Broadcasting with Port Constraints

Zhaohong Lu, Nicholas Chiapputo, Hua Sun
Department of Electrical Engineering, University of North Texas
Email: luzhaohong4773@gmail.com, nicholaschiapputo@my.unt.edu, hua.sun@unt.edu

***Abstract*—We consider the problem of broadcasting a common message (comprised of L symbols) from a source node to K destination nodes over a fully connected network, where each pair of nodes is connected by a noiseless link that can carry one symbol per time slot. The most important constraint is that each node has a single *port*, i.e., each node can only send to, or receive from, one other node at a given time. We are interested in the minimum number of time slots, T^*, required for this broadcast problem. We show that $T^* \geq \lceil \log_2(K+1) \rceil + L - 1$ through an information theoretic converse based on port constrained cut-set bounds. For achievability, we provide a routing based protocol that achieves the above T^* lower bound when $K \leq 100, L \leq 500$.**

I. Introduction

Broadcast is an important information theoretic primitive that has been used in a diverse array of applications, e.g., index coding [1], coded caching [2], [3], coded shuffling [4] and interprocessor connections [5].

In this work, we are interested in the problem of broadcasting a common message comprised of L symbols from a source node to K destination nodes. This problem is trivial if the source node is connected to the destination nodes through K orthogonal links (e.g., K separate wires) or a multicast link (e.g., a wireless broadcast channel), because it simply takes the source node L time slots to broadcast the L symbols (assuming each link can carry one symbol per time slot). The additional constraint we consider is that each node only has a single *port*, i.e.,

(Single port constraint): Each node can only send to, or receive from, one other node at a given time.

Such a port constraint has been considered in the literature and is motivated by the following applications.

1) (Millimeter Wave Communications) Wireless broadcast link becomes *directed* over high frequency band, i.e., the transmitter and the receiver need to steer their beams towards each other to establish the link between them. This is equivalent to the single port constraint and such a model has been studied in two hop Gaussian relay networks [6], [7].
2) (Processor Networks) On the circuit level, processors are connected by interconnection networks and due to physical constraints, a processor may not be able to simultaneously use all of its connections. In other words, at a given time, a processor needs to pick which node to communicate with and this constraint is termed the port constraint [8], [9] (and we adopt this name in our work. Note that the single port constraint is sometimes referred to as single link availability [10]).

Also, cheap nodes in the internet-of-things era might have port constraints because of the cost of establishing multiple communication sessions (where a port is similar to a socket of programming interfaces for various protocols).

With single port constraint, the broadcast problem is again trivial if we only allow communication between the source node and the destination nodes, because the source node needs to talk to the destination nodes one by one over KL time slots. To accelerate the broadcast procedure, we allow the destination nodes to communicate among themselves (where each node is subject to the single port constraint). In particular, we consider the best case of fully connected network topology, where each pair of nodes is connected by a noiseless link that can carry one symbol per time slot. Direct communication among the destination nodes is a natural assumption in wireless networks (e.g., device-to-device communication paradigm [3]) and processor interconnection networks (where the cooperation among the destination nodes is referred to as parallel computing and the network topology in the chip architecture is typically the Boolean n-cube instead of fully connected [8], [10]).

To recap, we consider the problem of broadcasting a common message of L symbols from a source node to K destination nodes over a fully connected network, where each node has a single port. We assume that each node is full-duplex (note that extensions to the half-duplex case are immediate), i.e., each node can simultaneously send and receive at the same time (although due to the single port constraint, he can only send to, and receive from, one other node over each time slot). The metric we use is the minimum number of time slots to complete this broadcast task in the information theoretic sense (note that prior work in processor networks only allows routing [8]–[10]). The motivation of this work is to characterize the information theoretic minimum number of time slots, T^*, for this single port broadcast problem.

For a concrete example, suppose we have $L = 3$ common symbols for $K = 5$ destination nodes. A single port constrained achievable scheme that uses $T = 5$ time slots is presented in Figure 1.

We will show later that this scheme is information theoretically optimal (for all schemes, i.e., not limited to routing).

The main results of this work are a universal lower bound on T^*, i.e., $T^* \geq \lceil \log_2(K+1) \rceil + L - 1$ for any K, L and a routing based achievable scheme that attains this lower bound for all cases where $K \leq 100, L \leq 500$.

We start by presenting the system model.

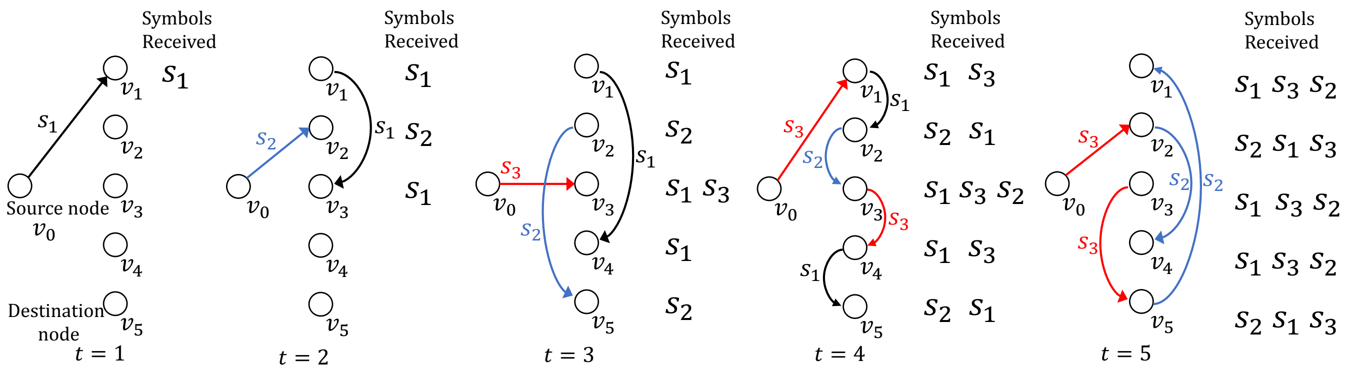

Fig. 1. Source node v_0 broadcasts $L=3$ symbols s_1, s_2, s_3 to $K=5$ destination nodes $v_1, \cdots, v_5$ over $T=5$ time slots by routing. Each full-duplex node can only send to (for a destination node, he must receive a symbol first before sending it out), and receive from one other node over each time slot.

II. System Model

Consider a fully connected network with one source node v_0 and K destination nodes $v_1, v_2, \cdots, v_K$. Each pair of nodes is connected by a noiseless link that can carry one symbol over a finite field $\mathbb{F}_p$ per time slot. Each node is full duplex, i.e., can simultaneously send and receive, and has a single port, i.e., can only send to, or receive from one other node at a given time. Over the t-th time slot, the input and output relationship is described as follows.

$$\begin{aligned} Y_j(t) = [&\theta_{j0}(t)X_0(t), \theta_{j1}(t)X_1(t), \cdots, \theta_{j(j-1)}(t)X_{j-1}(t),\\ &\theta_{j(j+1)}(t)X_{j+1}(t), \cdots, \theta_{jK}(t)X_K(t)], \quad (1)\\ &j \in \{0,1,\cdots,K\}, t \in \{1,2,\cdots,T\} \quad (2) \end{aligned}$$

where $X_i(t) \in \mathbb{F}_p, i \in \{0,1,\cdots,K\}$ is the transmitted signal from node v_i, $Y_j(t) \in \mathbb{F}_p$ is the received signal at node v_j, and $\theta_{ji}(t) \in \{0,1\}$ is a binary variable that indicates if the communication link from node v_i to node v_j is active, over the t-th time slot. From the single port constraint, we have $\forall t$

$$\forall i, |\{\theta_{ji}(t) : \theta_{ji}(t) = 1, \forall j \neq i\}| \leq 1 \quad (3)$$
$$\forall j, |\{\theta_{ji}(t) : \theta_{ji}(t) = 1, \forall i \neq j\}| \leq 1 \quad (4)$$

where (3) requires that for any node v_i and over any time slot t, at most one of $\theta_{0i}(t), \cdots, \theta_{(i-1)i}(t), \theta_{(i+1)i}(t), \cdots, \theta_{Ki}(t)$ is non-zero, i.e., v_i can only send to one other node. Similarly, (4) requires that over any time slot, node v_j can only receive from one other node.

The source node v_0 has a common message W for all destination nodes. The message W is comprised of L i.i.d. uniform symbols from $\mathbb{F}_p$, i.e., $W = (s_1, \cdots, s_L)$, where $s_l \in \mathbb{F}_p, \forall l \in \{1,2,\cdots,L\}$. At the beginning of communication, i.e., when $t=1$, the message W is available only at the source node. Over the t-th time slot, the transmitted signal from node v_i is a function of all information available at node v_i (i.e., received signals up to time $t-1$ and the message if the node is the source node v_0).

$$H(X_0(t)|W, Y_0(1), Y_0(2), \cdots, Y_0(t-1)) = 0 \quad (5)$$
$$H(X_i(t)|Y_i(1), Y_i(2), \cdots, Y_i(t-1)) = 0, i \neq 0 \quad (6)$$

Denote $\Theta \triangleq \{\theta_{ji}(t), \forall t, i, j, i \neq j\}$ as the collection of all θ variables across all nodes and over all time slots. For a given scheme, Θ is deterministic. We assume Θ is globally known at all nodes (i.e., we do not allow to use the port on-off information to communicate the message W). At the end of the T-th time slot, we require each destination node to be able to decode W with no error.

$$H(W|Y_j(1), Y_j(2), \cdots, Y_j(T)) = 0, j \neq 0 \quad (7)$$

In this work, we are interested in the finite block-length regime, where the message size L is assumed to be a constant (instead of the asymptotic regime where L is allowed to go to infinity). The metric we consider is the completion time T, whose minimum over all possible schemes is denoted as T^*. We wish to characterize T^*, as a function of K, L.

III. Main Result

Our main result is stated in Theorem 1.

Theorem 1: For the problem of broadcasting L symbols from a source node to K destination nodes over a fully connected network with single port full-duplex nodes, the minimum completion time is

$$T^* = \lceil \log_2(K+1) \rceil + L - 1 \quad (8)$$

for all cases where $K \leq 100$, $L \leq 500$.

The lower bound $T^* \geq \lceil \log_2(K+1) \rceil + L - 1)$ is valid for all values of K, L (i.e., with no restriction of $K \leq 100, L \leq 500$). This converse proof is presented in Section V. The proof is based on port constrained cut-set bounds.

The achievability proof is presented in Section IV, where we provide a routing scheme and show that this scheme achieves the completion time in Theorem 1 when $K \leq 100$, $L \leq 500$.

Remark: When $L \to \infty$, $T^*/L \to 1$, meaning that for large L, we need roughly L time slots to complete the broadcast task (i.e., as if there is only 1 destination node).

IV. Proof of Theorem 1: Achievability

In this section, we present a routing based achievable scheme, where the main difficulty is that even limited to routing, we have a very large number of possible choices and it is not clear which one will work. In particular, we need to determine the following.

1) Over each time slot, how many symbols shall we send and what are the symbol indices?
2) After 1) is determined, which set of nodes shall receive the symbols allocated?
3) After 1) and 2) are determined, for the nodes to receive the allocated symbols, which set of nodes shall send them (subject to the constraint that the symbols to send from the destination nodes must first be received)?

For each of the questions above, we could resort to brute force in principle. However, it is not hard to show that for each of them, the brute force search complexity will be exponential. We will instead pursue a computationally efficient approach (polynomial in K, L) that achieves the lower bound for all cases where $K \leq 100, L \leq 500$ (thus optimal).

To better illustrate the key ideas, let us start with the $K = 11, L = 4$ setting before considering the general case.

A. Illustrative Example: $K = 11, L = 4$

We present a scheme that uses $T = \lceil \log_2(K+1) \rceil + L - 1 = 7$ time slots to send the $L = 4$ symbols s_1, s_2, s_3, s_4 to the $K = 11$ destination nodes $v_1, \cdots, v_{11}$.

Corresponding to the 3 questions mentioned above, we divide the design of the scheme into 3 steps, resolving the 3 questions respectively. The steps are presented below.

1) Step 1 - Symbol Allocation: The information about which symbols to send over each time slot (corresponding to the first question above) is captured by the following symbol allocation matrix, denoted by $\mathbf{A}$, which has $L = 4$ rows (each row corresponds to one symbol) and $T = 7$ columns (each column corresponds to one time slot).

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 4 & 8 & 11 & 11 & 11 \\ 0 & 1 & 2 & 4 & 8 & 11 & 11 \\ 0 & 0 & 1 & 2 & 4 & 8 & 11 \\ 0 & 0 & 0 & 1 & 3 & 7 & 11 \end{bmatrix}_{L \times T} \tag{9}$$

where $\mathbf{A}(i, j)$ (the element located at the i-th row and j-th column) represents the total number of destination nodes that have received the symbol s_j at the end of time $t = j$. For example, $\mathbf{A}(2, 5) = 8$, meaning that at the end of time $t = 5$, 8 destination nodes (which 8 will be determined in the second step) have received the symbol s_2.

In order to find which symbols are sent over each time slot, we need the matrix $\overline{\mathbf{A}}$, whose columns are given by the differences between every two consecutive columns of $\mathbf{A}$.

$$\overline{\mathbf{A}} = \mathbf{A} - [\mathbf{0}_{L \times 1}, \mathbf{A}(:, 1 : T - 1)] \tag{10}$$

$$= \begin{bmatrix} 1 & 1 & 2 & 4 & 3 & 0 & 0 \\ 0 & 1 & 1 & 2 & 4 & 3 & 0 \\ 0 & 0 & 1 & 1 & 2 & 4 & 3 \\ 0 & 0 & 0 & 1 & 2 & 4 & 4 \end{bmatrix}_{L \times T} \tag{11}$$

where $\overline{\mathbf{A}}(i, j)$ represents the number of destination nodes that will receive the symbol s_i over time $t = j$. For example, $\overline{\mathbf{A}}(2, 5) = \mathbf{A}(2, 5) - \mathbf{A}(2, 4) = 4$, meaning that over time $t = 5$, 4 destination nodes (again, which 4 are not determined yet) will receive the symbol s_2.

The rationale behind this symbol allocation matrix design is as follows. At the end of time $t = T$, all symbols must be received so that every element in the T-th column of $\mathbf{A}$ must be $K = 11$. The other columns are inspired by the lower bound (in particular, Lemma 1), which says that over time slot t, at most $\min(2^{t-1}, K)$ symbols can be sent. In other words, the total number of symbols sent may first increase exponentially for small t and then saturate when it reaches K. As a result, consider the first 4 columns of $\overline{\mathbf{A}}$, where we send $1, 2, 4, 8$ symbols in total over the first 4 time slots, respectively. Over the last 3 time slots, we fully exploit the potential of the communication among the destination nodes and send 8 symbols per time slot. Further, to reach a balance of breadth and depth, we impose a delayed exponential increase for each symbol. For example, for s_1, we send it to $1, 1, 2, 4, 3$ destination nodes over the first 5 time slots. The first 1 symbol is sent from the source node while the other symbols are sent from the destination nodes (the last element of 3 follows from the fact that at that time, only 3 destination nodes are missing s_1). s_2 and s_3 follow from the same pattern as s_1, while s_4 is different because for s_4, the source node could also help (while for s_1, s_2, s_3, the source node needs to send *new* symbols. Note that this is necessary because if s_4 is sent out too late, i.e., later than $t = 5$, then it can never be finished by time $t = T$). The general design of $\mathbf{A}$ (and $\overline{\mathbf{A}}$) is presented in Section IV-B (see (14)).

We emphasize that we do not know *a priori* the existence of an achievable scheme that is consistent with the symbol allocation matrix $\mathbf{A}$ ($\mathbf{A}$ is simply a natural choice that turns out to work and we fix $\mathbf{A}$ first mainly to reduce the computational complexity). So we further need to design the receivers and transmitters for the allocated symbols. This is presented next.

2) Step 2 - Receiver Allocation: After the symbols are allocated, we associate them with the destination nodes, i.e., we determine which nodes will receive the allocated symbols (corresponding to the second question). Such associations are described in the following receiver allocation matrix, denoted by $\mathbf{Rx}$, with dimension $K \times T$.

$$\mathbf{Rx} = \begin{bmatrix} s_1 & * & * & s_2 & s_3 & s_4 & * \\ * & s_1 & * & s_2 & s_3 & s_4 & * \\ * & s_2 & * & * & s_1 & s_3 & s_4 \\ * & * & s_1 & s_3 & s_2 & s_4 & * \\ * & * & s_1 & s_4 & s_2 & s_3 & * \\ * & * & s_2 & * & s_1 & s_3 & s_4 \\ * & * & s_3 & * & s_1 & s_2 & s_4 \\ * & * & * & s_1 & s_2 & s_3 & s_4 \\ * & * & * & s_1 & s_2 & s_4 & s_3 \\ * & * & * & s_1 & s_4 & s_2 & s_3 \\ * & * & * & s_1 & s_4 & s_2 & s_3 \end{bmatrix}_{K \times T} \tag{12}$$

where $\mathbf{Rx}(i, j) = s_l$ indicates that over time $t = j$, s_l is sent to destination node v_i, and $\mathbf{Rx}(i, j) = *$ indicates that nothing is sent to v_i over time $t = j$. For example, $\mathbf{Rx}(2, 5) = s_3$, meaning that over time $t = 5$, s_3 is sent to v_2.

Note that the receiver allocation matrix $\mathbf{Rx}$ is consistent with $\overline{\mathbf{A}}$. For example, over time $t = 5$, $\overline{\mathbf{A}}$ requires that 3 destination nodes will receive s_1 (from $\mathbf{Rx}$, they are v_3, v_6, v_7), 4 destination nodes will receive s_2 (i.e., v_4, v_5, v_8, v_9), 2 destination nodes will receive s_3 (i.e., v_1, v_2), and 2 destination nodes will receive s_4 (i.e., v_{10}, v_{11}).

The receiver allocation matrix is designed as follows. We have 3 time segments. The first time segment is comprised of time slots $1 \leq t \leq 3$. This segment is simple, where the goal is to reach as many destination nodes as possible. Following $\overline{\mathbf{A}}$ which indicates $1, 2, 4$ symbols are sent over the first 3 time slots respectively, we choose v_1 to v_7 and over each time slot, the symbols are allocated lexicographically.

The second time segment is comprised of a single time slot $t = \lceil \log_2(K+1) \rceil = 4$. This segment is the most sophisticated. We consider symbols sequentially from s_1 to s_4. Note that we have 8 symbols in total (4 of s_1, 2 of s_2, 1 of s_3 and 1 of s_4, refer to the 4-th column of $\overline{\mathbf{A}}$). First, we continue to send to destination nodes that have not received any symbol (i.e., v_8, v_9, v_{10}, v_{11}). In this case, coincidently we have 4 s_1 to send, so v_8, v_9, v_{10}, v_{11} are the destination nodes for s_1 (if we do not have enough s_1 symbols, we consider s_2, s_3 etc. in order). After ensuing that each destination node has received at least one symbol, we next consider the remaining symbols. If we have s_1 symbols left, then the remaining s_1 symbols will be sent to the first nodes that have not received s_1. If we are left with symbols other than s_1, we send them to the first available nodes that have received s_1. In this case, we have 2 of s_2, 1 of s_3 and 1 of s_4 and there are 4 destination nodes that have received s_1 (v_1, v_2, v_4, v_5, refer to $\mathbf{Rx}$), which will be receiving these 4 s_2 to s_4 symbols over this time slot. This design (sending s_2, s_3, s_4 to nodes that have received s_1) is to facilitate later transmissions and is crucial to avoid conflicts.

The third time segment is comprised of the remaining time slots $5 \leq t \leq 7$. We use a greedy procedure for this segment, where the symbols are considered sequentially from s_1 to s_4, and for symbol s_l, if α_l destination nodes wish to receive s_l, we pick the first α_l available destination nodes that have not received s_l yet and have not been scheduled to receive any symbol in this time slot. For example, consider $t = 5$ (i.e., the 5-th column of $\mathbf{Rx}$). We start from s_1, which needs to be sent to 3 destination nodes. To find these 3 destination nodes, we observe that from the first 4 time slots, destination nodes v_1 and v_2 have already received s_1 while v_3 has not, so the first destination node to receive s_1 is v_3. Next, v_4, v_5 have received s_1, while v_6, v_7 have not, so the remaining two destination nodes to receive s_1 are v_6, v_7. Then we proceed to s_2, which needs to be sent to 4 destination nodes. We pick them to be v_4, v_5, v_8, v_9 because they are the first 4 *available* nodes that *miss* s_2 (note that v_1, v_2, v_3 have received s_2 and v_6, v_7 have been scheduled to receive s_1 so that they are not available due to the single port constraint). The destination nodes for s_3, s_4 are designed similarly.

This receiver allocation matrix $\mathbf{Rx}$ is highly non-trivial because of the second and third time segments. For the second segment, it is important to first reach all destination nodes and ensure remaining symbols other than s_1 are sent to nodes that have already received s_1. Surprisingly, such an operation is compatible with the greedy third segment and results complicated dependence over time. Again, we do not know *a priori* such $\mathbf{Rx}$ exists (verified through programming), and such $\mathbf{Rx}$ will work, which requires that the allocated symbols can be sent out by some schedule of the source and destination nodes with single port. The transmitter allocation design, the final step, is presented below.

3) Step 3 - Transmitter Allocation: To complete the description of the routing scheme, the transmitters are scheduled according to the following transmitter allocation matrix, denoted by $\mathbf{Tx}$, with dimension $K \times T$. This answers the third question. To make it easy to check that the memory constraint is satisfied (i.e., a symbol must be received by a destination node first before it is sent out. Refer to (6)), we put the $\mathbf{Rx}$ matrix and the $\mathbf{Tx}$ matrix together.

$$\mathbf{Rx}(\mathbf{Tx}) =$$
$$\begin{bmatrix}
s_1(v_0) & * & * & s_2(v_3) & s_3(v_4) & s_4(v_0) & * \\
* & s_1(v_1) & * & s_2(v_6) & s_3(v_7) & s_4(v_5) & * \\
* & s_2(v_0) & * & * & s_1(v_8) & s_3(v_1) & s_4(v_0) \\
* & * & s_1(v_1) & s_3(v_7) & s_2(v_1) & s_4(v_{10}) & * \\
* & * & s_1(v_2) & s_4(v_0) & s_2(v_2) & s_3(v_2) & * \\
* & * & s_2(v_3) & * & s_1(v_9) & s_3(v_4) & s_4(v_1) \\
* & * & s_3(v_0) & * & s_1(v_{10}) & s_2(v_3) & s_4(v_2) \\
* & * & * & s_1(v_1) & s_2(v_3) & s_3(v_7) & s_4(v_4) \\
* & * & * & s_1(v_2) & s_2(v_6) & s_4(v_{11}) & s_3(v_3) \\
* & * & * & s_1(v_4) & s_4(v_0) & s_2(v_6) & s_3(v_5) \\
* & * & * & s_1(v_5) & s_4(v_5) & s_2(v_8) & s_3(v_6)
\end{bmatrix} \tag{13}$$

where $\mathbf{Tx}(i, j) = v_k$ indicates that over time $t = j$, the symbol required (given by $\mathbf{Rx}(i, j)$) by destination node v_i is sent from node v_k. For example, $\mathbf{Tx}(2, 5) = v_7$, meaning that over time $t = 5$, symbol $\mathbf{Rx}(2, 5) = s_3$ is sent from v_7 to v_2.

The transmitter allocation matrix is designed as follows, where each time slot follows from the same greedy procedure. Importantly, we consider the symbols in a reverse[1] order, i.e., for each time slot, we consider the symbols from s_4 to s_1 sequentially, and if we wish to send symbol s_l to α_l destination nodes, the transmitters are chosen as the first α_l available nodes that have s_l and have not been scheduled to send other symbols (i.e., consider from the source node v_0 to the destination nodes $v_1, \cdots, v_K$ in order). For example, consider $t = 5$ (i.e., the 5-th column of the above matrix). We start from s_4, which is required by v_{10}, v_{11} (from $\mathbf{Rx}$). For v_{10}, the transmitter is chosen as the source node v_0 (the first node that has s_4) and for v_{11}, the transmitter is chosen as v_5 (the next node that has s_4. Note that v_5 has received s_4 in the previous time slot, refer to $\mathbf{Rx}(5, 4)$). Then we proceed to s_3, which is required by v_1, v_2. The transmitters are chosen as v_4, v_7 because they are the first nodes that have received s_3 (where v_4 received s_3 over time $t = 4$ and v_7 received s_3 over time $t = 3$). The transmitters for s_2 and s_1 are designed similarly.

[1]The rationale behind this reverse order is as follows. The last symbol is the most challenging (as a least number of nodes have received the last symbol), so we consider the symbols in a backwards manner.

Note that this transmitter allocation matrix $\mathbf{Tx}$ is a deterministic function of the receiver allocation matrix $\mathbf{Rx}$. From the design, we again do not know whether there exists a feasible allocation of $\mathbf{Tx}$ that is consistent with $\mathbf{Rx}$. For this step, we resort to programming verification. The outcome for the $K = 11, L = 4$ setting is shown above and in this case, $\mathbf{Tx}$ exists. In the next section, we will present the general algorithm for arbitrary K, L and we have verified that up to $K = 100, L = 500$, there exist feasible choices of $\mathbf{A}, \mathbf{Rx}, \mathbf{Tx}$ (these matrices fully specify a routing scheme).

B. The General Algorithm for Arbitrary K, L

In this section, we present a scheme that intends[2] to use $T = \lceil \log_2(K+1) \rceil + L - 1$ time slots to send the L symbols $s_1, \cdots, s_L$ to the K destination nodes $v_1, \cdots, v_K$. Following the example presented in the previous section, we invoke 3 steps for the scheme.

First, for any K, L, the symbol allocation matrix $\mathbf{A}$ (and $\overline{\mathbf{A}}$) with dimension $L \times T$ is set as follows.

$$\mathbf{A}(i,j) = \begin{cases} \min(2^{j-i}, K) & \text{if } i \le j, i \ne L \\ \min(2^{j-i+1} - 1, K) & \text{else if } i \le j, i = L \\ 0 & i > j \end{cases}$$
$$\overline{\mathbf{A}} = \mathbf{A} - [\mathbf{0}_{L\times 1}, \mathbf{A}(:, 1 : T-1)] \tag{14}$$

Second, the receiver allocation matrix $\mathbf{Rx}$ with dimension $K \times T$ (if exists) is generated from Algorithm 2.

Third, the transmitter allocation matrix $\mathbf{Tx}$ with dimension $K \times T$ (if exists) is generated from Algorithm 1.

Algorithm 1 Tx-Gen Algorithm.

```
1: Input: $K, L, T, \mathbf{Rx}$
2: Output: $\mathbf{Tx}$ or Failure
3: Initialize: Set each element of $\mathbf{Tx}$ as '$*$'.
4: for $t = 1 : T$ do {For each time slot...}
5:     Availability($v_k$) ← 'Y', $k \in \{0, 1, \cdots, K\}$ {each node is available to send.}
6:     for $l = L : 1$ do {For each symbol in reverse order...}
7:         for each $v_j$ s.t. $\mathbf{Rx}[j, t] = s_l$ do {for each destination node that requires this symbol.}
8:             $\mathcal{I} \leftarrow \{i : i = 0 \cup s_l \in \mathbf{Rx}(i, 1 : t-1)$, Availability($v_i$) = 'Y'} {find all available nodes that have the symbol.}
9:             if $\mathcal{I} = \emptyset$ then {If no qualified transmitter...}
10:                Exist and Return Failure
11:            end if {then $\mathbf{Tx}$ does not exist.}
12:            $i^* \leftarrow \mathrm{argmin}_i\ i \in \mathcal{I}$
13:            $\mathbf{Tx}(j, t) \leftarrow v_{i^*}$ {Otherwise pick the first available node to transmit.}
14:            Availability($v_{i^*}$) ← 'N' {The chosen transmitter can not send any more over this time slot.}
15:        end for ($v_j$)
16:    end for ($l$)
17: end for ($t$)
```

Algorithm 2 Rx-Gen Algorithm.

```
1: Input: $K, L, T, \overline{\mathbf{A}}$
2: Output: $\mathbf{Rx}$ or Failure
3: Initialize: Set each element of $\mathbf{Rx}$ as '$*$'.
4: Current = 1 {Start from the first destination node $v_1$.}
5: for $t = 1 : \lceil \log_2(K+1) \rceil - 1$ do {For the first segment...}
6:     for $l = 1 : L$ s.t. $\overline{\mathbf{A}}(l, t) \ne 0$ do {fill each symbol.}
7:         $\mathbf{Rx}$(Current : Current + $\overline{\mathbf{A}}(l, t) - 1, t) \leftarrow s_l$
8:         Current ← Current + $\overline{\mathbf{A}}(l, t)$
9:     end for ($l$)
10: end for ($t$)
11: Availability($v_k$) ← 'Y', $k \in \{1, \cdots, K\}$ {Now consider the second segment. Each node is available to receive.}
12: for each $l = 1 : L$ s.t. $\overline{\mathbf{A}}(l, t) \ne 0$ do {For each symbol...}
13:     for $\bar{l} = 1 : \overline{\mathbf{A}}(l, t)$ do {we have several instances.}
14:         $\mathcal{J}_1 \leftarrow$ [Current : $K$] {Find all untouched nodes.}
15:         $\mathcal{J}_2 \leftarrow \{j : s_1 \notin \mathbf{Rx}(j, 1 : t-1)$, Availability($v_j$) = 'Y'} {Find all available nodes that do not have $s_1$.}
16:         $\mathcal{J}_3 \leftarrow \{j : s_1 \in \mathbf{Rx}(j, 1 : t-1)$, Availability($v_j$) = 'Y'} {Find all available nodes that have $s_1$.}
17:         if $\mathcal{J}_1 \ne \emptyset$ then {Fill untouched nodes first.}
18:             $\mathbf{Rx}$(Current, $\lceil \log_2(K+1) \rceil$) ← $s_l$
19:             Current ← Current + 1
20:         else if $\mathcal{J}_1 = \emptyset$ and $l = 1$ then {If we have $s_1$ left...}
21:             $j^* \leftarrow \mathrm{argmin}_j\ j \in \mathcal{J}_2$ {pick the first to receive.}
22:             $\mathbf{Rx}(j^*, \lceil \log_2(K+1) \rceil) \leftarrow s_l$
23:             Availability($v_{j^*}$) ← 'N' {The chosen receiver can not receive any more over this time slot.}
24:         else if $\mathcal{J}_1 = \emptyset$ and $l > 1$ then {If have $s_2, \cdots$ left...}
25:             $j^* \leftarrow \mathrm{argmin}_j\ j \in \mathcal{J}_3$ {pick the first node that already has $s_1$ to receive.}
26:             $\mathbf{Rx}(j^*, \lceil \log_2(K+1) \rceil) \leftarrow s_l$
27:             Availability($v_{j^*}$) ← 'N'
28:         end if
29:     end for ($\bar{l}$)
30: end for ($l$)
31: for $t = \lceil \log_2(K+1) \rceil - 1 : T$ do {For the third segment...}
32:     Availability($v_k$) ← 'Y', $k \in \{1, \cdots, K\}$ {for each new time slot, each node is available to receive...}
33:     for $l = 1 : L$ s.t. $\overline{\mathbf{A}}(l, t) \ne 0$ do {for each symbol...}
34:         for $\bar{l} = 1 : \overline{\mathbf{A}}(l, t)$ do
35:             $\mathcal{J} \leftarrow \{j : s_l \notin \mathbf{Rx}(j, 1 : t-1)$, Availability($v_j$) = 'Y'} {find all available nodes that do not have $s_l$.}
36:             if $\mathcal{J} = \emptyset$ then {If no qualified receiver...}
37:                 Exist and Return Failure
38:             end if {then $\mathbf{Rx}$ does not exist.}
39:             $j^* \leftarrow \mathrm{argmin}_j\ j \in \mathcal{J}$
40:             $\mathbf{Rx}(j^*, t) \leftarrow s_l$ {Otherwise pick the first available node to receive and this node can not receive any more over this time slot.}
41:             Availability($v_{j^*}$) ← 'N'
42:         end for ($\bar{l}$)
43:     end for ($l$)
44: end for ($t$)
```

[2]The algorithm may return failure if no feasible solutions are found.

C. Programming Results

We have implemented the above algorithm in Python and have verified that for cases where $K \leq 100, L \leq 500$, we have feasible outputs[3]. The source code (with complete outputs) is available on Github [11].

V. Proof of Theorem 1 : Converse

In this section, we show that $T \geq \lceil \log_2(K+1) \rceil + L - 1$ for arbitrary K, L. Let us start with a lemma.

Lemma 1: For all $t \in \{1, 2, \cdots, T\}$, we have

$$\sum_{j=1}^{K} I(W; Y_j(t) | Y_j(1), \cdots, Y_j(t-1)) \leq \min(2^{t-1}, K) \quad (15)$$

Proof: We first show that the LHS of (15) is no larger than K. This follows simply from the fact that there are K terms in the summation and each term is no larger than $H(Y_j(t))$, where $H(Y_j(t)) \leq 1$ in p-ary units.

Next, we show by induction that the LHS of (15) is no larger than 2^{t-1}.

The basis case is when $t = 1$, and here we need to prove that $\sum_{j=1}^{K} I(W; Y_j(1)) \leq 1$. When $t = 1$, from (5), the message W is available only at the source node such that only $X_0(1)$ can be non-zero. Combining with the single port constraint (3), (4), we know that at most 1 destination node can hear $X_0(t)$. So $\sum_{j=1}^{K} I(W; Y_j(1)) \leq 1$.

Now suppose (15) holds for $t-1, t \geq 2$, meaning that at time $t-1$ there are (at most) 2^{t-2} destination nodes that receive information about W. From the single port constraint (3), (4), these 2^{t-2} destination nodes are receiving from 2^{t-2} nodes, resulting in at most 2^{t-1} nodes in total that have at least one symbol information about W. Therefore at time t, at most 2^{t-1} terms in the LHS of (15) are non-zero and the bound (15) follows.

Since both the basis and the inductive step have been performed, by mathematical induction, we have proved that (15) holds for all t. ∎

From (7), we have that for $j \in \{1, 2, \cdots, K\}$

$$L = H(W) = I(W; Y_j(1), Y_j(2), \cdots, Y_j(T)) \quad (16)$$

Adding (16) for all $j \in \{1, 2, \cdots, K\}$, we have

$$KL = \sum_{j=1}^{K} I(W; Y_j(1), Y_j(2), \cdots, Y_j(T)) \quad (17)$$

$$= \sum_{t=1}^{T} \sum_{j=1}^{K} I(W; Y_j(t) | Y_j(1), \cdots, Y_j(t-1)) \quad (18)$$

$$\overset{(15)}{\leq} 2^0 + 2^1 + \cdots + 2^{\lfloor \log_2 K \rfloor} + K(T - \lfloor \log_2 K \rfloor - 1) \quad (19)$$

$$= 2^{\lfloor \log_2 K \rfloor + 1} - 1 + K(T - \lfloor \log_2 K \rfloor - 1) \quad (20)$$

[3]In fact, we have run the code for many other larger cases and have not yet found one instance where it fails. An immediate open problem is to study if the algorithm always works for any K, L.

where to obtain (19), when $t \leq \lfloor \log_2 K \rfloor + 1$ we use 2^{t-1} as the RHS of (15) and when $t > \lfloor \log_2 K \rfloor + 1$ we use K as the RHS of (15).

From (20), we have

$$T \geq L - 1 + \lceil 2 + \lfloor \log_2 K \rfloor - \frac{2^{\lfloor \log_2 K \rfloor + 1} - 1}{K} \rceil \quad (21)$$

where the ceiling operation follows from the fact that T must be an integer. To complete the proof, we are left to show that $\lceil 2 + \lfloor \log_2 K \rfloor - \frac{2^{\lfloor \log_2 K \rfloor + 1} - 1}{K} \rceil = \lceil \log_2(K+1) \rceil$. This proof is presented in Lemma 2.

Lemma 2: For any integer K, we have

$$\lceil 2 + \lfloor \log_2 K \rfloor - \frac{2^{\lfloor \log_2 K \rfloor + 1} - 1}{K} \rceil = \lceil \log_2(K+1) \rceil \quad (22)$$

Proof: For any positive integer K, we can find an integer q such that $2^q \leq K \leq 2^{q+1} - 1$ (i.e., $q = \lfloor \log_2 K \rfloor$). Then

$$2^{q+1} - 1 \geq K \text{ and } 2^{q+1} - 1 < 2^{q+1} \leq 2K \quad (23)$$

$$\Rightarrow 1 \leq \frac{2^{q+1} - 1}{K} < 2 \Rightarrow \lceil -\frac{2^{q+1} - 1}{K} \rceil = -1 \quad (24)$$

Next, note that $\lfloor \log_2 K \rfloor$ is q and

$$\text{LHS of (22)} = \lceil 2 + q - \frac{2^{q+1} - 1}{K} \rceil \quad (25)$$

$$\overset{(24)}{=} q + 1 = \text{RHS of (22)} \quad (26)$$

Therefore, Lemma 2 holds. ∎

The converse proof is now complete.

References

[1] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," in *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06.*, 2006, pp. 197 – 206.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, 2016.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[5] M. Barnett, L. Shuler, R. van de Geijn, S. Gupta, D. Payne, and J. Watts, "Interprocessor collective communication library (intercom)," in *Proceedings of IEEE Scalable High Performance Computing Conference*. IEEE, 1994, pp. 357–364.

[6] Y. H. Ezzeldin, M. Cardone, C. Fragouli, and G. Caire, "Gaussian 1-2-1 networks: Capacity results for mmwave communications," *arXiv preprint arXiv:1801.02553*, 2018.

[7] G. K. Agarwal, Y. H. Ezzeldin, M. Cardone, and C. Fragouli, "Secure communication over 1-2-1 networks," *arXiv preprint arXiv:1801.03061*, 2018.

[8] S. L. Johnsson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Transactions on computers*, vol. 38, no. 9, pp. 1249–1268, 1989.

[9] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on parallel and distributed systems*, vol. 8, no. 11, pp. 1143–1156, 1997.

[10] D. P. Bertsekas, C. Özveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *Journal of Parallel and Distributed Computing*, vol. 11, no. 4, pp. 263–275, 1991.

[11] [Online]. Available: https://github.com/NickChiapputo/SinglePort