# Sharing Information in Time-Division Duplexing Channels: A Network Coding Approach

Daniel E. Lucani, Muriel Médard
RLE, MIT
Cambridge, Massachusetts, 02139
Email: {dlucani,medard}@mit.edu

Milica Stojanovic
Northeastern University
Boston, Massachusetts, 02115
Email: millitsa@mit.edu

David R. Karger
CSAIL, MIT
Cambridge, Massachusetts, 02139
Email: karger@mit.edu

***Abstract*—We study random linear network coding for time-division duplexing channels for sharing information between nodes. We assume a packet erasure channel with nodes that cannot transmit and receive information simultaneously. Each node will act as both a sender of its own information and a receiver for the information of the other nodes. When a node acts as the sender, it transmits coded data packets back-to-back before stopping to wait for the receivers to acknowledge the number of degrees of freedom, if any, that are required to decode correctly the information. This acknowledgment comes in the header of the coded packets that are sent by the other nodes. We study the mean time to complete the sharing process between the nodes. We provide a simple algorithm to compute the number of coded packets to be sent back-to-back depending on the state of the system. We present numerical results for the case of two nodes sharing data and show that the mean completion time of our scheme is close to the performance of a full duplex network coding scheme and can outperform full duplex schemes with no coding.**

## I. Introduction

The use of network coding in channels in which time division duplexing is necessary, i.e. when a node can only transmit or receive, but not both at the same time was considered in Reference [2]. This type of channel is typically called half-duplex. However, Reference [2] used the more general term time division duplexing (TDD) to emphasize that the nodes do not use the channel in any pre-determined fashion, but instead may *vary* the amount of time allocated to transmit and receive. Some examples of time division duplexing channels are infrared devices (IrDA), and underwater acoustic modems. Other applications may be found in very high latency channels, e.g. in satellite, and deep space communications, because of the considerable delay experienced by feedback.

Reference [2] studied the case of transmitting a block of $M$ data packets through a link using random linear network coding. The objective was to minimize the mean time to complete transmission of the block of packets. Reference [3] extended this analysis for the problem of energy consumption of the scheme, showing that there exists, under the minimum energy criterion, an optimal number of coded data packets to be transmitted back-to-back before stopping to wait for an acknowledgment (ACK). This reference also showed that choosing the number of coded data packets to optimize mean completion time, as in [2], provides a very good trade-off between energy consumption and completion time.

Reference [4] further extended this work for the case of broadcast. In this setting, a transmitter with $M$ data packets has the objective to broadcast those packets reliably to $N$ receivers. This reference assumes that the receivers are not allowed to cooperate to share their received coded packets in order to decode, i.e. each receiver must decode the information from the coded packets sent directly from the transmitter.

Previous references have considered the case in which only one node has information to transmit, i.e. without considering that more than one node might need to transmit or share its information with others. We now analyze the problem of two nodes that want to share disjoint information under the TDD constrain. We present a random linear network coding scheme that operates in a similar fashion to that in [2]. In fact, the problem studied in [2] is a subset of the work presented in this paper. In particular, we study the mean completion time of the two-node sharing scheme for TDD and compare it to a full duplex network coding scheme and a scheduling policy. Finally, we provide a simple algorithm to determine the number of coded data packets to be transmitted back-to-back before stopping. We show that this algorithm converges in a small number of iterations.

The paper is organized as follows. In Section II, we outline the problem, we study the mean completion time of our scheme, and we introduce a search algorithm to minimize the mean completion time. Section III presents some comparison schemes and provides expressions for their mean completion time for the case of two nodes sharing disjoint information. Section IV provides numerical results for different scenarios, comparing the different schemes. Section V extends the analysis and algorithm to the case of $N$ nodes in the network. Conclusions are summarized in Section VI.

## II. Random Network Coding for Sharing Information in TDD channels: two nodes

Two nodes want to share information through a TDD channel, i.e. a channel in which nodes can transmit and receive, but not both at the same time. Each node $i$ has $M_i$ data packets or disjoint random linear combinations that he wants to share with the other node, as in Figure 1. A node $i$ can


Fig. 1. Nodes with disjoint information. Node $i$ has $M_i$ disjoint data packets (or independent random linear combinations of packets). Both nodes want to have all the information at the end of the exchange.

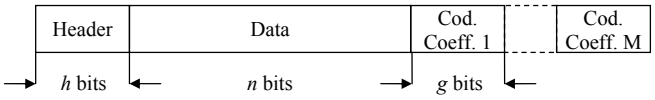


Fig. 2. [2] Structure of coded data packet: a header of size $h$ bits, $n$ data bits, $M$ coding coefficients (Cod. Coeff.) of size $g$ bits each.

transmit information at a given data rate $R_i$ [bps] to the other node. We assume an independent packet erasure channel. Each node will act as both sender and receiver of information. When a node operates as the sender, it uses random linear network coding [1] to generate coded data packets. Each coded data packet contains a linear combination of the $M$ data packets of $n$ bits each, as well as the random coding coefficients used in the linear combination. Each coefficient is represented by $g$ bits. For encoding over a field size $q$, we have that $g = \log_2 q$ bits. A coded packet is preceded by an information header of size $h$. Thus, the total number of bits per packet is $h + n + gM$. Figure 2 shows the structure of each coded packet considered in our scheme.

The node acting as a sender at some point can transmit coded packets back-to-back before stopping to wait for the other node to transmit its own packets and acknowledge how many degrees of freedom (dof) it still requires to have all the information. In general, there is no explicit ACK packet. The acknowledgement to node $i$ comes piggybacked in the header of each coded packet sent from node $j$, unless no coded packet has to be sent. As in [2], we assume that the field size $q$ is large enough so that the expected number of successfully received packets at the receiver, in order to decode the original data packets, is approximately $M_i$ for transmissions from node $i$.

Transmission begins at one of the nodes, say node 1 with $M_1$ information packets, which are encoded into $N_{(M_1,M_2,0)} \geq M_1$ random linear coded packets, and transmitted. After receiving the coded packets, node 2 which has $M_2$ information packets, generates $N_{(i_1,i_2,1)} \geq M_2$ random linear coded packets and transmits them, where the state $(i_1, i_2, t)$ represents the $i_1$ dofs required by node 2 to decode the information of node 1, the $i_2$ dofs required by node 1 to decode the information of node 2, and the node that is acting as transmitter $t$, with $t = 0, 1$ for nodes 1 and 2, respectively. Node 2 includes in the header of each coded packet an ACK of the dofs needed by 2 to decode the information that 1 is trying to send, i.e. $i_1$. If all $M_1$ packets are decoded successfully by node 2, node 1 becomes a receiver and will send only an explicit ACK packet in the following rounds stating how many dofs it requires to decode, because node 1 does not need to send any more data. Otherwise, node 1 sends

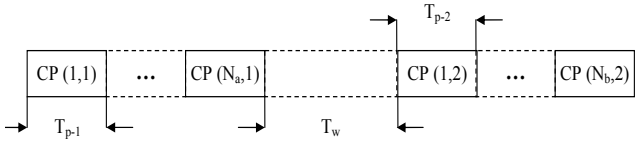


Fig. 3. Network coding TDD scheme for sharing packets between two nodes. Note that the feedback comes piggybacked in the coded packets $CP$, and that $CP(\cdot, i)$ corresponds to a coded packet sent from node $i$.

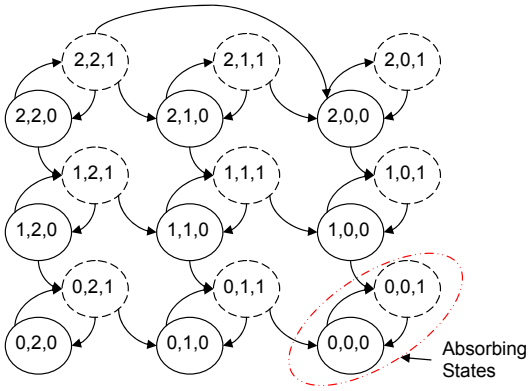


Fig. 4. Example of the Markov chain for a block size of $M_1 = M_2 = 2$.

$N_{(i_1,i_2,0)}$ coded packets and piggybacks an ACK in each header informing node 2 about how many dofs are missing at node 1, i.e. $i_2$. This process is repeated until all packets have been shared successfully by both nodes. We assume that all ACKs suffer no erasures. We also assume that if node $i$ has completed its transmission to another another $j$, and $i$ gets enough information to decode the data from $j$, then node $i$ can stop the transmission process, unless new information is available for transmission. These is a consequence of the ACKs suffering no erasures. We are interested in the optimal number $N_{(i_1,i_2,t)}$ of coded packets to be transmitted back-to-back.

Figure 3 illustrates the time window allocated to the system to transmit $N_{(i_1,i_2,1)}$ coded packets. Each coded packet $CP(1,1)$, $CP(2,1)$, etc. is of duration $T_{p-1}$ and each coded packet $CP(1,2)$, $CP(2,2)$ is of duration $T_{p-2}$, for packets sent from nodes 1 and 2, respectively. The waiting time $T_w$ is equivalent to the propagation time $T_{prop}$ in this problem.

The process is modelled as a Markov chain. The states $(s_1, s_2, t)$ are defined by the number of dofs required, $s_k$ at receiver $k$, to successfully decode all packets, and the node $t$ that acts as transmitter in this state. Thus, the states range from $(M_1, M_2, 1)$ to $(0, 0, 0)$. This is a Markov chain with $(M_1 + 1)(M_2 + 1) - 2$ transient states and two recurrent states (state $(0, 0, 0)$ and $(0, 0, 1)$). Finally, note that a transition occurs every time that a batch of back-to-back coded packets is received at one receiver.

Figure 4 provides an example for a block size of 2 packets at each node. We have highlighted in this figure the absorbing states. Note that not all possible transitions from one state to the others have been included in this figure. However, any transient state $(s_1, s_2, 0)$ can only have

transitions to a state $(s_1', s_2', 1)$. Similarly, any transient state $(s_1, s_2, 1)$ can only have transitions to a state $(s_1', s_2', 0)$. In Figure 4, this translates into no transitions from transient dashed states to other dashed states, or from transient solid-line states to other solid-line states. This observation is crucial in the development of an algorithm to compute the values of $N_{(s_1,s_2,t)}$.

The transition probabilities from state $(s_1, s_2, t)$ to state $(s_1', s_2', t')$ are

$$P_{(i,j,t)\to(i',j',t')} = P\big(X_1(n)=i', X_2(n)=j', T(n)=t' | X_1(n-1)=i, X_2(n-1)=j, T(n-1)=t\big)$$

where $X_i(n)$ is the number of dof required at receiver $i$ at the end of transmission $n$, and $T(n)$ is the designated transmitter at time $n$.

Given the characteristics of the Markov chain, we have that

$$P_{(s_1,s_2,t)\to(s_1',s_2',t')} = \tag{1}$$

$$\begin{cases} P\left(s_1'|s_1, N_{(s_1,s_2,t)}\right) & \text{if , } s_2 = s_2', \, t = 0, \, t' = 1 \\ P\left(s_2'|s_2, N_{(s_1,s_2,t)}\right) & \text{if , } s_1 = s_1', \, t = 1, \, t' = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $N_{(s_1,s_2,t)}$ represents the number of coded packets sent to produce the transition. Note that $P\left(s_j'|s_j, N_{(s_i,s_j,t)}\right)$ have a similar structure to the transition probabilities studied in [2]. As in [4], the main difference is that the value of the number of coded packets sent back-to-back is no longer associated with the starting state of a particular node, but with a value determined from the state of the system, i.e. $N_{(s_i,s_j,t)}$.

For $0 < s_j' < s_j$, this can be translated into

$$P\left(s_j'|s_j, N_{(s_1,s_2,t)}\right) = \tag{3}$$

$$f(s_j, s_j')(1 - Pe_j)^{s_j - s_j'} {Pe_j}^{N_{(s_1,s_2,t)} - s_j + s_j'} \tag{4}$$

where

$$f(s_j, s_j') = \begin{cases} \binom{N_{(s_1,s_2,t)}}{s_j - s_j'} & \text{if } N_{(s_1,s_2,t)} \geq s_j, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and $Pe_j$ represents the erasure probability of a coded packet sent from node $j$. For $s_j = s_j' > 0$ the expression for the transition probability reduces to: $P\left(s_j|s_j, N_{(s_1,s_2,t)}\right) = {Pe_j}^{N_{(s_1,s_2,t)}}$. Note that for $P\left(0|0, N_{(s_1,s_2,t)}\right) = 1$. Finally, for $s_j' = 0$

$$P\left(s_j'=0|s_j, N_{(s_1,s_2,t)}\right) = 1 - \sum_{s_j'=1}^{s_j} P\left(s_j'|s_j, N_{(s_1,s_2,t)}\right). \tag{6}$$

### A. Expected Time for completing transmission

The expected time for completing the sharing process of all data packets between the nodes constitutes the expected time of absorption, i.e. the time to reach states $(0, 0, 0)$ or $(0, 0, 1)$ for the first time, given that the initial state is either $(M_1, M_2, 0)$ or $(M_1, M_2, 1)$, depending on which node starts transmitting. This can be expressed in terms of the expected time for completing the transmission given that the Markov Chain is in state is $(s_1, , s_2, t)$, $T_{(s_1,s_2,t)}$ , $\forall s_1 = 0, 1, ..M_1, \forall s_2 = 0, 1, ..M_2, \forall t = 0, 1$. For our scheme, $T_{p-i} = \frac{h+n+gM_i}{R_i}$, $T_{prop}$ is the propagation time, and the waiting time $T_w$ shown in Figure 3 is $T_w = T_{prop}$.

Let us define $T^{(i,j,t)}$ as the time it takes to transmit $N_{(i,j,t)}$ coded data packets and reach the other node. It is easy to show that $T^{(i,j,t)} = N_{(i,j,t)}T_{p-t} + T_w$. The mean completion time when the system is in state $(i, j, t)$ is given by

$$T_{(i,j,t)} = T^{(i,j,t)} + \sum_{(i',j',t')} P_{(i,j,t)\to(i',j',t')} T_{(i',j',t')}. \tag{7}$$

We can express this in vector form as $\bar{T} = \bar{\mu} + P\bar{T}$, where $\bar{T} = [T_{(i,j,t)}]$, $\bar{\mu} = [T^(i,j,t))]$ and $P$ is the corresponding transition probability. Thus, the mean completion time starting as every state is given by $\bar{T} = [I - P]^{-1}\bar{\mu}$. Since we are interested in the mean completion time when we start at states $(M_1, M_2, 0)$ or $(M_1, M_2, 1)$, we can use Cramer's rule as

$$T_{(M_1,M_2,t)} = \frac{\det\left(\Gamma \leftarrow_{(M_1,M_2,t)} \bar{\mu}\right)}{\det(\Gamma)} \tag{8}$$

where $\Gamma = I - P$, and the notation $\Gamma \leftarrow_{(M_1,M_2,t)} \bar{\mu}$ represents a matrix that has all columns as the $\Gamma$ matrix except the column corresponding to state $(M_1, M_2, t)$ which is substituted by the vector $\bar{\mu}$, $t$ can take value 0 or 1.

### B. Minimizing The Mean Completion Time

The expected time for each state depends on all the expected times for the previous states. However, optimizing the values of all $N_{(i,j,t)}$ is not as straightforward as the recursive method used in [2]. Also, note that there are $2(M_1 + 1)(M_2 + 1)$ states in our Markov chain, and that we have $2(M_1 + 1)(M_2 + 1) - 2$ integer variables that we need to optimize.

Let us consider exploiting the structure of the problem to determine an algorithm to estimate the values of $N_{(i,j,t)}, \forall i = 1, ..., M_1, \forall j = 1, ..., M_2, \forall t = 0, 1$. Let us express the mean completion time for states with $t = 0$ as

$$T_{(i,j,0)} = T^{(i,j,0)} + \sum_{i'} P_{(i,j,0)\to(i',j,1)} T_{(i',j,1)} \tag{9}$$

and the mean completion time for states with $t = 1$ as

$$T_{(i,j,1)} = T^{(i,j,1)} + \sum_{j'} P_{(i,j,1)\to(i,j',0)} T_{(i,j',0)}. \tag{10}$$

Notice that the mean completion time for any state of the form $(i, j, 0)$ depends on the mean completion time of states of the form $(i, j, 1)$ but not on states of the form $(i, j, 0)$. Thus, we can substitute equation (10) into (9), and use the

fact that $T^{(i,j,t)} = N_{(i,j,t)}T_{p-t} + T_{prop}$ to obtain

$$\begin{aligned} T_{(i,j,0)} &= N_{(i,j,0)}T_{p-0} + 2T_{prop} \\ &+T_{p-1}\left[\sum_{i'} N_{(i',j,1)}P_{(i,j,0)\to(i',j,1)}\right] \\ &+\sum_{i',j'} T_{(i',j',0)}P_{(i,j,0)\to(i',j,1)}P_{(i',j,1)\to(i',j',0)}. \end{aligned} \tag{11}$$

A similar expression can be found for $T_{(i,j,1)}$.

Note that these expressions are redolent of the mean completion time for a link presented in [2]. Reference [2] showed that the cost for transitioning from the current state to other states was of the form $N_iT_p+2T_{prop}+T_{ack}$. In our expressions, we have a similar cost with some changes, namely there is no cost for transmitting an ACK packet because the ACKs are piggybacked in the coded packets. Also, we have an additional term, i.e. $T_{p-1}\left[\sum_{i'} N_{(i',j,1)}P_{(i,j,0)\to(i,j,1)}\right]$, which represents the mean additional waiting time for one transmitter due to the transmission of the other node.

Let us define $\hat{N}_{(i,j,t)}(n)$ as the estimate for $N_{(i,j,t)}$ at step $n$ of the algorithm, and $P_{(i,j,0)\to(i',j,1)}(n)$ and $P_{(i,j,0)\to(i,j',1)}(n)$ are the transition probabilities based on the estimates $\hat{N}_{(i,j,t)}(n)$ for the $n$-th step.

*Algorithm 1:* Search algorithm for case of two nodes

- `STEP 1: INITIALIZE`
  -Set $\hat{N}_{(i,j,1)}(0) = j$ and $\hat{N}_{(i,j,0)}(0) = i$.
  -Set $n = 1$.
- `STEP 2: TRANSMISSION FROM NODE 1 TO NODE 2:`
  `FOR` $j' = 1, 2, ..., M_2$
  Compute $\hat{N}_{(i,j',0)}(n), \forall i = 1, ..., M_1$ to minimize the completion time of a TDD link, as in [2], with transition cost $\hat{N}_{(i,j',0)}(n)T_{p-0} + 2T_{prop} + T_{p-1}\left[\sum_{i'} \hat{N}_{(i',j',1)}(n-1)P_{(i,j',0)\to(i',j',1)}(n)\right]$
  `END FOR`
- `STEP 3: TRANSMISSION FROM NODE 2 TO NODE 1:`
  `FOR` $i' = 1, 2, ..., M_1$
  Compute $\hat{N}_{(i',j,1)}(n), \forall j = 1, ..., M_1$ to minimize the completion time of a TDD link, as in [2], with transition cost $\hat{N}_{(i',j,1)}(n)T_{p-0} + 2T_{prop} + T_{p-1}\left[\sum_{j'} \hat{N}_{(i',j',0)}(n)P_{(i',j,0)\to(i',j',0)}(n)\right]$
  `END FOR`
- `STOPPING CRITERIA:`
  `IF` $\hat{N}_{(i,j,t)}(n) = \hat{N}_{(i,j,t)}(n-1)$, $\forall i, j, t$
  Stop
  `ELSE`
  $n = n + 1$, and go to Step 2.
  `END IF`

The proposed algorithm computes the $N_{(i,j,t)}$ by using the search algorithm for a link, as in [2], with the appropriate costs. The algorithm is iterative and has two phases. The first one tries to solve the problem of a link for the case in which the first transmitter is operating. This means that we are looking to optimize the variables $N_{(i,j,0)}, \forall i$ assuming the variables $N_{(i',j,1)}$ to be fixed and $j$ also fixed. We repeat the process for every value of $j$. After finding the optimal values for $N_{(i,j,0)}, \forall i, j$, the algorithm proceeds to the second phase, which is to compute the optimal values of $N_{(i,j,1)}$ given the new values of $N_{(i,j,0)}$, keeping these last values fixed. The algorithm stops when $N_{(i,j,t)}$ becomes stable, i.e. when the current iteration provides the same result as the previous iteration.

Let us emphasize that the $N_{(i,j,t)}$'s do not need to be computed in real time. They can be pre-computed and stored in the receiver as look-up tables, as explained in [2]. Thus, the computational load on the nodes is minimal, because they only have to choose the appropriate $N_{(i,j,t)}$'s from the tables considering channel conditions at the time of transmission.

## III. Comparison Schemes

In this section, we present two full duplex schemes as a basis for comparison. First, we consider a full duplex network coding scheme that minimizes the mean completion time of the sharing process. Secondly, we extend the work of [5], which deals with broadcast, to determine the mean completion time for a round robin scheduling policy for sharing information between two nodes. This policy considers no coding of the data packets, no channel state information, and nodes that only ACK when they have received all information.

We consider that both schemes send the ACK in the header of the coded data packets that each node sends. As in [5], we restrict the analysis to independent symmetric channels, i.e. $Pe_1 = Pe_2$ and $T_{p-0} = T_{p-1} = T_p$, and no erasures in the ACKs for tractability. Note that we had no such restrictions in our TDD network coding scheme and will not have it for the full duplex network coding scheme. Our contribution to the work of broadcast scheduling policies in [5] includes 1) considering the effect of $T_{prop}$, $T_p$, 2) the characterization of a full duplex channel, and 3) resetting the analysis to match the problem of sharing information between two nodes.

*1) Data Sharing with Network Coding in Full Duplex Channel (DSNC Full Duplex):* Each node generates random linear combinations of its original $M_i$ data packets, and sends those coded packets back-to-back through the channel to the other node. We assume that both nodes start transmitting at the same time to reduce the completion time of the sharing process. This problem can be modelled through a Markov chain with states $(i, j)$, where $i$ and $j$ represent the dofs required by the to decode at node 1 and 2, respectively. Since both nodes start transmitting at the same time, and we assume the packets to take the same time to be transmitted $T_p$, then transitions occur every arrival of a coded packet. The transition probabilities are modeled as $P_{(i,j)\to(i',j')} = P_{i\to i'}P_{j\to j'}$ where we assume independence of the channels. Note that

$$P_{i\to i'} = \begin{cases} Pe & \text{if } i = i' \neq 0, \\ 1 & \text{if } i = i' = 0, \\ 1 - Pe & \text{if } i = i' + 1, \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $M \geq i, i' \geq 0$.

Using a similar procedure as in previous sections, we can express the mean number of coded packets from each node to complete the sharing process in vector form as $\bar{T} = \bar{1} + P\bar{T}$, where $\bar{T} = [T_{(i,j)}]$ is the vector of mean completion times starting at each state $(i,j)$, $\bar{1} = [1]$ is a vector of all ones, and $P$ is the corresponding transition probability. Thus, the mean completion time for the sharing process is

$$E[T] = T_w + T_p \frac{\det\left(\Gamma \leftarrow_{(M,M)} \bar{1}\right)}{\det\left(\Gamma\right)} \tag{13}$$

where $T_w = 2T_{prop} + T_{delay}$ , $T_{delay} = T_h + T_p\left(1 - \mathtt{R}\left(T_{prop}, T_p\right)\right)$, $T_h = h/R$. Note that $T_{delay}$ represents the delay to send the ACK because it is piggybacked to the header of the coded packets. Note that the function $\mathtt{R}(x,y)$ returns the remainder of $x/y$.

*2) Round Robin Data Sharing in Full Duplex Channel (DSRR Full Duplex):* The objective is to transmit $M_1$ data packets from node 1 to node 2, and $M_2$ data packets from node 2 to node 1. We consider the simpler problem when $M_1 = M_2 = M$. We assume that both nodes start transmitting at the same time. Note that packet $k$ in the block of each node is transmitted every $(mM+k)T_p$ time units for $m = 0, 1, 2, ...$ until the other nodes gets all $M$ packets. The sharing process is completed when both nodes have received all information. Note that this problem is very similar to that in [5]. Using a similar analysis,

$$E[T] = T_w + T_p M\left(\gamma + E[\max_{i,k} X_k^i]\right) \tag{14}$$

where $1 + X_k^i$ is the number of transmissions of packet $k$ needed to reach node $i$ from the other node, $\gamma \in (1/2, 1)$, $T_w = 2T_{prop} + T_{delay}$ ,$T_{delay} = T_h + T_p\left(1 - \mathtt{R}\left(T_{prop}, T_p\right)\right)$, $T_h = h/R$, and

$$E[\max_{i,k} X_k^i] = \sum_{t=1}^{\infty}\left[1 - \left(1 - Pe^t\right)^{2M}\right]. \tag{15}$$

Note that $\gamma = 1$ and $\gamma = 1/2$ give us an upper and lower bound on the mean completion time, respectively.

*3) Round Robin Data Sharing in TDD Channel:* This approach is similar to DSRR Full Duplex but considering a TDD channel. Again, the objective is to transmit $M_1$ data packets from node 1 to node 2, and $M_2$ data packets from node 2 to node 1. We consider the simpler problem when $M_1 = M_2 = M$ and that each transmitter sends all $M$ packets before stopping to listen for a transmission of the other. The data packets also contain feedback indicating if the node should keep transmitting or if all packets have been received successfully at the other node. Note that this problem is very similar to that in [4]. Using a similar analysis, the mean completion time $E[T]$ is bounded by

$$E[T] \leq (T_w + 2T_p M)\left(1 + E[\max_{i,k} X_k^i]\right) \tag{16}$$

and

$$E[T] \geq (T_w + T_p M)\left(1 + E[\max_{i,k} X_k^i]\right) + T_p M \tag{17}$$

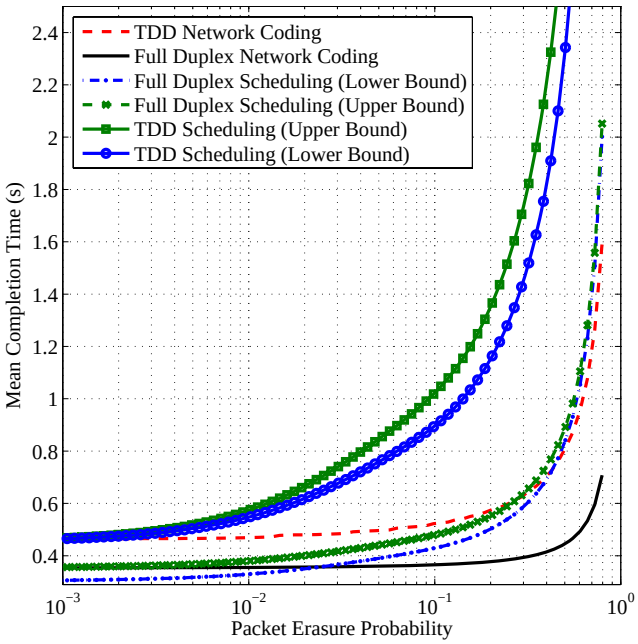


Fig. 5. Mean completion time for the TDD scheme choosing the $N_{(i,j,t)}$'s through the search algorithm proposed in Section II-B, two full duplex schemes, and a TDD scheme with no coding. We use the following parameters $R = 1.5$ Mbps, $h = 80$ bits, $g = 20$ bits, a block size per node of $M = 15$.

using the same definitions as in DSRR Full Duplex.

## IV. Numerical Results

This section provides numerical results that compare the performance of our network coding scheme for sharing disjoint information between two nodes in TDD channels. We consider a GEO satellite example where the propagation time $T_{prop} = 125$ ms [2], and data packets of size $n = 10,000$ bits. We assume symmetric uplink and downlink channels, i.e. $Pe_1 = Pe_2 = Pe$ and $R_1 = R_2 = R = 1.5$ Mbps. We compare the performance of the scheme in terms of mean completion time when the $N_{(i,j,t)}$'s are chosen to minimize the mean completion time using the proposed search algorithm under different packet erasure probabilities $Pe$. We consider that both nodes have he same number of packets at the start of the process, which means that $M_1 = M_2 = M$ data packets. We show that our TDD scheme can outperform a full duplex round robin scheduling scheme for large $Pe$. Also, our TDD scheme performs at most 3 dB above that of a full duplex network coding scheme. For small $Pe$, the difference between our scheme and the full duplex network coding is much less than 3 dB. Finally, we show that the number of iterations required for our algorithm to convergence is very small for a wide range of $Pe$.

Figure 5 shows the mean completion time for the TDD scheme choosing the $N_{(i,j,t)}$'s through the search algorithm proposed in Section II-B, two full duplex schemes, and a TDD scheme with no coding.

Figure 5 illustrates that choosing $N_{(i,j,t)}$'s using the search algorithm provides very good performance in terms of mean completion time for a wide range of packet erasure probabil-

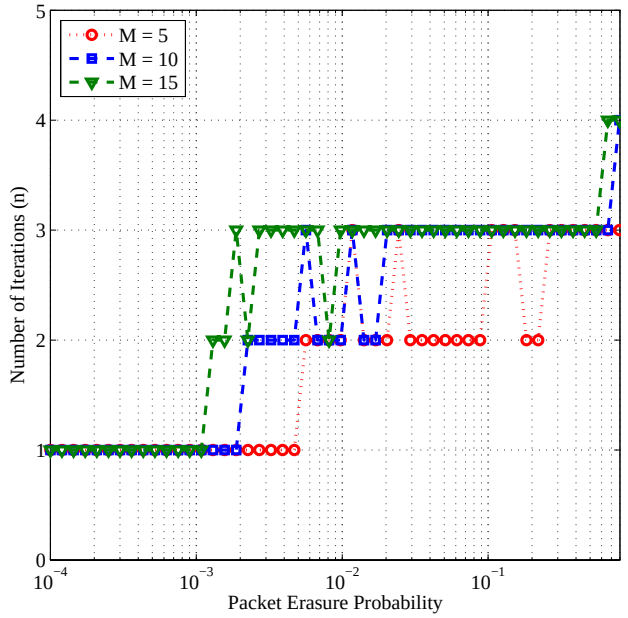


Fig. 6. Number of iterations of the algorithm to reach a stable solution under different packet erasure probabilities. We use the following parameters $R = 1.5$ Mbps, $h = 80$ bits, $g = 20$ bits.

ities, when compared to full duplex schemes and the TDD scheme with no coding. In general, our scheme outperforms the TDD with no coding. This difference is most noticeable at large $Pe$, but even at moderate schemes there is a clear advantage of our scheme. Also, note that at low packet erasure probabilities, our TDD scheme is only 1 dB away from the performance of the DSNC full duplex scheme, which is the optimal scheme in terms of mean completion time. Since we have packets to be transmitted from both nodes, we expected the difference between these two schemes to be around 3 dB, i.e. twice the completion time for the TDD scheme because it has half of the channels that the network coding full duplex scheme has. The main reason for this improvement is related to the relatively small number of coded packets that are being transmitted. Thus, the main cost in the completion time is the $T_{prop}$ of the packets. For very high packet erasure probabilities, our TDD scheme is closer to the expected 3 dB. In this case, the number of coded packets transmitted in order to decode the information is the dominant cost to the completion time.

Figure 5 also shows that for high packet erasures, $Pe \geq 0.4$, our TDD scheme using the proposed search algorithm to choose the $N_{(i,j,t)}$'s, outperforms a full duplex scheduling scheme (SDRR full duplex). This is clear because the lower bound on the mean completion time of the SDRR full duplex exceeds the mean completion time of our scheme at around $Pe \geq 0.4$. It is possible that we could outperform SDRR full duplex with our TDD scheme even for $Pe > 0.2$, which is the point at which the upper bound of SDRR full duplex intersects with the mean completion time of our scheme. Note that for $Pe = 0.8$ our TDD scheme outperforms SDRR full duplex by about 1.1 dB. Thus, even with a single channel for data and feedback, i.e. half of the resources, we can perform better than scheduling by tailoring coding and feedback appropriately.

Figure 5 shows that if we have a full duplex system for two nodes to share data they clearly should do so using network coding, specially for high packet erasures. Note that for $Pe = 0.8$ SDNC full duplex exceeds by more than 4 dB the performance of the scheduling scheme SDRR full duplex. It is important to point out that for low packet erasures the lower bound on the mean completion time of SDRR full duplex is loose while the upper bound is tight. In fact, the performance of SDRR full duplex is always equal or worse to that of the full duplex network coding scheme.

Finally, Figure 6 illustrates the number of iterations that the search algorithm proposed in this work before it reaches a stable solution. We observe that for different values $M$ and a wide range of $Pe$, the number of iterations is very low, always lower or equal to 5 iterations in the example. Thus, the algorithm converges very fast, especially if the $Pe$ is low. Thus, we have an algorithm that converges fast in a search that involves a very large amount of integer variables, e.g. with $M = 5$ and $M = 15$ initial packets in each node, we need to optimize 70 and 510 variables, respectively, for every value of $Pe$.

## V. Random Network Coding for Sharing Information in TDD channels: N nodes

The problem for $N$ nodes constitutes a natural and simple extension from the case of two nodes. Each node $i$ has $M_i$ data packets or disjoint random linear combinations that he wants to share with the other node. We assume that the nodes transmit following a round robin assignment, where the order of transmission has been predefined. Each node $i$ can transmit information at a given data rate $R_i$ [bps] to the other nodes. We assume a memoryless packet erasure channel $Pe_{i,j}$ for transmissions from node $i$ to node $j$, and that the channels are independent. We also assume that each transmission from a node can be received by each of the other nodes. Finally, we assume that the next transmitter node will wait for all nodes to receive the previous information. For later analysis, let us define $T_{prop-i}$ as the propagation time from node $i$ to the node that is farthest from it.

Again, the process is modelled as a Markov chain. The states $\big((s_{1,2},..,s_{1,N}),(s_{2,1},s_{2,3},..,s_{2,N}),..,(s_{N,1},..,s_{N,N-1}),t\big)$ are defined by the number of dofs required $s_{a,b}$ required by node $b$ to successfully decode all packets from node $a$, and the node $t$ that acts as transmitter in this state. Note that $(s_{a,1},...,s_{a,N})$ represents the degrees of freedom that other nodes require from node $a$ in order to decode his information. In order to simplify notation, let us define $S = \big((s_{1,2},..,s_{1,N}),(s_{2,1},s_{2,3},..,s_{2,N}),..,(s_{N,1},..,s_{N,N-1})\big)$, so that $(S,t)$ represents a state of the Markov chain, and $S_a = (s_{a,1},s_{a,2},..,s_{a,N})$ being the state of the receivers of node $a$. This is a Markov chain with $N(M_1+1)^{N-1}(M_2+1)^{N-1}..(M_N+1)^{N-1}-N$ transient states and $N$ recurrent states. Finally, note that a transition occurs every time that a batch of back-to-back coded packets is received at one receiver.

The number of variables to be optimized in order to provide an optimal solution increases exponentially with the number of nodes $N$, because we would have to consider a variable per state, i.e. $N_{((S_1,S_2,...,S_N),t)}, \forall S_i, t$. However, we can use a similar approach to [4] to reduce the number of variables, i.e. consider only the maximun degrees of freedom $s_a = \max_b s_{a,b}$ that the receivers of $a$ need in order to completely decode the information. This reduces the number of variables to optimize to $N(M_1+1)(M_2+1)...(M_N+1)-N$ and we will rename them $N_{((s_1,s_2,...,s_N),t)}$, which represent the number of coded packets to send. Given the characteristics of the Markov chain, the transition probabilities from state $(S,t)$ to state $(S',t')$ are

$$P_{(S,t)\to(S',t')} = \begin{cases} P\left(S'_a|S_a,N_{((s_1,...,s_N),t)}\right) & \text{if,} S_b = S'_b, \forall b \neq a, \\ & t = a,\ t' = t_{next}(a),\ \forall a \\ 0 & \text{otherwise} \end{cases}$$

where $t_{next}(a)$ represents the next node that should transmit after node $a$ has transmitted, and $P\left(S'_a|S_a,N_{((s_1,...,s_N),t)}\right)$ is the probability of transitioning from $S_a$ to $S'_a$ when node $a$ has transmitted $N_{((s_1,...,s_N),t)}$ coded packets. Assuming that a transmission from any node to the other $N-1$ nodes goes through independent channels, we have that

$$P\left(S'_a|S_a,N_{((s_1,...,s_N),t=a)}\right) = \tag{18}$$

$$\prod_{j\neq a} P\left(s'_{a,j}|s_{a,j},N_{((s_1,...,s_N),t=a)}\right) \tag{19}$$

where $P\left(s'_{a,j}|s_{a,j},N_{((s_1,...,s_N),t)}\right)$ has the same distribution studied in Section II, and represents the transition probability related to the knowledge of node $j$ with respect to the data node $a$ has, when node $a$ sends $N_{((s_1,...,s_N),t)}$ coded packets. For ease of notation, we will substitute $N_{((s_1,...,s_N),t)}$ for $Nt$. For $0 < s_{a,j'} < s_{a,j}$, this can be translated into

$$P\left(s'_{a,j}|s_{a,j},Nt\right) = \tag{20}$$

$$f(s_{a,j},s_{a,j'})(1-Pe_{a,j})^{s_{a,j}-s_{a,j'}}{Pe_{a,j}}^{Nt-s_{a,j}+s_{a,j'}} \tag{21}$$

For $s_{a,j} = s_{a,j'} > 0$ the expression for the transition probability reduces to $P\left(s_{a,j}|s_{a,j},Nt\right) = {Pe_{a,j}}^{Nt}$. Note that for $P\left(0|0,Nt\right) = 1$. Finally, for $s'_j = 0$ $P\left(s_{a,j'}=0|s_{a,j},Nt\right) = 1 - \sum_{s_{a,j'}=1}^{s_{a,j}} P\left(s_{a,j'}|s_{a,j},Nt\right)$.

We can define $P$ as the transition probability for our system.

### A. Expected Time for completing transmission

The expected time for completing the sharing process of all data packets between the nodes constitutes the expected time of absorption, i.e. the time to reach any state $((0,...0),...,(0,...0),t)$ for some $t$ for the first time, given that the initial state is $((M_1,...,M_1),...,(M_N,...,M_N),t)$ for some $t$, depending on which node starts transmitting. This can be expressed in terms of the expected time for completing the transmission given that the Markov chain is in state $(S,t)$, $T_{(S,t)}$ , $\forall S \forall t$. For our scheme, we consider that the transmission time of a packet from node $i$ is given by $T_{p-i} = \frac{h+n+gM_i}{R_i}$.

Let us define $T^{(S,t)}$ as the time it takes to transmit $N_{(s_1,...,s_N,t)}$ coded data packets and reach the node that is farthest away from $t$. It is easy to show that $T^{(S,t)} = N_{(s_1,...,s_N,t)}T_{p-t} + T_{prop-t}$.

The mean completion time when the system is in state $(S,t)$ is given by

$$T_{(S,t)} = T^{(S,t)} + \sum_{(S',t')} P_{(S,t)\to(S',t')}T_{(S',t')} \tag{22}$$

which can be expressed in vector form as $\bar{T} = [I-P]^{-1}\bar{\mu}$, where $\bar{T} = [T_{(S,t)}]$, $\bar{\mu} = [T^{(S,t)}]$, and $P$ is the corresponding transition probability. Since we are interested in the mean completion time when we start at states $((M_1,...,M1,...,(M_N,...,M_N),t)$ for some $t$, we can use Cramer's rule as in Section II.

### B. Minimizing The Mean Completion Time

Note that even after reducing the number of variables to optimize, the optimization becomes computationally prohibitive because the number of states in the Markov chain increases exponentially with the number of nodes. However, we can compute the variables of interest by using a slightly modified version of the algorithm for two nodes. This new algorithm uses heuristics to obtain good estimates of the variables while reducing computation. Since we have assumed that only one node transmits at each time and that this transmitter will broadcast the information to all other nodes, we can use similar heuristics to those presented in [4] for the case of broadcast. These heuristics rely on solving the link case [2] considering as packet erasure probability of the link a function of the packet erasure probabilities of the different channels in broadcast. This approximation will allow us to use a similar algorithm to that proposed for the case of two nodes, with only slight modifications.

The heuristic that showed best performance in [4] was the 'Worst Link Channel' heuristic. This heuristic approximates the system as a link to the receiver with the worst channel, i.e. the worst packet erasure probability. Then, we compute $N_{(s_1,...,s_N),t}, \forall s_i$ and $\forall t$ to minimize the mean completion time as in [2] with similar modifications to that in the algorithm for two nodes, namely using as round trip time that depends on both the physical round trip time and the transmissions of other nodes in the system.

Note that using a similar procedure to that of Section II-B, we can obtain that

$$\begin{aligned} T_{(S,t_1)} &= N_{(s_1,..,s_N,t_1)}T_{p-t_1} + \sum_{b=1}^{N}\left(T_{prop-t_b}\right) \\ &+ \sum_{n=2,..,N} T_{p-t_n}E_{t_2,..,t_n}\left[N_{(i_1,..,i_N,t_n)}|(S,t_1)\right] \\ &+ \sum_{S',S^{(2)},..,S^{(N)}} T_{(S',t_1)}\mathbf{P}_{(S,t_1)\to(S',t_1)} \end{aligned}$$

where

$$\mathbf{P}_{(S,t_1)\rightarrow(S',t_1)} = P_{(S,t_1)\rightarrow(S^{(2)},t_2)} P_{(S^{(2)},t_2)\rightarrow(S^{(3)},t_3)} \cdots P_{(S^{(N)},t_N)\rightarrow(S',t_1)},$$

and

$$E_{t_2,\ldots,t_n}\left[N_{(i_1,\ldots,i_N,t_n)}|(S,t_1)\right] = \quad (23)$$

$$\sum_{S^{(2)},\ldots,S^{(n)}} N_{(S^{(n)},t_n)} \mathbf{P}_{(S,t_1)\rightarrow(S^{(n)},t_n)} \quad (24)$$

where

$$\mathbf{P}_{(S,t_1)\rightarrow(S^{(n)},t_n)} = \quad (25)$$

$$P_{(S,t_1)\rightarrow(S^{(2)},t_2)} \cdots P_{(S^{(n-1)},t_{n-1})\rightarrow(S^{(n)},t_n)}. \quad (26)$$

Let us define $\hat{N}_{(s_1,s_2,\ldots,s_N,t_k)}(n)$ as the estimate for $N_{(s_1,s_2,\ldots,s_N,t_k)}$ at step $n$ of the algorithm. Then, our algorithm can be written as follows

*Algorithm 2:* Search Algorithm for N nodes

- STEP 0: INITIALIZE
  -Set $\hat{N}_{(s_1,s_2,\ldots,s_N,t_k)}(0) = s_k$, $\forall k$.
  -Set $n = 1$.
- STEP 1: TRANSMISSION FROM NODE 1:
  -Set $\hat{N}_{(s_1,s_2,\ldots,s_N,t_k)}(n) = \hat{N}_{(s_1,s_2,\ldots,s_N,t_k)}(n-1), \forall k$.
  FOR $s'_2 = 1, 2, \ldots, M_2$
  ...
  FOR $s'_N = 1, 2, \ldots, M_N$
  Compute $\hat{N}_{(s_1,s'_2,\ldots,s'_N,t_1)}(n), \forall s_1 = 1, \ldots, M_1$ to minimize the completion time of a TDD link with $Pe = \max_b Pe_{1,b}$, and with transition cost $\hat{N}_{(s_1,s'_2,\ldots,s'_N,t_1)}(n) T_{p-1} + \sum_{b=1}^{N} \left(T_{prop-t_b}\right) + \sum_{y=2,\ldots,N} T_{p-t_y} E_{t_2,\ldots,t_y}\left[\hat{N}_{(i_1,\ldots,i_N,t_y)}(n)|(S,t_1)\right]$ where $S = (s_1, s'_2, \ldots, s'_N)$.
  END FOR
  ...
  END FOR
- STEP k (=2,...,N): TRANSMISSION FROM NODE k TO NEXT NODE:
  FOR $s'_1 = 1, 2, \ldots, M_1$
  ...
  FOR $s'_N = 1, 2, \ldots, M_N$
  Compute $\hat{N}_{(s'_1,\ldots,s_k,\ldots,s'_N,t_k)}(n), \forall s_k = 1, \ldots, M_k$ to minimize the completion time of a TDD link with $Pe = \max_b Pe_{k,b}$, and with transition cost $\hat{N}_{(s'_1,\ldots,s'_N,t_k)}(n) T_{p-k} + \sum_{b=1}^{N} \left(T_{prop-t_b}\right) + \sum_{y=k+1,..,N,1,..k-1} T_{p-t_y} E_{t_k,..,t_y}\left[\hat{N}_{(i_1,..,t_y)}(n)|(S,t_k)\right]$ where $S = (s'_1, \ldots, s_k, \ldots, s'_N)$.
  END FOR
  ...
  END FOR
- STEP N+1: STOPPING CRITERIA
  IF $\hat{N}_{(s_1,\ldots,s_N,t)}(n) = \hat{N}_{(s_1,\ldots,s_N,t)}(n)(n-1)$, $\forall s_1, \ldots, s_N, t$
  Stop
  ELSE
  $n = n + 1$, and go to Step 1.
  END IF

## VI. Conclusion

This paper provides an extension to the use of random linear network coding over time division duplexing channels. We study the case of sharing disjoint information between two nodes. Each node $i$ has a block of $M_i$ data packets, and both nodes want to have all information at the end of the sharing process. Similar to our work in [2] the scheme considers that a number of coded data packets are transmitted back-to-back before stopping to wait for the other node to transmit its own coded data packet and acknowledge how many degrees of freedom, if any, are required to decode the information correctly.

We provide a simple algorithm to compute the number of coded packets to be sent before stopping given the state of the system. This algorithm is based on an iterative computation of the number of coded packets, and repeatedly using the minimization procedure of a link presented in [2] but with a slightly different cost associated to it. We present numerical examples showing that the number of iterations required for the algorithm to converge is very low for wide ranges of the packet erasure probability.

## Acknowledgment

This work was supported in part by the National Science Foundation under grants No. 0831728, 0831728 and CNS-0627021, by ONR MURI Grant No. N00014-07-1-0738, subcontract # 060786 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-06-C-2020 (CBMANET), subcontract # 18870740-37362-C issued by Stanford University and supported by the DARPA.

## References

[1] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, B. Leong,"A Random Linear Network Coding Approach to Multicast", Trans. Info. Theory, vol. 52, no. 10, pp.4413-4430, Oct. 2006

[2] D. E. Lucani, M. Stojanovic, M. Médard, "Random Linear Network Coding For Time Division Duplexing: When To Stop Talking And Start Listening", in Proc. INFOCOM'09, Rio de Janeiro, Brazil, pp. 1800-1808, Apr. 2009

[3] D. E. Lucani, M. Stojanovic, M. Médard, "Random Linear Network Coding For Time Division Duplexing: Energy Analysis", in Proc. ICC'09, Dresden, Germany, Jun. 2009

[4] D. E. Lucani, M. Médard, M. Stojanovic, "Broadcasting in Time-Division Duplexing: A Random Linear Network Coding Approach", in Proc. NetCod'09, Lausanne, Switzerland, pp. 62-67, Jun. 2009

[5] A. Eryilmaz, A. Ozdaglar, M. Médard, "On Delay Performance Gains from Network Coding", In Proc. CISS'06, pp. 864-870, Princeton, NJ, USA, Mar. 2006