mathbf{P}_2   (\mathbf{P}_2 + \mathbf{P}_3)$	$\mathbf{P}_3   (\mathbf{P}_1 + \mathbf{P}_3)$	$?   (\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3)$
$R_2$	$\mathbf{P}_1   (\mathbf{P}_1 + \mathbf{P}_2)$	–	$\mathbf{P}_3   (\mathbf{P}_1 + \mathbf{P}_3)$	$?   (\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3)$
$R_3$	$\mathbf{P}_1   (\mathbf{P}_1 + \mathbf{P}_2)$	$\mathbf{P}_2   (\mathbf{P}_2 + \mathbf{P}_3)$	–	$?   (\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3)$

**TABLE I.** Demonstration of *Example 1*:  $R_i$  corresponds to Receiver- $i$ , ‘–’ denotes OFF channel states, and the entry  $\mathbf{a} | \mathbf{b}$  gives the optimal transmissions with scheduling and coding, respectively. With scheduling, no choice of  $\{\mathbf{P}_i\}$  in slot 4 can complete the file at all the receivers for the given channel realization.

#### OPTIMAL SCHEDULING FOR SYMMETRIC CHANNELS:

We use Dynamic Programming to find the characterization of the optimal scheduling policy for symmetric channel conditions, i.e.  $c_i = c$  for all  $i \in \{1, \dots, N\}$ . Given  $\mathbf{C}[t]$ , the scheduler can choose any one of the packets  $\{\mathbf{P}_1, \dots, \mathbf{P}_K\}$  for transmission. A little thought reveals the need of memory about the previous receptions at each of the receivers. For this purpose, we define  $M_{i,k}[t]$  to be the memory bit associated with Packet- $k$  and Receiver- $i$ . In particular,  $M_{i,k}[t] = 1$  (or 0) implies that Receiver- $i$  has not received (or has received) Packet- $k$  in the slots  $1, \dots, t-1$ . Moreover, we will use  $\mathbb{M}[t]$  to denote the matrix of memory bits  $[M_{i,k}[t]]_{i=1, \dots, N}^{k=1, \dots, K}$ .

We let  $\Pi$  denote the set of feasible stationary policies that can be implemented by the base station. Each policy  $\pi \in \Pi$  defines a mapping from the pair  $(\mathbb{M}[t], \mathbf{C}[t])$  to the set  $\{1, \dots, K\}$  describing the packet to be sent at time  $t$ . Note that the policy is stationary in the sense that it is only a function of the matrix and channel conditions at the time. The i.i.d. nature of the arrivals and departures imply that this is the optimal policy among all policies, including those that are time dependent.

To characterize the optimal policy we let  $J^\pi(\mathbb{M}, \mathbf{C}) = \mathbb{E}[\# \text{ slots to reach } \theta \text{ with policy } \pi \mid \mathbb{M}[0] = \mathbb{M}, \mathbf{C}[0] = \mathbf{C}]$ , where  $\theta$  denotes the zero matrix. Then,  $J^*(\mathbb{M}, \mathbf{C}) \triangleq \min_{\pi \in \Pi} J^\pi(\mathbb{M}, \mathbf{C})$  is the minimum completion time of the optimal algorithm if it starts from  $\mathbb{M}$  and the first channel is  $\mathbf{C}$ . Also,  $\pi^*(\mathbb{M}, \mathbf{C}) \triangleq \arg \min_{\pi \in \Pi} J^\pi(\mathbb{M}, \mathbf{C})$  gives the optimal policy.

Observe that once we solve  $J^*(\mathbb{M}, \mathbf{C})$  for all  $\mathbf{C}$ , we can compute  $J^*(\mathbb{M}) \triangleq \mathbb{E}_{\mathbf{C}}[J^*(\mathbb{M}, \mathbf{C})]$ , where the expectation is over the channel realizations. Thus,  $J^*(\mathbb{M})$  denotes the mean completion time of the optimal algorithm starting from

$\mathbb{M}$ . Hence, we are interested in  $J^*([1]_{N \times K})$  where  $[a]_{N \times K}$  denotes the all  $a$  matrix of dimensions  $N \times K$ .

Before we write the recursion for  $J^*(\mathbb{M}, \mathbf{C})$ , let us define the function  $f(\cdot)$  where  $\hat{\mathbb{M}} = f(\mathbb{M}, \mathbf{C}, k)$  implies that

$$\begin{aligned} \hat{M}_{i,k} &= M_{i,k} - M_{i,k} C_i & \forall i \in \{1, \dots, N\}, \\ \hat{M}_{i,j} &= M_{i,j} & \forall i \in \{1, \dots, N\}, j \neq k. \end{aligned}$$

This function describes the next state of the memory matrix given that Packet- $k$  is served and the channel matrix is  $\mathbf{C}$  in the current slot. Then, we can write the following recursion:

$$J^*(\mathbb{M}, \mathbf{C}) = \arg \min_{k \in \{1, \dots, K\}} \{J^*(f(\mathbb{M}, \mathbf{C}, k)) + \mathbf{1}_{\{\mathbb{M} \neq \theta\}}\},$$

where  $\mathbf{1}_{\{A\}}$  is the indicator function of the event  $A$ .

The monotone nature of the  $f(\cdot)$  function enables us to compute  $J^*(\mathbb{M}, \mathbf{C})$  and  $\pi^*(\mathbb{M}, \mathbf{C})$  recursively starting from the base state  $J^*(\theta) = 0$  (c.f. [2]). This DP formulation characterizes the optimal policy and its performance, and in theory it can be computed starting from a  $1 \times 1$  matrix and increasing  $N$  and  $K$  successively. However, as  $N$  and  $K$  grows, the necessary number of operations grows exponentially and quickly becomes impossible to handle. Thus, we propose an efficient heuristic policy below and simulate its performance for comparison.

#### HEURISTIC POLICY:

We have observed in the above discussions that the optimal scheduling rule has a complicated structure. Yet, it is possible to find practical scheduling algorithms that will perform close to the optimal. Here, we describe a heuristic policy which we believe will provide near optimal performance.

At any given time slot  $t$ , let us denote the set of nodes with an ON channel (also called the set of *active receivers*) by  $\mathcal{A}[t] \triangleq \{i \in \{1, \dots, N\} : C_i[t] = 1\}$ . Under the symmetric conditions that we assumed, the packet that would provide the most *benefit* should intuitively be transmitted over the channel. We propose that the benefit of a packet be measured in the number of nodes in  $\mathcal{A}[t]$  that has not yet received that packet. The underlying idea is to transfer the maximum number of useful packets over the channel at any given time. These remarks point to the heuristic algorithm given next.

#### HEURISTIC BROADCAST SCHEDULING (HBS):

```

If ( $t = 1$ )
     $M_{i,k}[t] \leftarrow 1$  for all  $k \in \{1, \dots, K\}, i \in \{1, \dots, N\}$ ;
While  $\left( \sum_{k=1}^K \sum_{i=1}^N M_{i,k}[t] > 0 \right)$ 
     $\mathcal{K}[t] \triangleq \{k \in \{1, \dots, K\} : \exists i \in \mathcal{A}[t] \text{ with } M_{i,k}[t] = 1\}$ ;
    If ( $\mathcal{K}[t] \neq \emptyset$ )
         $\mathcal{T}[t] \triangleq \arg \max_{k \in \mathcal{K}[t]} \sum_{i \in \mathcal{A}[t]} M_{i,k}[t]$ ;
        Pick a  $k^* \in \mathcal{T}[t]$ ;
         $M_{i,k^*}[t] \leftarrow 0$  for all  $i \in \mathcal{A}[t]$ ;
        Transmit Packet- $k^*$  over the channel at slot  $t$ ;
     $t \leftarrow t + 1$ ;

```

In the algorithm, each packet in  $\mathcal{K}[t]$  has at least one receiver with an ON channel in slot  $t$  which demands that packet.

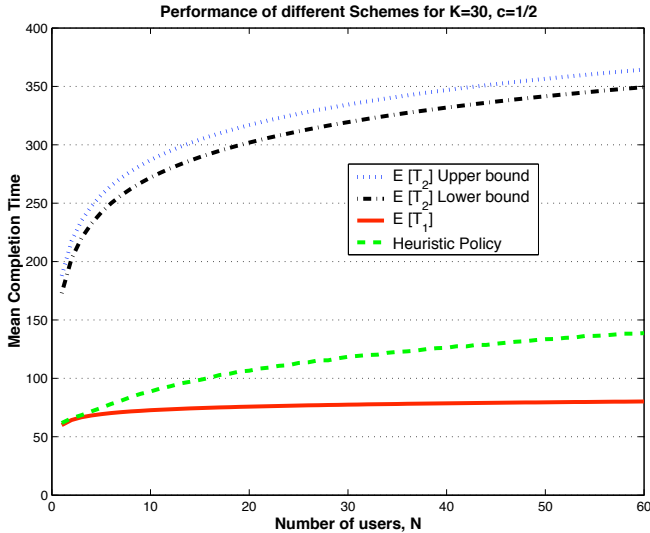


Fig. 2. Performance of the two schedulers for a file of length 30 packets and  $c = 1/2$ .

Clearly, those packets that are not in  $\mathcal{K}[t]$  should not be chosen for transmission. If  $\mathcal{K}[t] \neq \emptyset$ , then we define  $\mathcal{T}[t]$  to be the set of packets in  $\mathcal{K}[t]$  that yield the most benefit in slot  $t$ . Then, a packet from  $\mathcal{T}[t]$  is picked for transmission in slot  $t$ . In our simulations, we considered a random picking of one of the packets in  $\mathcal{T}[t]$ . However, the performance can be slightly improved by using more sophisticated methods. For example, for  $N = 2$ , the packet picked from  $\mathcal{T}[t]$  may be chosen amongst those packets that has already been received by the OFF receiver. Then, every time a receiver is ON, it will receive a useful packet until all its packets are complete. Thus, this algorithm gives the optimal policy for  $N = 2$ . The generalization of the picking method to general  $K$  is complicated and requires increasing memory to operate. On the other hand, the complexity of HBS at each iteration of the loop is  $O(KN)$  and requires no extra memory, and hence it is relatively easy to implement.

### C. Comparison

In this section, we perform numerical computations and simulations to compare the performance of various schemes we have discussed so far. A typical scenario is depicted in Figure 2, where a file of size 30 packets is to be transmitted to a varying number of receivers, where each channel is ON or OFF equiprobably at every time slot.

The figure demonstrates the strength of the coding policy to the scheduling policy with and without CSI. We further observe that as  $N$  increases the advantage of using coding improves.

In this section, we have seen that either with or without CSI, coding provides a considerable gain in the mean delay to download a given file to multiple receivers over a time-varying medium. Moreover, its operation is significantly easier than the scheduling policy. However, it requires an additional decoding operation at the receivers, which may or may not be critical

depending of the file sizes and the computation capacity of the receivers.

## IV. MULTIPLE UNICAST TRANSMISSIONS

In this section, we consider the scenario where  $N$  receivers with symmetric channel conditions demand a unique files, i.e.  $F = N$ . In this case, it is not clear whether coding will have the dominating behavior as it did in the broadcast scenario. Again, the availability of CSI is important. We will first study the scheduling case and then move on to the coding case.

### A. Scheduling for Multiple Unicasts

1) *Scheduling without CSI*: Without CSI, the obvious optimal scheduling is again Round Robin, except that it must be performed across files and across packets in each file. In particular, in the first round the first packet of each file is transmitted one after another, and in the next round the second packets are transmitted consecutively. When the end of a file is reached, we move to the first packet and continue until all the packets of a file is received by its receiver. Only then we remove that file from the RR scheduler and continue with the remaining ones.

The mean delay performance of this scheduling rule is easy to compute using recursive arguments, which is omitted here since it does not add any significant insights to our analysis.

2) *Scheduling with CSI*: Here, the constraint is to serve at most one receiver at every time slot. This problem is a special case of a problem studied by Tassiulas and Ephremides in [13] with no arrivals to the system. The following policy is introduced in [13].

**LONGEST CONNECTED QUEUE (LCQ):**

```

t ← 0;
Qi ← Ki for all i ∈ {1, ⋯, N};
Do
  t ← t + 1;
  i* [t] ← arg max1 ≤ i ≤ N {Ci[t]Qi};
  if (Ci*[t] ≠ 0)
    Transmit Pi*, Qi*;
    Qi* ← max(0, Qi* - 1);
While (∑i=1N Qi > 0);
Return t; // Completion time

```

In the policy,  $Q_i$  is used both as a pointer to the index of the next packet to be transmitted to Receiver- $i$ , and also as the number of packets yet to be transmitted to Receiver- $i$ . Thus, LCQ is a myopic policy that favors the receiver with the maximum number of packets to be received among all connected receivers. We repeat the result of [13] for future reference.

**Proposition 3 ([13]).** *Under symmetric channel conditions (i.e.  $c_i = c$  for all  $i$ ), LCQ is minimizes the delay over all scheduling policies. In other words,*

$$T^{LCQ} \preceq_{st} T^\pi,$$

where  $T^{LCQ}$  denotes the completion time under the LCQ policy and  $\pi$  is any other feasible scheduling policy<sup>3</sup>.

This result is very strong and implies that  $\mathbb{E}[T^{LCQ}] \leq \mathbb{E}[T^\pi]$  for any feasible scheduling policy  $\pi$ .

### B. Coding for Multiple Unicasts

A deep understanding of achievable rates for multiple unicast sessions in a network is still an open problem. In general, it is not clear whether network coding should be performed, and if it should what the strategy must be. We will tackle this problem for the downlink model at hand.

We define the set of *coding classes* that partitions  $\mathcal{F}$  (or equivalently  $\mathcal{N}$ ) into  $J$  subsets. We use  $\mathcal{C}_j$  to denote the files (or equivalently receivers) in Class- $j$ . We set the restriction that only those files within the same class will be linearly coded with random coefficients as in RBC, while files of different classes will not be mixed. Notice that for each class, say  $\mathcal{C}_j$ , this strategy effectively results in a single file of length  $K^j \triangleq \sum_{f \in \mathcal{C}_j} K_f$  that is demanded by  $b_j \triangleq |\mathcal{C}_j|$  distinct receivers. Hence, the multiple unicasts problem is converted into a special case of multiple multicasts with each multicast having a disjoint set of receivers. Notice that the description of the strategy is yet incomplete, because we must describe how to “schedule” the transmissions of different classes. We will investigate this question with and without CSI.

1) *Coding without CSI*: In this case, as in Section III-B.1, we assume that each receiver informs the base station when it can decode its own file, which in turn implies that it can decode all the files within its class. The optimal policy is again going to be of the form of Round Robin over the coding classes. We will consider the case of  $b_j = b$  and  $K^j = K$  equal for all  $j$ . If  $J$  denotes the total number of coding classes, then only a combination from  $\mathcal{C}_j$  will be transmitted in slot  $(mJ + j)$  for  $m = 0, 1, \dots$  until all the receivers get their files.

Notice that the analysis of the RR scheduler of Section III-B.1 does not directly apply to this case, because here once all the receivers of a class, say  $\mathcal{C}_j$ , decode their file, then that class can be extracted from the round robin cycle. Nevertheless, similar analysis based on recursive formulations can be used for this setting. This analysis is omitted here due to space constraints. We remark that without CSI the gain in grouping subsets of users as described above is only due to the decreasing size of the cycles as groups complete their receptions. If the period of each cycle were kept constant at its starting value of  $J$  throughout the operation, then grouping would have no effect on the average delay performance, because in such a scenario we would be comparing the expected number of slots before  $K$  ON channels are observed to  $1/b$  times the expected number of slots before  $bK$  ON channels are observed.

2) *Coding with CSI*: In the presence of CSI, we must determine the optimal partitioning of the files  $\{\mathcal{C}_j\}$ , and also find the optimal scheduling policy across these classes. The following proposition finds the optimal policy using stochastic coupling arguments.

**Proposition 4.** *Under the symmetric channel conditions (i.e.  $c_i = c$  for all  $i \in \mathcal{N}$ ), the mean delay minimizing partitioning is obtained when  $b_j = 1$  for all  $j$ , and the optimal policy is to implement LCQ.*

*Proof:* Consider any given partitioning of the files, say  $\mathcal{P} = \{\mathcal{C}_j\}_{j=1}^J$ , and let  $\pi_{\mathcal{P}}$  denote the optimal policy for this partitioning, which is not known in general. Also, let  $T^{\pi_{\mathcal{P}}}$  be the random variable that denotes the completion time of all the files under the policy  $\pi_{\mathcal{P}}$ . In other words,  $T^{\pi_{\mathcal{P}}}$  is the first slot when each receiver in Class- $j$  received  $K^j$  linear combinations of the packets from within their class, for all  $j$ . We use  $\omega = (\mathbf{C}[1], \mathbf{C}[2], \dots)$  to denote a sample path of the channel state process. Notice that the policy and  $\omega$  determines  $T^{\pi_{\mathcal{P}}}(\omega)$ .

Next, we will define a new policy  $\tilde{\pi}$  and show that it satisfies  $T^{\tilde{\pi}}(\omega) \leq T^{\pi_{\mathcal{P}}}(\omega)$  for all feasible  $\omega$ . For a given  $\omega$ , if  $\pi_{\mathcal{P}}$  serves Class- $j$  in slot  $t$ , then  $\tilde{\pi}$  will send only the head-of-line packet of one of the connected receivers in the same class which received the minimum service so far. In other words, amongst the connected receivers in Class- $j$ , only the receiver that has the maximum number of remaining packets is served. Notice that this policy does not do any coding, and hence requires Receiver- $f$  in Class- $j$  to successfully receive  $K_f$  packets of its file instead of  $K^j$  packets as in  $\pi_{\mathcal{P}}$ .

To see that  $T^{\tilde{\pi}}(\omega) \leq T^{\pi_{\mathcal{P}}}(\omega)$ , observe that whenever Class- $j$  is served under  $\pi_{\mathcal{P}}$ , at most one packet (or one degree of freedom) can be received by each receiver in that class. Thus, before all of its receivers can decode their own packet, Class- $j$  must be served at least  $K^j$  times. But, with  $\tilde{\pi}$  we can send a single degree of freedom to one of the connected receivers in Class- $j$  whenever that class is served under  $\pi_{\mathcal{P}}$ . Since for each  $f \in \mathcal{C}_j$ , only  $K_f$  degrees of freedom are required for Receiver- $f$  with  $\tilde{\pi}$ , all the receivers complete their reception when Class- $j$  is served  $K^j = \sum_{f \in \mathcal{C}_j} K_f$  times. These arguments prove that for any feasible sample paths the completion of the new policy is not larger than that of  $\pi_{\mathcal{P}}$  for any partition  $\mathcal{P}$ .

To complete the proof, we need to show that  $T^{LCQ} \preceq T^{\tilde{\pi}}$ . To that end, we note that  $\tilde{\pi}$  is actually a scheduling policy, where at each slot a single packet is transmitted over the channel. Thus, an application of Proposition 3 completes the proof. ■

### C. Comparison

In this section, we compare the typical performance of various policies for reasonable parameters. We take  $b_j = b$  for all  $j$  and  $K_f = K$  for all  $f \in \mathcal{F}$ . Moreover, we let  $K = 30$  and  $N = F = 12$  and study the mean completion time behavior of the scheduling and coding strategies with and without CSI. Regarding the channel connectivity statistics, we assume that  $c_i = 1/2$  for all the channels. Figure 3 depicts the simulation results of the policies discussed above for varying number of classes. In the figure, we observe that the performance of the LCQ scheduler serves as a lower bound as we have proved in Proposition 4. Since the optimal coding policy is not specified for an arbitrary  $b$ , in the simulation we use the following heuristic policy: at each time slot among the classes

<sup>3</sup> $\preceq_{st}$  is a stochastic ordering as described in [13].

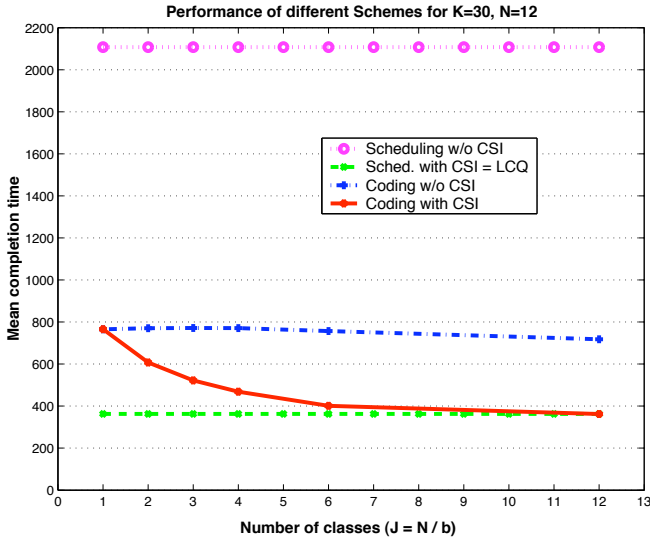


Fig. 3. Mean completion time of various strategies for  $c = 1/2$ .

with the maximum number of connected receivers, the policy serves the class with the maximum degrees of freedom yet to be transmitted. This policy, when  $b = 1$  is the same as the LCQ policy. For this policy, we observe that the mean delay value achieved decreases to half its value when  $b$  is decreased from 12 to 1. We also observe that in agreement with our arguments, the performance of the coding without CSI improves as  $b$  decreases, but this decrease is rather insignificant.

Without CSI, the performance of scheduling is significantly worse than the coding solution. In this particular case, we observe almost a threefold delay with scheduling as opposed to coding. Given that the single-hop multiple unicasts scenario does not improve the capacity of the channel, the presence of such a considerable delay gain is particularly striking.

The fact that both with and without CSI the performance of the coding strategy improves as  $b$  goes to one implies that for unicast transmissions, it is best to code within files, but not across them.

## V. DISCUSSIONS AND FUTURE DIRECTIONS

In this work, we have introduced a simple model where delay performance of network coding can be investigated and compared to the traditional method of scheduling. Under various scenarios, we have identified the optimal policies and derived analytical expressions for the delay expressions. We observed that with easily implementable coding strategies, significant delay gains can be obtained.

We have analyzed two scenarios: the case when all receivers demand a single file (broadcast case), and the case when each receiver demands a different file (multiple unicasts case). Under both scenarios, when the channel side information is not available we have observed the advantage of using random coding strategies over pure scheduling approaches. However, when CSI is present, it turned out that coding gives considerable gains for the broadcast scenario, whereas scheduling is the best policy for the multiple unicasts scenario under

symmetric channel conditions. The assumption of perfect CSI is unrealistic for actual systems. Instead, if we assume the availability of CSI for a fraction of the time, then by using random coding, significant gains can still be achieved, whereas scheduling will be more vulnerable to the lack of CSI.

One rule of thumb we obtained from our analysis was to code across packets within a file, but to avoid coding across files. This observation will help us in finding the optimal policy for the general multiple multicast scenario.

An important extension is to consider more general channel statistics. More interestingly, one can consider a fully connected network where each of the  $N$  nodes wants to broadcast its own file to the rest of the receivers over time varying channels (see [3] for a variation of this problem).

Another direction is to use our framework to analyze the performance of various strategies under differentiated services. In particular, we want to specify priority classes with varying delay constraints, and find the policy that minimizes a weighted sum of the mean completion times.

## REFERENCES

- [1] R. Ahlswede, Ning Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, July 2000.
- [2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. M.I.T. Press, McGraw-Hill Book Company, London, England, 2001.
- [3] S. Deb and M. Médard. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. In *Proceedings of the 43rd Annual Allerton Conference on Communication, Control and Computing*, October 2004.
- [4] T. Ho. Networking from a network coding perspective. PhD thesis, MIT, 2004.
- [5] D. Katabi, S. Katti, W. Hu, H. Rahul, and M. Médard. The importance of being opportunistic: On practical network coding for wireless environments. In *Proceedings of Annual Allerton Conference on Communication, Control and Computing*, October 2005.
- [6] D. Katabi, S. Katti, W. Hu, H. Rahul, and M. Médard. On practical network coding for wireless environments. In *Proceedings of IEEE International Zurich Seminar on Communications*, 2006.
- [7] R. Koetter and M. Médard. Beyond routing: An algebraic approach to network coding. *IEEE Transactions on Information Theory*, 11:782–795, October 2003.
- [8] S.-Y. R. Li, R. W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49:371–381, February 2003.
- [9] D. S. Lun, M. Médard, R. Koetter, and M. Effros. Further results on coding for reliable communication over packet networks. In *Proceedings of IEEE International Symposium for Reliable Communication over Packet Networks*, pages 1848–1852, 2005. (An extended version has been submitted to the *IEEE Transactions on Information Theory*).
- [10] D. S. Lun, P. Pakzad, C. Fragouli, M. Médard, and R. Koetter. An analysis of finite-memory random linear coding on packet streams. In *Proceeding of NetCod*, 2006.
- [11] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee. Achieving minimum cost multicast: A decentralized approach based on network coding. In *Proceedings of IEEE Infocom*, pages 1607–1617, 2005.
- [12] S. Shakkottai P. Gupta S. Bhadra. Min-cost selfish multicast with network coding. In *Proceeding of NetCod*, 2006.
- [13] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39:466–478, March 1993.
- [14] B. S. Tsybakov. File transmission over wireless fast fading downlink. *IEEE Transactions on Information Theory*, 48:2323–2337, August 2002.
- [15] Yunnan Wu. Network coding for multicasting. PhD thesis, Princeton University, November 2005.