Dynamic Index Coding for Wireless Broadcast Networks

Michael J. Neely, Arash Saber Tehrani, and Zhen Zhang

***Abstract*—We consider a wireless broadcast station that transmits packets to multiple users. The packet requests for each user may overlap, and some users may already have certain packets. This presents a problem of broadcasting in the presence of side information, and is a generalization of the well-known (and unsolved) index coding problem of information theory. We represent the problem by a bipartite demand graph. Uncoded transmission is optimal if and only if this graph is acyclic. Next, we define a code-constrained capacity region that restricts attention to any prespecified set of coding actions. A dynamic max-weight algorithm that acts over variable length frames is developed. The algorithm allows for random packet arrivals and supports any traffic inside the code-constrained capacity region. A simple set of codes that exploit cycles in the demand graph are shown to be optimal for a class of broadcast relay problems.**

***Index Terms*—Network coding, optimization, queuing analysis.**

I. Introduction

CONSIDER a wireless broadcast station that transmits packets to N wireless users. Packets randomly arrive to the broadcast station. Each packet p is desired by one or more users in the set $\{1, \ldots, N\}$. Further, there may be one or more users that already have the packet stored in their cache. The broadcast station must efficiently transmit all packets to their desired users. We assume time is slotted with unit slots $t \in \{0, 1, 2, \ldots\}$, and that a single packet can be transmitted by the broadcast station on every slot. This packet is received error-free at all users. We assume that only the broadcast station can transmit, so that users cannot transmit to each other.

If the broadcast station has P packets at time 0, and no more packets arrive, then the mission can easily be completed in P slots by transmitting the packets one at a time. However, this approach ignores the side-information available at each user. Indeed, it is often possible to complete the mission in fewer than P slots if packets are allowed to be *mixed* before transmission. A simple and well-known example for two users is the following: suppose user 1 has packet B but wants packet A, while user 2 has packet A but wants packet B. Sending each packet individually would take 2 slots, but these demands can be met in just one slot by transmitting the mixed packet $A + B$, the bitwise XOR of A and B. Such examples are introduced in [2]–[4] in the context of wireless network coding.

The general problem, where each packet is contained as side information in an arbitrary subset of the N users, is much more complex. This problem is introduced by Birk and Kol in [5] and [6], and is known as the index coding problem. Methods for completing a general index coding mission in minimum time are unknown. However, the recent work [7] shows that if one restricts to a class of linear codes, then the minimum time is equal to the rank of the minimum rank matrix that solves a certain matrix completion problem. Unfortunately, the matrix completion problem is NP-hard in general. NP-hardness results for index coding over binary fields are shown in [8]. Work in [9] shows that linear index coding is equivalent to certain difficult problems in matroid theory as well as to linear network coding for multihop networks. However, such network coding problems are known to be difficult even to approximate [10].

Overall, optimal (linear or nonlinear) index coding seems to be intractable. Nevertheless, it is important to develop systematic approaches to these problems. That is because current wireless cellular systems cannot handle the huge traffic demands that are expected in the near future. This is largely due to the consistent growth of wireless video traffic. Fortunately, much of the traffic is for *popular content*. That is, users often download the same information. Thus, it is quite likely that a system of N users will have many instances of side information, where some users already have packets that others want. This naturally creates an index coding situation. Thus, index coding is both rich in its mathematical complexity and useful for supporting future wireless traffic.

The problem considered in this paper is even more complex because packets can arrive randomly over time. This is a practical scenario and creates the need for a *dynamic* approach to index coding. We assume there are M *traffic types*, where a type is defined by the subset of users that *desire* the packets and the subset that *already has* the packets. Let λ_m be the arrival rate, in packets/slot, for type m traffic. We approach this problem by restricting coding actions to an abstract set $\mathcal{A}$. We then show how to achieve the code constrained capacity region $\Lambda_{\mathcal{A}}$, being the set of all rate vectors $(\lambda_1, \ldots, \lambda_M)$ that can be supported using coding actions in the set $\mathcal{A}$. The set $\Lambda_{\mathcal{A}}$ is typically a strict subset of the *capacity region* Λ, which does not restrict the type of coding action. Our work can be applied to any set $\mathcal{A}$. Hence, it can be used with any desired codes, including *scalar linear* [7], *vector linear* [11], *nonlinear* [12], and heuristics based on set clusterings and graph colorings (see, for example, [8], [13]).

Manuscript received December 06, 2011; revised August 24, 2012; accepted August 01, 2013. Date of publication August 27, 2013; date of current version October 16, 2013. This work was supported in part by the NSF Career under Grant CCF-0747525, in part by NSF Grant 0964479, and in part by the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053. This paper was presented in part at the IEEE INFOCOM conference, Orlando, FL, USA, 2012.

The authors are with the Electrical Engineering Department, University of Southern California, Los Angeles, CA 90089 USA (e-mail: mjneely@usc.edu; arash.saber.tehrani@gmail.com; zhzhang@usc.edu).

Communicated by F. Baccelli, Associate Editor for Communication Networks.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2013.2279876

However, we focus attention on a simple class of codes that only use bitwise XOR operations and that exploit cycles in a demand graph. In special cases of broadcast relay problems, we show these codes can achieve the full capacity region Λ. We also consider a class of Reed–Solomon erasure codes [14] for extended problems.

Prior work in [15] develops an achievable rate region for index coding under a restricted class of linear *pollution-free* coding schemes. The solution is a linear program with known traffic rates, and does not consider a dynamic setting where data arrives randomly with unknown rates. Our framework can be applied to dynamic versions of the problem in [15] by defining the set $\mathcal{A}$ to be those coding actions that satisfy the pollution-free requirement. However, one can achieve a larger region simply by adding more coding actions to $\mathcal{A}$. For example, the simple three-cycle XOR coding action, described in Section II-C, does not meet the pollution-free requirement but is important for achieving capacity in a large class of broadcast relay networks.

The capacity region Λ is directly related to the conceptually simpler *static* problem of clearing a fixed batch of packets in minimum time. Further, index coding concepts are most easily developed in terms of the static problem. Thus, this paper is divided into two parts: we first introduce the index coding problem in the static case, and we describe example coding actions in that case. Section III extends to the dynamic case and develops two max-weight index coding techniques, one that requires knowledge of the arrival rates $(\lambda_1, \ldots, \lambda_M)$, and one that does not. The max-weight algorithms developed in this paper are new and contribute to the general theory of dynamic scheduling. They can be used in other types of networks where controllers make sequences of actions, each action taking a different number of slots and delivering a different vector of packets.

While the static index coding problem has been studied before [5]–[7], our work provides new insight even in the static case. We introduce a new directed bipartite demand graph that allows for arbitrary demand subsets and possibly "multiple multicast" situations, where some packets are desired by more than one user. We also form a useful weighted compressed graph that facilitates the solution to the minimum clearance time problem in certain cases. This extends the graph models in [7], which do not consider the possibility of multiple multicast sessions. Work in [7] develops a maximum acyclic subgraph bound on clearance time for problems without multiple multicast sessions. We extend this bound to our general problem using a different and independent proof technique. Further, we consider a class of broadcast relay problems for which the bound can be achieved with equality.

The next section introduces index coding in the static case, shows its relation to a bipartite demand graph, and presents the acyclic subgraph bound. Section III introduces the general dynamic formulation and develops our max-weight algorithms. Section IV considers an important class of broadcast relay networks for which a simple set of codes are optimal.

II. Static Minimum Clearance Time Problem

This section introduces the index coding problem in the static case, where we want to clear a fixed batch of packets in minimum time. Consider a wireless system with N users, P packets,

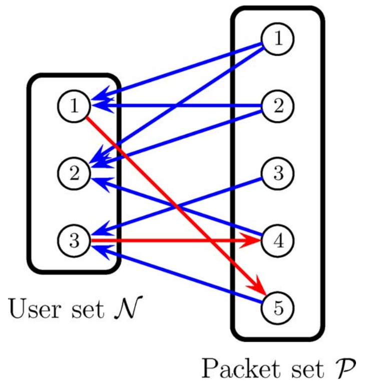

Fig. 1. Example directed bipartite demand graph with three users and five packets.

and a single broadcast station. We assume N and P are positive integers. Let $\mathcal{N}$ and $\mathcal{P}$ represent the set of users and packets, respectively,

$$\mathcal{N} = \{1, \ldots, N\} \qquad \mathcal{P} = \{1, \ldots, P\}.$$

The broadcast station has all packets in the set $\mathcal{P}$. Each user $n \in \mathcal{N}$ *has* an arbitrary subset of packets $\mathcal{H}_n \subseteq \mathcal{P}$, and wants to receive an arbitrary subset of packets $\mathcal{R}_n \subseteq \mathcal{P}$. Assume $\mathcal{H}_n \cap \mathcal{R}_n = \phi$, where ϕ represents the empty set. Assume that all packets consist of B bits, all packets are independent of each other, and the B-bit binary string for each packet is uniformly distributed over each of the 2^B possibilities.

We can represent this system by a directed bipartite demand graph $\mathcal{G}$ defined as follows (see Fig. 1):

1) User nodes $\mathcal{N}$ are on the left.
2) Packet nodes $\mathcal{P}$ are on the right.
3) A directed link (n, p) from a user node $n \in \mathcal{N}$ to a packet node $p \in \mathcal{P}$ exists if and only if user n has packet p. That is, if and only if $p \in \mathcal{H}_n$.
4) A directed link (p, n) from a packet node $p \in \mathcal{P}$ to a user node $n \in \mathcal{N}$ exists if and only if user n wants to receive packet p. That is, if and only if $p \in \mathcal{R}_n$.

As an example for the three-user, five-packet graph of Fig. 1, the *have* and *receive* sets for nodes 1 and 2 are

$$\mathcal{H}_1 = \{5\} \qquad \mathcal{R}_1 = \{1, 2\}$$
$$\mathcal{H}_2 = \phi \qquad \mathcal{R}_2 = \{1, 2, 4\}.$$

We restrict attention to packets that at least one node wants. Thus, without loss of generality, throughout we assume the graph $\mathcal{G}$ is such that all packet nodes $p \in \mathcal{P}$ contain at least one outgoing link. Thus,

$$\mathcal{P} = \{1, \ldots, P\} = \cup_{n=1}^{N} \mathcal{R}_n. \tag{1}$$

In this static problem, the broadcast station has all packets in the set $\mathcal{P}$ at time 0, and no more packets ever arrive. Every slot $t \in \{0, 1, 2, \ldots\}$ the broadcast station can transmit one B-bit message over the broadcast channel. This message is received without error at all of the user nodes in the set $\mathcal{N}$. The goal is for the broadcast station to send messages until all nodes receive the packets they desire.

Define a mission-completing coding action with T slots to be a sequence of messages that the broadcast station transmits over the course of T slots, such that all users are able to decode their desired packets at the end of the T slots. We restrict attention to deterministic zero-error codes that enable correct decoding with probability 1. The initial information held by each user $n \in \mathcal{N}$ is given by the set of packets $\mathcal{H}_n$ (possibly empty). Let $\mathcal{M} \triangleq \{M_1, \ldots, M_T\}$ represent the messages transmitted by the broadcast station over the course of the T slot coding action. At the end of this action, each node $n \in \mathcal{N}$ has information $\{\mathcal{H}_n, \mathcal{M}\}$. Because the coding action is assumed to complete the mission, this information is enough for each node n to decode its desired packets $\mathcal{R}_n$. That is, we can write

$$\{\mathcal{H}_n, \mathcal{M}\} \iff \{\mathcal{H}_n, \mathcal{M}, \mathcal{R}_n\} \tag{2}$$

where the above represents equivalence in the information set, meaning that the information on the left-hand side can be perfectly reconstructed from the information on the right-hand side, and vice versa. Clearly the information on the left in (2) is a subset of the information on the right, and hence can trivially be reconstructed. The information on the right in (2) can be reconstructed from that on the left because the code is mission-completing.

For a given graph $\mathcal{G}$ with P packet nodes, define $T_{min}(\mathcal{G})$ as the minimum clearance time of the graph, being the minimum number of slots required to complete the mission, considering all possible coding techniques (including nonlinear codes). Clearly, $T_{min}(\mathcal{G}) \leq P$. Our goal is to understand $T_{min}(\mathcal{G})$.

For a directed graph, we say that a simple directed cycle of length L is a sequence of nodes $\{x_1, x_2, \ldots, x_L, x_1\}$ such that (x_i, x_{i+1}) is a link in the graph for all $i \in \{1, \ldots, L-1\}$, (x_L, x_1) is a link in the graph, and all nodes $\{x_1, \ldots, x_L\}$ involved in the cycle are distinct. For simplicity, throughout this paper we use the term *cycle* to represent a simple directed cycle. We say that the graph $\mathcal{G}$ is *acyclic* if it contains no cycles. Note that directed acyclic graphs have a much different structure than undirected acyclic graphs. Indeed, the graph in Fig. 1 is acyclic even though its undirected counterpart (formed by replacing all directed links with undirected links) has cycles.

Our first result is to prove that if the directed bipartite demand graph $\mathcal{G}$ is acyclic, then coding cannot reduce the minimum clearance time. A related result was proven in [7] using a different graph structure for the special case without "multiple multicasts," so that each packet is desired by at most one user. That result uses an argument based on machinery of the mutual information function. It also treats a more general case where codes can have errors. Further, it is developed as a consequence of a more general and more complex result. Our paper restricts to zero-error codes, but allows the possibility of multiple-multicast sessions. We also use a different proof technique, developed independently, which emphasizes the logical consequences of users being able to decode their information. Our proof uses only the following two facts.

Fact 1: Every directed acyclic graph with a finite number of nodes has at least one node with no outgoing links. Such a node is called a "leaf" node.

Fact 2: If the graph contains only one user node, then $T_{min}(\mathcal{G}) = P$, where P is the number of packets that this user desires.

Fact 1 follows simply by starting at any node in the graph and traversing a path from node to node, using any outgoing link, until we find a leaf node (such a path cannot continue forever because the graph is finite and has no cycles). Fact 2 is a basic information theory observation about the capacity of a single error-free link.

Theorem 1: If the graph $\mathcal{G}$ is acyclic, then $T_{min}(\mathcal{G}) = P$, where P is the total number of packets in the graph.

Proof: See Appendix A. □

As an example, because the graph $\mathcal{G}$ in Fig. 1 is acyclic, we have $T_{min}(\mathcal{G}) = 5$. Theorem 1 shows that coding cannot help if $\mathcal{G}$ is acyclic, so that the best one can do is just transmit all packets one at a time. Therefore, any type of coding must exploit cycles on the demand graph.

A. Lower Bounds From Acyclic Subgraphs

Theorem 1 provides a simple lower bound on $T_{min}(\mathcal{G})$ for any graph $\mathcal{G}$. Consider a graph $\mathcal{G}$, and form a subgraph $\mathcal{G}'$ by performing one or more of the following *pruning operations*:

1) Remove a packet node, and all of its incoming and outgoing links.
2) Remove a user node, and all of its incoming and outgoing links.
3) Remove a packet-to-user link (p, n).

After performing these operations, we must also delete any residual packets that have no outgoing links. Any sequence of messages that completes the mission for the original graph $\mathcal{G}$ will *also* complete the mission for the subgraph $\mathcal{G}'$. This leads to the following simple lemma.

Lemma 1: For any subgraph $\mathcal{G}'$ formed from a graph $\mathcal{G}$ by one or more of the above pruning operations, we have

$$T_{min}(\mathcal{G}') \leq T_{min}(\mathcal{G}).$$

Combining this lemma with Theorem 1, we see that we can take a general graph $\mathcal{G}$ with cycles, and then perform the above pruning operations to reduce to an acyclic subgraph $\mathcal{G}'$. Then, $T_{min}(\mathcal{G})$ is lower bounded by the number of packets in this subgraph. Thus, the best lower bound corresponds to the acyclic subgraph generated from the above operations, and that has the largest number of remaining packets. Note that the above pruning operations do not include the removal of a user-to-packet link (n, p) (without removing either the entire user or the entire packet), because such links represent side information that can be helpful to the mission.

B. Cyclic Code Actions

Because the general index coding problem is difficult, it is useful to restrict the solution space to consider only sequences of simple types of coding actions. Recall that coding actions must exploit cycles. One natural action is the following: suppose there is a cycle in $\mathcal{G}$ that involves a subset of K users. For simplicity, label the users $\{1, \ldots, K\}$. In the cycle, user 2 wants to receive a packet X_1 that user 1 has, user 3 wants to receive a packet

X_2 that user 2 has, and so on. Finally, user 1 wants to receive a packet X_K that user K has. The structure can be represented by

$$1 \to 2 \to 3 \to \ldots \to K \to 1 \tag{3}$$

where an arrow from one user to another means the left user has a packet the right user wants. Of course, the users in this cycle may want many other packets, but we are restricting attention only to the packets $X_1, \ldots, X_K$. Assume these packets are all distinct.

In such a case, we can satisfy all K users in the cycle with the following $K-1$ transmissions: for each $k \in \{1, \ldots, K-1\}$, the broadcast station transmits a message $M_k \triangleq X_k + X_{k+1}$, where addition represents the mod-2 summation of the bits in the packets. Each user $k \in \{2, \ldots, K\}$ receives its desired information by adding M_{k-1} to its side information

$$X_k + M_{k-1} = X_k + (X_{k-1} + X_k) = X_{k-1}.$$

Finally, user 1 performs the following computation (using the fact that it already has packet X_1):

$$\begin{aligned} &X_1 + M_1 + M_2 + \ldots + M_{K-1} \\ &= X_1 + (X_1 + X_2) + (X_2 + X_3) + \cdots + (X_{K-1} + X_K) \\ &= (X_1 + X_1) + (X_2 + X_2) + \cdots + (X_{K-1} + X_{K-1}) \\ &\quad + X_K \\ &= X_K. \end{aligned}$$

Thus, such an operation can deliver K packets in only $K-1$ transmissions. We call such an action a K-cycle coding action. We define a 1-cycle coding action to be a direct transmission. Note that 2-cycle coding actions are the most "efficient," having a packet/transmission efficiency ratio of $2/1$, compared to $K/(K-1)$ for $K \geq 2$, which approaches 1 (the efficiency of a direct transmission) as $K \to \infty$. While it is generally suboptimal to restrict to such cyclic coding actions, doing so can still provide significant gains in comparison to direct transmission. Further, we show in Section IV that such actions are optimal for certain classes of broadcast relay problems.

C. Edge-Disjoint Cycles

Consider any bipartite demand graph $\mathcal{G}$ with N users and P packets. Define a *multicast packet* as a packet node $p \in \mathcal{P}$ that has more than one outgoing edge (so that this packet must be delivered to more than one user). Define a *unicast packet* as a packet node $p \in \mathcal{P}$ that has only one outgoing edge. Consider any collection of distinct edge-disjoint cycles in $\mathcal{G}$ that involve only unicast packets. Define the *size* of the collection as the number of cycles in the collection. Note that no two cycles in this collection can share a packet node. That is because any unicast packet has exactly one outgoing edge, so two cycles that share the packet must also share that edge (and hence cannot be edge-disjoint). Define $C(\mathcal{G})$ as the size of the largest collection of edge-disjoint cycles in $\mathcal{G}$ that involve only unicast packets.

Lemma 2: Cyclic coding actions can be used to achieve a clearance time of $P - C(\mathcal{G})$, and hence

$$T_{min}(\mathcal{G}) \leq P - C(\mathcal{G}).$$

Proof: Identify a collection of $C(\mathcal{G})$ edge-disjoint cycles that do not involve any multicast packets. Define p_c as the number of packets in the cth cycle of this collection. For each cycle c, use a p_c-cycle coding action to deliver all p_c packets of this cycle using $p_c - 1$ slots. This takes $\sum_{c=1}^{C(\mathcal{G})}[p_c - 1]$ slots. Since none of these packets is a multicast packet, none have to be delivered to any additional users. There are $P - \sum_{c=1}^{C(\mathcal{G})} p_c$ remaining packets, and these can be delivered sequentially using direct (uncoded) transmission. This completes the mission in the following time:

$$\left(\sum\nolimits_{c=1}^{C(\mathcal{G})}[p_c - 1]\right) + \left(P - \sum\nolimits_{c=1}^{C(\mathcal{G})} p_c\right) = P - C(\mathcal{G}).$$

□

Lemma 3: If all distinct cycles of $\mathcal{G}$ are edge-disjoint and involve only unicast packets, then

$$T_{min}(\mathcal{G}) = P - C(\mathcal{G}).$$

Proof: Lemma 2 shows that $T_{min}(\mathcal{G}) \leq P - C(\mathcal{G})$. It remains to show the reverse inequality. If $C(\mathcal{G}) = 0$, then $\mathcal{G}$ is acyclic and $T_{min}(\mathcal{G}) = P$ by Theorem 1. Otherwise, prune $\mathcal{G}$ to form an acyclic graph as follows: there are $C(\mathcal{G}) > 0$ cycles. For each cycle, choose a single packet that participates in the cycle. All $C(\mathcal{G})$ of these chosen packets are distinct. Delete all of these packets (and their corresponding incoming and outgoing links) to form a pruned graph $\mathcal{G}'$. This pruned graph is acyclic and has exactly $P - C(\mathcal{G})$ packets, and so $T_{min}(\mathcal{G}') = P - C(\mathcal{G})$ by Theorem 1. However, these pruning operations ensure $T_{min}(\mathcal{G}') \leq T_{min}(\mathcal{G})$ by Lemma 1. □

Appendix E presents additional clearance time results in terms of a weighted compressed graph $\mathcal{WC}(\mathcal{G})$ that involves only user nodes.

D. Simple Coding Gain Analysis

Here, we present a coding gain analysis for a randomly formed graph $\mathcal{G}$. Consider a system where each user desires a single unicast packet, and independently has each other packet in its cache with probability θ. Let $\{1, \ldots, N\}$ represent the set of users and let $\{p_1, \ldots, p_N\}$ represent the set of packets. Assume each user $n \in \{1, \ldots, N\}$ desires only packet p_n, and has packet p_i in its cache with probability θ (independently for each $i \neq n$). Assume N is even, and consider the following $(N/2)$-step method for greedily finding a collection of edge-disjoint 2-cycles:

1) *Step 1*: Define $\mathcal{N}_1$ as the set of all users $\{1, \ldots, N\}$. Define $u_1 = 1$. Greedily select any 2-cycle that involves user u_1 and exactly one other user $\tilde{u}_1 \in \mathcal{N}_1$. If such a 2-cycle exists, define $X_1 = 1$ and define $\mathcal{N}_2 = \mathcal{N}_1 - \{u_1, \tilde{u}_1\}$. Else, define $X_1 = 0$ and define $\mathcal{N}_2 = \mathcal{N}_1 - \{u_1\}$.
2) *Step* $k \in \{2, \ldots, N/2\}$: Define u_k as the lowest index user in the set $\mathcal{N}_k$. Greedily select any 2-cycle that involves user u_k and another user $\tilde{u}_k \in \mathcal{N}_k$. If such a 2-cycle exists, define $X_k = 1$ and define $\mathcal{N}_{k+1} = \mathcal{N}_k - \{u_k, \tilde{u}_k\}$. Else, define $X_k = 0$ and define $\mathcal{N}_{k+1} = \mathcal{N}_k - \{u_k\}$.

At each step $k \in \{1, \ldots, N/2\}$, the set $\mathcal{N}_k$ consists of users that have not been pairwise tested for a 2-cycle on any previous step. This algorithm finds a collection of X distinct 2-cycles,

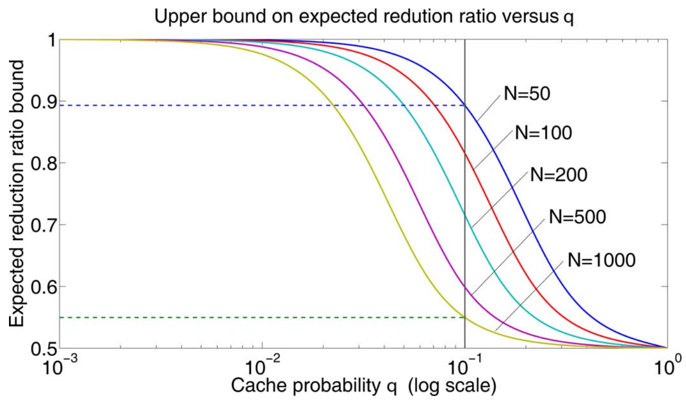

Fig. 2. Expected reduction ratio bound for greedy 2-cycle coding.

where $X = \sum_{k=1}^{N/2} X_k$. All 2-cycles involve distinct user nodes and hence they are edge-disjoint. This method does not necessarily produce the *largest* collection of edge-disjoint cycles, and so $X \leq C(\mathcal{G})$. Taking expectations gives

$$\mathbb{E}[X] = \sum_{k=1}^{N/2} \mathbb{E}[X_k] = \sum_{k=1}^{N/2} Pr[X_k = 1].$$

Two particular users n and m can be used to form a 2-cycle if the graph $\mathcal{G}$ has edges (n, p_m) and (m, p_n). The probability of this is θ^2, and this event is independent over different user pairs. Then, $Pr[X_1 = 1] = 1 - (1-\theta^2)^{N-1}$. Further, at each step $k \in \{1, \ldots, N/2\}$, the set $\mathcal{N}_k$ has at least $N - 2(k-1)$ users (including user u_k), and so for each $k \in \{1, \ldots, N/2\}$ we have

$$Pr[X_k = 1] \geq 1 - (1-\theta^2)^{N-2(k-1)-1}.$$

Thus,

$$\begin{aligned}\mathbb{E}[X] &\geq \sum_{k=1}^{N/2} [1 - (1-\theta^2)^{N-2(k-1)-1}] \\ &= \frac{N}{2} - \sum_{k=1}^{N/2} (1-\theta^2)^{2k-1} \\ &= \frac{N}{2} - \left[\frac{(1-\theta^2)(1-(1-\theta^2)^N)}{1-(1-\theta^2)^2}\right].\end{aligned}$$

Since we have X edge-disjoint 2-cycles, we can use 2-cycle coding actions to reduce the number of required transmissions from N to $N - X$. This gives a reduction ratio of $(N-X)/N = 1 - X/N$. Thus, the expected value of this ratio satisfies the following upper bound:

$$1 - \frac{\mathbb{E}[X]}{N} \leq \frac{1}{2} + \frac{1}{N}\left[\frac{(1-\theta^2)(1-(1-\theta^2)^N)}{1-(1-\theta^2)^2}\right].$$

For any $\theta > 0$, the upper bound converges to $1/2$ as $N \to \infty$. Fig. 2 plots this upper bound as a function of θ for different values of N. For example, if $\theta = 0.1$, the upper bound is 0.8930 for $N = 50$ (ensuring at least a 10.7% improvement for this case), and is 0.5497 when $N = 1000$ (ensuring at least a 45.03% improvement for this case).

E. Examples of Stronger Coding Actions

Improvements beyond a factor of 2 can be achieved by considering stronger coding actions. Here is a simple but important example: suppose user 1 wants packet A and has packets B and C, user 2 wants packet B and has packets A and C, and user 3 wants packet C and has packets A and B. These demands can be fulfilled with the single transmission $A + B + C$, being a binary XOR of packets A, B, C. The efficiency ratio of this action is $3/1$, which is larger than the efficiency of any K-cycle coding action.

A more general class of actions involve multicasting to a cluster of users: let $\mathcal{K}$ be a set of K distinct packets. Let $\mathcal{U}$ be the set of users who desire at least one packet in the set $\mathcal{K}$. For each user $u \in \mathcal{U}$, define $h_u(\mathcal{K})$ as the number of packets in $\mathcal{K}$ that user u already has as side information, so that $h_u(\mathcal{K}) \in \{0, 1, \ldots, K-1\}$. Define $h_{min}(\mathcal{K}) = \min_{u \in \mathcal{U}} h_u(\mathcal{K})$. Reed–Solomon erasure coding over finite fields can be used to correctly deliver *all* K packets to *all* users in the cluster $\mathcal{U}$ with $K - h_{min}(\mathcal{K})$ slots [14]. This can also be done using random codes with a success probability that can be made arbitrarily close to 1 by choosing a suitably large field size [16]. This type of coding is used in the *partition multicast* index coding heuristic in [17].

III. Dynamic Index Coding

Now consider a dynamic setting where the broadcast station randomly receives packets from M traffic flows. Each flow $m \in \{1, \ldots, M\}$ contains fixed-length packets that must be delivered to a subset $\mathcal{N}_m$ of the users, and these packets are contained as side-information in a subset $\mathcal{S}_m$ of the users. Assume $\mathcal{N}_m \cap \mathcal{S}_m = \phi$, since a user $n \in \mathcal{N}_m$ who wants a particular packet clearly does not already have this packet as side information. In the general case, M can be the number of all possible disjoint subset pair combinations. However, typically the value of M will be much smaller than this, such as when each traffic flow represents a stream of packets from a very large file, and there are only M active file requests.

Assume time is slotted with unit slots $t \in \{0, 1, 2, \ldots\}$, and let $\boldsymbol{A}(t) = (A_1(t), \ldots, A_M(t))$ be the number of packets that arrive from each flow on slot t. For simplicity of exposition, assume the vector $\boldsymbol{A}(t)$ is i.i.d. over slots with expectation

$$\mathbb{E}[\boldsymbol{A}(t)] = \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$$

where λ_m is the arrival rate of packets from flow m, in units of packets/slot. Assume that second moments of $A_m(t)$ are bounded for each flow m. Packets of each flow m are stored in a separate queue kept at the broadcast station, and exit the queue upon delivery to their intended users.

We now segment the timeline into variable length frames, each frame consisting of an integer number of slots. At the beginning of each frame r, the network controller chooses a coding action $\alpha[r]$ within an abstract set $\mathcal{A}$ of possible actions. For each $\alpha \in \mathcal{A}$, there is a *frame size* $T(\alpha)$ and a *clearance vector* $\boldsymbol{\mu}(\alpha)$. The frame size $T(\alpha)$ is the number of slots required to implement action α, and is assumed to be a positive integer. The clearance vector $\boldsymbol{\mu}(\alpha)$ has components $(\mu_1(\alpha), \ldots, \mu_M(\alpha))$, where $\mu_m(\alpha)$ is the number of type m packets delivered as a result of action α. We assume $\mu_m(\alpha)$ is a nonnegative integer. When frame r ends, a new frame starts and the controller chooses a (possibly new) action $\alpha[r+1] \in \mathcal{A}$. We assume each coding action only uses packets that are delivered as a result of that action, so that there is no "partial information" that can be exploited on future frames. We further assume there are a finite (but arbitrarily large) number of coding actions in the set $\mathcal{A}$, and that there are positive numbers T_{max} and μ_{max} such that $1 \leq T(\alpha) \leq T_{max}$ and $0 \leq \mu(\alpha) \leq \mu_{max}$ for all $\alpha \in \mathcal{A}$.

Assume that frame 0 starts at time 0. Define $t[0] = 0$, and for $r \in \{0, 1, 2, \ldots\}$ define $t[r]$ as the slot that starts frame r. Let $\boldsymbol{Q}[r] = (Q_1[r], \ldots, Q_M[r])$ be the queue backlog vector at the beginning of each frame $r \in \{0, 1, 2, \ldots\}$. Then, $Q_m[0] = 0$ for all $m \in \{1, \ldots, M\}$, and

$$Q_m[r+1] = \max[Q_m[r] - \mu_m(\alpha[r]), 0] + arrivals_m[r] \quad (4)$$

where $arrivals_m[r]$ is the number of type m arrivals during frame r:

$$arrivals_m[r] \triangleq \sum_{\tau=t[r]}^{t[r]+T(\alpha[r])-1} A_m(\tau). \quad (5)$$

The $\max[\cdot, 0]$ operator in the queue update (4) in principle allows actions $\alpha[r] \in \mathcal{A}$ to be chosen independently of the queue backlog at the beginning of a frame. In this case, if the action $\alpha[r]$ attempts to deliver one or more packets from queues that are empty, *null* packets are created and delivered. In practice, these null packets do not need to be delivered.

Our focus is on index coding problems with action sets $\mathcal{A}$ defined by a specific set of coding options, such as the set of all cyclic coding actions. For example, an action α that is a 2-cycle coding action that uses packets of type m and k has $T(\alpha) = 1$ and $\boldsymbol{\mu}(\alpha)$ being a binary vector with 1s in entries m and k and zeros elsewhere. However, the above model is general and can also apply to other types of problems, such as multihop networks where actions $\alpha \in \mathcal{A}$ represent some sequence of multihop network coding.

A. Code-Constrained Capacity Region

We say that queue $Q_m[r]$ is *rate stable* if

$$\lim_{R\to\infty} \frac{Q_m[R]}{R} = 0 \quad \text{(with probability 1)}.$$

It is not difficult to show that $Q_m[R]$ is rate stable if and only if the arrival rate λ_m is equal to the delivery rate of type m traffic [18]. The code-constrained capacity region $\Lambda_{\mathcal{A}}$ is the set of all (nonnegative) rate vectors $(\lambda_1, \ldots, \lambda_M)$ for which there exists an algorithm for selecting $\alpha[r] \in \mathcal{A}$ over frames that makes all queues rate stable.

Theorem 2: A (nonnegative) rate vector $\boldsymbol{\lambda}$ is in the code-constrained capacity region $\Lambda_{\mathcal{A}}$ if and only if there exist probabilities $p(\alpha)$ such that $\sum_{\alpha\in\mathcal{A}} p(\alpha) = 1$ and

$$\lambda_m \leq \frac{\sum_{\alpha\in\mathcal{A}} p(\alpha)\mu_m(\alpha)}{\sum_{\alpha\in\mathcal{A}} p(\alpha)T(\alpha)} \qquad \forall m \in \{1, \ldots, M\}. \quad (6)$$

Proof: The proof that such probabilities $p(\alpha)$ necessarily exist whenever $\boldsymbol{\lambda} \in \Lambda_{\mathcal{A}}$ is given in Appendix B. Below we prove sufficiency. Suppose such probabilities $p(\alpha)$ exist that satisfy (6). We want to show that $\boldsymbol{\lambda} \in \Lambda_{\mathcal{A}}$. To do so, we design an algorithm that makes all queues $Q_m[r]$ in (4) rate stable. By rate stability theory in [18], it suffices to design an algorithm that has a frame average arrival rate to each queue $Q_m[r]$ that is less than or equal to the frame average service rate (both in units of packets/frame).

Consider the algorithm that, every frame r, independently chooses action $\alpha \in \mathcal{A}$ with probability $p(\alpha)$. Let $\alpha^*[r]$ represent this random action chosen on frame r. Then, $\{T(\alpha^*[r])\}_{r=0}^{\infty}$ is an i.i.d. sequence, as is $\{\mu_m(\alpha^*[r])\}_{r=0}^{\infty}$ for each $m \in \{1, \ldots, M\}$. By the law of large numbers, the frame average arrival rate $\overline{arrivals}_m$ and the frame average service $\overline{\mu}_m$ (both in packets/frame) are equal to the following with probability 1:

$$\begin{aligned}\overline{\mu}_m &= \mathbb{E}\left[\mu_m(\alpha^*[r])\right] = \sum\nolimits_{\alpha\in\mathcal{A}} p(\alpha)\mu_m(\alpha)\\ \overline{arrivals}_m &= \lambda_m \mathbb{E}\left[T(\alpha^*[r])\right] = \lambda_m \sum\nolimits_{\alpha\in\mathcal{A}} p(\alpha)T(\alpha).\end{aligned}$$

We thus have for each $m \in \{1, \ldots, M\}$:

$$\frac{\overline{arrivals}_m}{\overline{\mu}_m} = \frac{\lambda_m \sum_{\alpha\in\mathcal{A}} p(\alpha)T(\alpha)}{\sum_{\alpha\in\mathcal{A}} p(\alpha)\mu_m(\alpha)} \leq 1$$

where the final inequality follows by (6). □

B. Max-Weight Queuing Protocols

Theorem 2 shows that all traffic can be supported by a stationary and randomized algorithm that independently chooses actions $\alpha^*[r] \in \mathcal{A}$ with probability distribution $p(\alpha)$. This does not require knowledge of the queue backlogs. However, computing probabilities $p(\alpha)$ that satisfy (6) would require knowledge of the arrival rates λ_m, and is a difficult computational task even if these rates are known. We provide two *dynamic* algorithms that use queue backlog information. These can also be viewed as online computation algorithms for computing probabilities $p(\alpha)$. Both are similar in spirit to the max-weight approach to dynamic scheduling in [19], but the variable frame lengths require a new approach that contributes to the general theory of dynamic scheduling.

Our first algorithm assumes knowledge of the arrival rates λ_m.

Max-Weight Code Selection Algorithm 1 (Known $\boldsymbol{\lambda}$): At the beginning of each frame r, observe the queue backlogs $Q_m[r]$ and perform the following:

1) Choose coding action $\alpha[r] \in \mathcal{A}$ as the maximizer of

$$\sum_{m=1}^{M} Q_m[r][\mu_m(\alpha[r]) - \lambda_m T(\alpha[r])] \quad (7)$$

where ties are broken arbitrarily.

2) Update the queue equation via (4).

The next algorithm uses a ratio rule, and does not require knowledge of the rates λ_m:

Max-Weight Code Selection Algorithm 2 (Unknown $\boldsymbol{\lambda}$): At the beginning of each frame r, observe the queue backlogs $Q_m[r]$ and perform the following:

1) Choose coding action $\alpha[r] \in \mathcal{A}$ as the maximizer of

$$\sum_{m=1}^{M} Q_m[r]\left[\frac{\mu_m(\alpha[r])}{T(\alpha[r])}\right] \tag{8}$$

where ties are broken arbitrarily.

2) Update the queue equation via (4).

Theorem 3: Suppose that $\boldsymbol{\lambda} \in \Lambda_{\mathcal{A}}$. Then, all queues are rate stable under either of the two algorithms above.

Proof: See Appendix C. □

It can further be shown that if there is a value ρ such that $0 \leq \rho < 1$, and if $\boldsymbol{\lambda} \in \rho\Lambda_{\mathcal{A}}$, being a ρ-scaled version of $\Lambda_{\mathcal{A}}$, then both algorithms give average queue size $O(1/(1-\rho))$. Thus, the average backlog bound increases to infinity as the arrival rates are pushed closer to the boundary of the capacity region. This is proven in Appendix D.

A related situation of variable frame lengths is treated in [20] for selecting variable-size coding schemes for transmission over a single point-to-point channel. It uses semi-Markov decision theory to develop a delay-optimal threshold rule. The threshold is approximated by truncating the state space of the queue and using knowledge of the arrival probability distribution to perform value iterations. That method cannot be extended to multiqueue systems without a curse of dimensionality. However, Algorithm 2 above can be applied to a multiqueue version of the problem in [20] to yield stability with $O(1/(1-\rho))$ average queue bounds, without requiring precomputations or knowledge of the arrival rates. Of course, Algorithm 2 does not necessarily optimize average delay.

C. Implementation for Cyclic Coding

Consider using the above algorithms in the special case when all packets are unicast packets and the set $\mathcal{A}$ consists of direct transmissions and cyclic coding actions. Fix $K \geq 2$. A K-cycle coding action must specify both the cycle of K users involved and the particular packet that each user sends. Consider a K-cycle of users given by

$$n_1 \rightarrow n_2 \rightarrow n_3 \ldots \rightarrow n_K \rightarrow n_1.$$

Let $p_1, p_2, \ldots, p_K$ be the packets chosen. Then, p_1 is contained as side information at user 1 and desired by user 2, p_2 is contained as side information at user 2 and desired by user 3, and so on. All packets $p_1, \ldots, p_K$ are unicast and are desired by different users, and hence are from distinct sessions. Let $m_1, \ldots, m_K$ be the corresponding sessions. Define $\mathcal{M}_1$ as the set of sessions in $\{1, \ldots, M\}$ that involve packets contained as side information at user 1 and desired by user 2. Define $\mathcal{M}_2$ as the set of sessions in $\{1, \ldots, M\}$ that involve packets contained as side information at user 2 and desired by user 3, and so on. Then, we must choose sessions $m_1, \ldots, m_K$ so that $m_i \in \mathcal{M}_i$ for all $i \in \{1, \ldots, K\}$. The value of (8) for this K-cycle coding action is

$$\frac{1}{K-1}\sum_{i=1}^{K} Q_{m_i}[r].$$

Given the particular cycle of K users, the above value is maximized over all possible session choices by greedily choosing $m_i = \arg\max_{m \in \mathcal{M}_i} Q_m[r]$ for each $i \in \{1, \ldots, K\}$. The value is then compared across different cycles of K users and different choices of K. A similar greedy selection can be done for maximizing the expression (7).

D. Implementation for Cluster Multicast Coding

Consider the cluster multicast coding actions described in Section II-F, which use Reed–Solomon erasure coding [14].[1] From the M queues, choose a cluster of K queues and label this set $\mathcal{K}$. We can send a single packet from each queue in the set $\mathcal{K}$ using $K - h_{min}(\mathcal{K})$ slots. This is a max-weight variation on the *partition multicast* method for index coding in [17]. The max-weight expression of Algorithm 2 for this cluster $\mathcal{K}$ on frame r is

$$\sum_{m \in \mathcal{K}} \frac{Q_m[r]}{K - h_{min}(\mathcal{K})}. \tag{9}$$

E. Example Simulation for 3 Users

Define $\mathcal{A}$ as the action space that restricts to direct transmissions, 2-cycle coding actions, 3-cycle coding actions, and the 1-slot $A+B+C$ coding action described in Section II-F. Fig. 3 presents simulation results for this action space and for a system with $N = 3$ users. We simulate algorithms 1 and 2 and compare against uncoded transmissions. All packets are intended for at most one user. Packets intended for user $n \in \{1, 2, 3\}$ arrive as independent Bernoulli processes with identical rates λ. We assume each packet is independently in the cache of the other two users with probability $\theta = 0.5$. Thus, there are four types of packets intended for user 1: Packets not contained as side information anywhere, packets contained as side information at user 2 only, packets contained as side information at user 3 only, and packets contained as side information at both users 2 and 3. Users 2 and 3 similarly have four traffic types, for a total of $M = 12$ traffic types.

Each data point in Fig. 3 represents a simulation over 5 million frames at a given value of λ. The figure plots the resulting total average number of packets in the system (summed over all 12 queues). The case of direct (uncoded) transmission is also shown. Uncoded transmission can support a maximum rate of $\lambda = 1/3$ (for a total traffic rate of 1). It is seen that algorithms 1 and 2 can significantly outperform uncoded transmission, achieving stability at rates up to $\lambda = 0.57$ (for a total traffic rate of 1.71). Remarkably, Algorithm 2 yields slightly less average backlog than Algorithm 1, even though it does not use traffic arrival rates λ_m. While not shown in the figure, it is interesting to note that when the $A + B + C$ coding option was

[1]Alternatively, we could use random coding as in [16] if the system can tolerate a small but nonzero error probability.

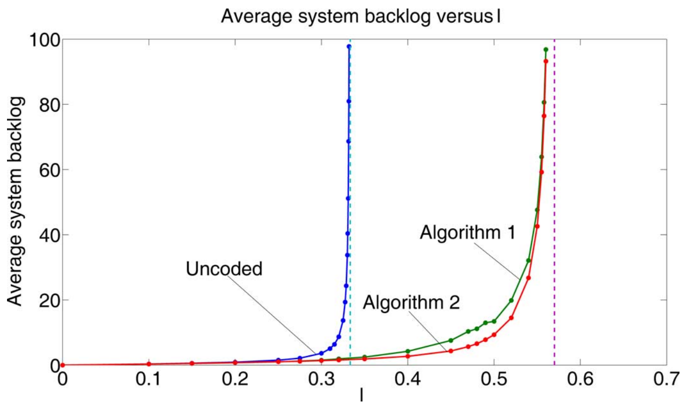

Fig. 3. Simulation of dynamic index coding for a three user system. Average system backlog is in units of packets and includes all queues in the system.

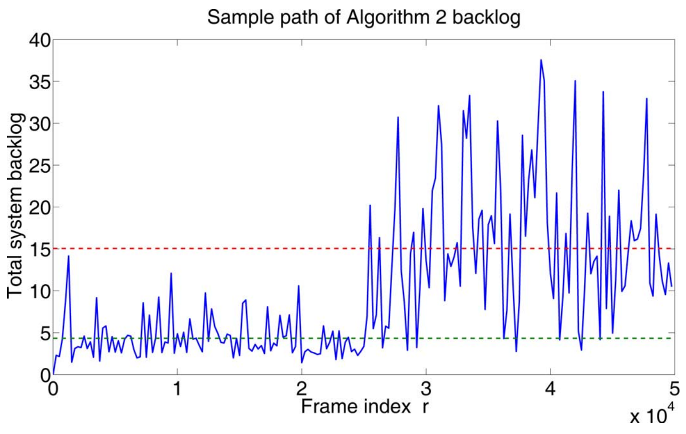

Fig. 4. Sample path of total backlog for Algorithm 2 over 50 000 frames. Statistics change halfway through the simulation. The dashed horizontal lines are the average backlogs associated with the two different sets of statistics.

removed from the set $\mathcal{A}$, the vertical asymptote for algorithms 1 and 2 was at $\lambda = 0.5333$ rather than $\lambda = 0.57$.

Because Algorithm 2 does not require arrival rate information, it is highly adaptive to unexpected changes. Fig. 4 illustrates a sample path of total average backlog under Algorithm 2 in a scenario where the traffic changes halfway through the simulation. The simulation is over 50×10^3 frames. In the first half of the simulation, the arrival rate is $\lambda = 0.45$ and the cache probability is the same as before ($\theta = 0.5$). In the second half, the arrival rate increases to $\lambda = 0.5$ and the cache probability for packets intended for user 1 decreases to $\theta_1 = 0.4$ (the cache probabilities for traffic intended for the other users do not change). The algorithm quickly adapts and the average backlog settles into the larger value associated with the new statistics.

F. Example Simulation for 50 Users

Fig. 5 shows a simulation for 50 users using the cluster multicasting technique with Reed–Solomon erasure codes (as described in Section III-D). Specifically, the coding action space $\mathcal{A}$ restricts to using erasure codes on clusters of size 1, 2, 3, and 4. There are 50 sessions. Packets arrive from each session according to independent Bernoulli processes with rate λ. Each user wants packets from a distinct session. At time 0, the side information at each user is drawn randomly and independently with probability θ for each of the $M - 1$ other sessions that it

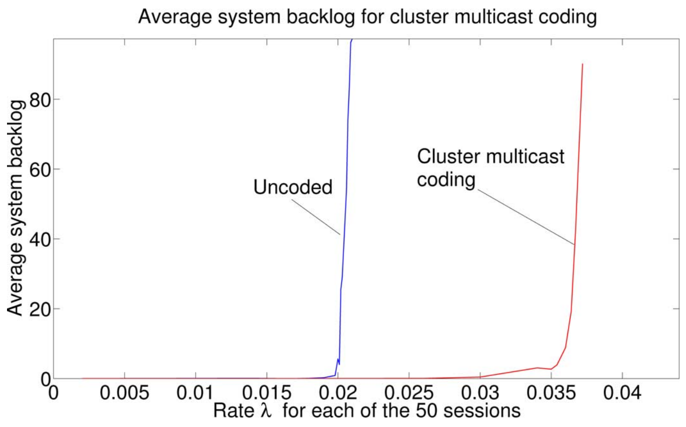

Fig. 5. Simulation over 10^5 frames of cluster multicast coding for 50 users and side information probability $\theta = 0.2$.

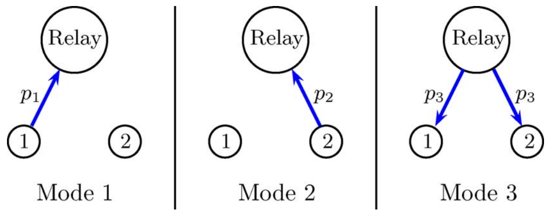

Fig. 6. Illustration of a 2-user broadcast relay system, with the three possible transmission modes shown. In mode 3, packet p_3 is received at both users.

does not desire. Results are plotted for the case $\theta = 0.2$. The case of uncoded transmission is also plotted. The simulation shows that erasure coding increases throughput by roughly 80% in this example.

IV. Broadcast Relay Networks

Consider now the following related problem: there are again N users and a single broadcast station. However, the broadcast station initially has no information, and acts as a relay to transfer independent unicast data between the users. Further, the users only know their own data, and initially have no knowledge of data sourced at other users. Time is again slotted, and every slot we can choose from one of $N + 1$ modes of transmission. The first N transmission modes involve an error-free packet transmission from a single user to the relay. The $(N+1)$th transmission mode is where the relay broadcasts a single packet that is received error-free at each of the N users. Fig. 6 illustrates an example system with two users, where the three possible transmission modes are shown. For simplicity, we assume the user transmissions cannot be overheard by other users, and the users first send all packets to the relay. The relay then can make coding decisions for its downlink transmissions.

A. Minimum Clearance Time Relay Problem

First consider a static problem where a batch of packets must be delivered in minimum time. Let P_{ij} represent the number of packets that user i wants to send to user j, where $i, j \in \{1, \ldots, N\}$. All packets are independent, and the total number of packets is P, where

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij}.$$

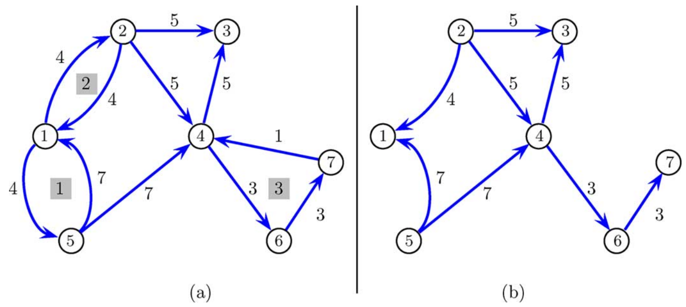

Fig. 7. (a) Graph $\mathcal{WC}(\mathcal{G})$ with three edge-disjoint cycles, and (b) its pruned graph $\mathcal{WC}(\mathcal{G}')$.

This problem is related to the index coding problem as follows: suppose on the first P slots, all users send their packets to the relay on the uplink channels. It remains for the relay to send all users the desired data, and these users have side information. The resulting side information graph $\mathcal{G}$ is the same as in the general index coding problem. However, it has the following *special structure*: the only user that has side information about a packet is the source user of the packet. Specifically,

1) Each packet is contained as side information in exactly one user. Thus, each packet node of $\mathcal{G}$ has a single incoming link from some user that is its source.
2) Each packet has exactly one user as its destination. Thus, each packet node of $\mathcal{G}$ has a single outgoing link to some user that is its destination.

This special structure leads to a simplified graphical model for demands, which we call the weighted compressed graph $\mathcal{WC}(\mathcal{G})$ of $\mathcal{G}$. The graph $\mathcal{WC}(\mathcal{G})$ is formed from $\mathcal{G}$ as follows: it is a directed graph defined on the user nodes $\mathcal{N}$ only, and contains a link (a, b) if and only if the original graph $\mathcal{G}$ specifies that user node a has a packet that user node b wants. Further, each link (a, b) is given a positive integer weight P_{ab}, the number of packets user a wants to send to user b. It is easy to show that $\mathcal{WC}(\mathcal{G})$ is acyclic if and only if $\mathcal{G}$ is acyclic. Hence, coding can only help if $\mathcal{WC}(\mathcal{G})$ contains cycles, and so $T_{min}(\mathcal{G}) = P$ whenever $\mathcal{WC}(\mathcal{G})$ is acyclic.

We say the weighted compressed graph $\mathcal{WC}(\mathcal{G})$ has *edge-disjoint cycles* if each of its links participates in at most one simple cycle. An example is shown in Fig. 7. Consider such a graph that has C edge-disjoint cycles. Let w_c^{min} be the min-weight link on each edge-disjoint cycle $c \in \{1, \ldots, C\}$.

Theorem 4: If the broadcast relay problem has a weighted compressed graph $\mathcal{WC}(\mathcal{G})$ with edge-disjoint cycles, then

$$T_{min}(\mathcal{G}) = P - \sum_{c=1}^{C} w_c^{min} \tag{10}$$

and so the full clearance time (including the P uplink transmissions) is the above number plus P. Further, optimality can be achieved over the class of cyclic coding actions, as described in Section II-C.

As an example, the graph $\mathcal{WC}(\mathcal{G})$ in Fig. 7(a) has $P = 48$, three edge-disjoint cycles with $w_1^{min} = 4$, $w_2^{min} = 4$, $w_3^{min} = 1$, and so $T_{min}(\mathcal{G}) = 48 - 4 - 4 - 1 = 39$.

Proof (Theorem 4): First prune the graph $\mathcal{WC}(\mathcal{G})$ by removing the min-weight link on each of the edge-disjoint cycles (breaking ties arbitrarily). This removes the corresponding packets from the original graph $\mathcal{G}$ to produce a new graph $\mathcal{G}'$ with exactly $P - \sum_{c=1}^{C} w_c^{min}$ packets. The weighted compressed graph $\mathcal{WC}(\mathcal{G}')$ is the subgraph of $\mathcal{WC}(\mathcal{G})$ with the min-weight links on each edge-disjoint cycle removed [see Fig. 7(a) and (b)]. Both $\mathcal{G}'$ and $\mathcal{WC}(\mathcal{G}')$ are acyclic, and so

$$T_{min}(\mathcal{G}) \geq T_{min}(\mathcal{G}') = P - \sum_{c=1}^{C} w_c^{min}.$$

It remains only to construct a coding algorithm that achieves this lower bound. This can be done easily by using w_c^{min} separate cyclic coding actions for each of the edge-disjoint cycles (using a k-cycle coding action for any cycle of length k), and then directly transmitting the remaining packets. □

A subgraph of $\mathcal{WC}(\mathcal{G})$ is a graph on the same nodes $\mathcal{N}$ but with some link weights P_{ij} set to 0. The following lemma shows that the number of packets in any acyclic subgraph of $\mathcal{WC}(\mathcal{G})$ is a lower bound on the minimum clearance time.

Lemma 4: Let $\mathcal{WC}(\mathcal{G})$ be the weighted compressed graph for a broadcast relay problem with demand graph $\mathcal{G}$. If an acyclic subgraph of $\mathcal{WC}(\mathcal{G})$ contains P' packets, then $T_{min}(\mathcal{G}) \geq P'$.

Proof: The proof is similar to that of Theorem 4 and omitted for brevity. □

B. Traffic Structure and Optimality of Cyclic Coding

Suppose we have a broadcast relay problem with N users, packet matrix (P_{ij}), and with the following additional structure: each user $i \in \{1, \ldots, N\}$ wants to send data to only one other user. That is, the matrix P_{ij} has at most one nonzero entry in each row $i \in \{1, \ldots, N\}$. We now prove the resulting graph $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles. To see this, suppose it is not true. Then, there are two different cycles that share a link (a, b) that leads to a link (b, k) for cycle 1 and (b, m) for cycle 2, where $k \neq m$. This means node b has two outgoing links, a contradiction because matrix (P_{ij}) has at most one nonzero entry in row b, and hence at most one outgoing link from node b. We conclude that $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles, and so cyclic coding is optimal via Theorem 4.

A similar argument holds if each user wants to *receive* from at most one other user, so that P_{ij} has at most one nonzero entry in every column. Again, $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles, and so cyclic coding is optimal.

C. Optimality for $N \leq 3$

This section proves that cyclic coding is optimal for broadcast relay networks when there are only two or three users, regardless of the number of packets and the side information configurations. Assume there are $N = 3$ users (the case $N = 2$ follows as a special case when link weights into and out of node 3 are set to 0). The first panel of Fig. 8 illustrates the general graph $\mathcal{WC}(\mathcal{G})$ for the case $N = 3$. The link weights are P_{ij} for $i, j \in \{1, 2, 3\}$ $(i \neq j)$, where P_{ij} are nonnegative integers. The number of packets is $P = \sum_{i=1}^{N} \sum_{j=1}^{N} P_{ij}$. We construct a clearance time procedure that involves only direct transmission, 2-cycle coding actions, and 3-cycle coding actions. If the 3-node graph $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles, we are done (recall Theorem 4).

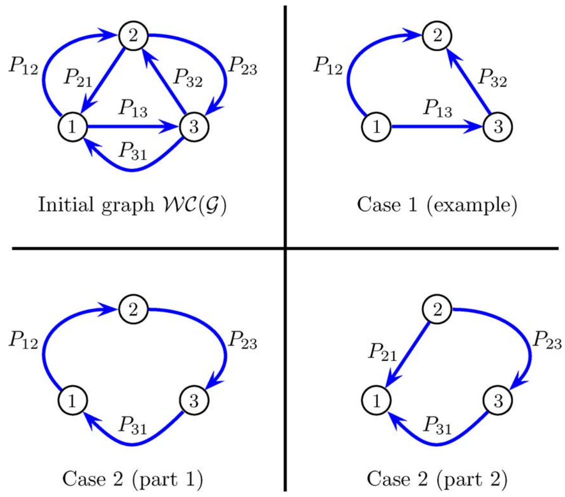

Fig. 8. Illustration of the general broadcast relay graph $\mathcal{WC}(\mathcal{G})$ with $N = 3$ users, and the two cases required for the proof.

Consider now the general case where cycles might share edges. Define min_{12}, min_{23}, min_{31} as the weight of the min-weight link for each of the three possible 2-cycles

$$\begin{aligned} min_{12} &= \min[P_{12}, P_{21}] \\ min_{23} &= \min[P_{23}, P_{32}] \\ min_{31} &= \min[P_{31}, P_{13}]. \end{aligned}$$

Now prune the graph $\mathcal{WC}(\mathcal{G})$ by removing the min-weight link for each 2-cycle. This results in a graph $\mathcal{WC}(\mathcal{G}')$ with three nodes and (at most) three links, and with a total number of packets equal to $P - min_{12} - min_{23} - min_{31}$, as shown in Cases 1 and 2 in the figure. We have two cases.

Case 1: The resulting graph $\mathcal{WC}(\mathcal{G}')$ is acyclic. An example of this case is shown as case 1 in Fig. 8. Thus, we know $T_{min}(\mathcal{G}) \geq T_{min}(\mathcal{G}') = P - min_{12} - min_{23} - min_{31}$. This clearance time bound can be achieved by using 2-cycle coding actions on each of the three 2-cycles, and then transmitting the remaining packets uncoded.

Case 2: The resulting graph $\mathcal{WC}(\mathcal{G}')$ consists of a single 3-cycle. The cycle must either be clockwise or counter-clockwise. Without loss of generality, assume clockwise (see Fig. 8, case 2 part 1). Note that

$$P_{12} \geq P_{21} \quad P_{23} \geq P_{32} \quad P_{31} \geq P_{13}. \tag{11}$$

This is because we have formed $\mathcal{WC}(\mathcal{G}')$ by removing the min-weight link on each of the three 2-cycles.

Let $z = \min[P_{12} - P_{21}, P_{23} - P_{32}, P_{31} - P_{13}]$, so that z is a nonnegative integer. Without loss of generality, assume the min value for z is achieved by link $(1, 2)$, so that $z = P_{12} - P_{21}$. To $\mathcal{WC}(\mathcal{G}')$, add back the link $(2, 1)$ (with weight P_{21}), and remove the link $(1, 2)$, to yield a graph $\mathcal{WC}(\mathcal{G}'')$ that is an acyclic subgraph of the original graph $\mathcal{WC}(\mathcal{G})$, as shown in case 2 part 2 of Fig. 8. The acyclic subgraph $\mathcal{WC}(\mathcal{G}'')$ contains exactly $P_{21} + P_{23} + P_{31}$ packets. Thus, the minimum clearance time of the original graph $\mathcal{WC}(\mathcal{G})$ is at least $P_{21} + P_{23} + P_{31}$. However, this can easily be achieved. Do the following: Perform 2-cycle coding actions on each of the three 2-cycles of the original graph $\mathcal{WC}(\mathcal{G})$, to remove a number of packets on each cycle equal to the min weight link of that cycle. This removes $2P_{21} + 2P_{32} + 2P_{13}$ packets in $P_{21} + P_{32} + P_{13}$ slots (recall the three min weights are given by (11)). Then perform 3-cycle coding actions to remove $3z$ packets in $2z$ slots. Then, perform direct transmission to remove the remaining packets, being a total of $x = P - 2P_{21} - 2P_{32} - 2P_{13} - 3z$. The total number of transmissions is

$$\begin{aligned} P_{21} &+ P_{32} + P_{13} + 2z + x \\ &= P_{21} + P_{32} + P_{13} + 2z \\ &\quad + P - 2P_{21} - 2P_{32} - 2P_{13} - 3z \\ &= P - P_{21} - P_{32} - P_{13} - z \\ &= P_{12} + P_{23} + P_{31} - z \\ &= P_{12} + P_{23} + P_{31} - (P_{12} - P_{21}) \\ &= P_{21} + P_{23} + P_{31} \end{aligned}$$

and so the above scheme is optimal.

D. Dynamic Broadcast Relay Scheduling

Now consider the dynamic case where packets from source user i and destination user j arrive with rate λ_{ij} packets/slot. Suppose we have an abstract set of coding actions $\mathcal{A}$, where each action involves a subset of packets, and first transmits these packets to the relay before any coding at the relay. Let $T(\alpha)$ be the number of slots to complete the action, and $(\mu_{ij}(\alpha))$ be the matrix of packets delivered by the action. It can be shown that capacity can be approached arbitrarily closely by repetitions of minimum-clearance time scheduling on large blocks of the incoming data (similar to the capacity treatment in [12] for a limit of large packet size). Hence, if $T_{min}(\mathcal{G})$ can be optimally solved using only cyclic-coding actions, then capacity is also achieved in the max-weight algorithms when $\mathcal{A}$ is restricted to cyclic-coding actions. It follows that such actions are optimal for rate matrices λ_{ij} with at most one nonzero entry per row, and for rate matrices λ_{ij} with at most one nonzero entry per column.

Similarly, it follows that cyclic coding is optimal for the case of $N = 2$ or $N = 3$. Thus, Theorem 2 establishes the full capacity region Λ in those cases. The case $N = 2$ yields a 2-D capacity region (being the set of all supportable 2×2 rate matrices with zeros on the diagonal). The case $N = 3$ yields a 6-D capacity region (being the set of all supportable 3×3 rate matrices with zeros on the diagonal).

E. Counterexamples

Can we minimize clearance time by grabbing any available 2-cycle, then any available 3-cycle if no 2-cycle is available, and so on? Not necessarily. A simple counterexample is shown in Fig. 9(a). The graph has five users and seven packets $\{A, \ldots, G\}$, where each link has a single packet. Using the middle 3-cycle $2 \to 3 \to 4 \to 2$ by transmitting $B + D$ and $D + E$ leaves a remaining acyclic graph with four packets, and hence would take four more transmissions, for a total of six slots. However, using the two side cycles (with two transmissions each) and then transmitting the remaining packet E clears everything in five slots, which is optimal because the

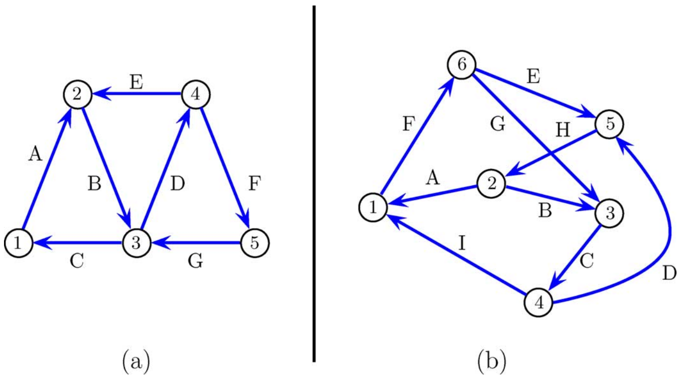

Fig. 9. Two example graphs $\mathcal{WC}(\mathcal{G})$ for broadcast relay problems.

maximum acyclic subgraph has five packets (just remove links B and D).

One may wonder if all broadcast relay graphs can be optimally cleared with cyclic coding. Section IV-C shows this is true for $N = 2$ and $N = 3$. However, this is not true in general. Fig. 9(b) shows a counterexample with $N = 6$. Suppose each link has a single packet, so that we have nine packets $\{A, \ldots, I\}$. It can be shown that the maximum acyclic subgraph has seven packets, and so $T_{min}(\mathcal{G}) \geq 7$, but the best cyclic coding method uses eight slots. Here is a way to achieve seven slots: send messages $M_1 = E + G + F$, $M_2 = H + E$, $M_3 = H + D$, $M_4 = A + B + H$, $M_5 = C + B$, $M_6 = C + G$, $M_7 = C + I + D$. The decodings at users $2, 3, 4, 5, 6$ are straightforward by combining their side information with just a single message. The decoding at user 1 is done as follows: $M_1 + M_2 + M_3 + M_6 + M_7 = F + I$. Since user 1 knows F, it can decode I. $M_3 + M_4 + M_5 + M_7 = A + I$, since user 1 knows I it can get A.

V. Conclusion

This paper presents a dynamic approach to index coding. This problem is important for future wireless communication where instances of side information can be exploited. While optimal index coding for general problems seems to be intractable, this paper develops a code-constrained capacity region that restricts actions to a prespecified set of codes. Two max-weight algorithms were developed that support randomly arriving traffic whenever the arrival rate vector is inside the code-constrained capacity region. The first algorithm requires knowledge of the rate vector, and the second does not. Simulations verify network stability up to the boundary of the code-constrained capacity region and illustrate improvements in both throughput and delay over uncoded transmission.

It was shown that, for coding to provide gains in comparison to direct transmission, it must exploit cycles in the demand graph. A simple set of codes based on cycles was considered and shown to be optimal (so that the code-constrained capacity region is equal to the unconstrained capacity region) for certain classes of problems. This was proven by providing coding techniques that match a fundamental acyclic subgraph bound. These results add to the theory of information networks, and can be used to improve efficiency in communication systems.

Appendix A
Proof of Theorem 1

Proof (Theorem 1): We already know that $T_{min}(\mathcal{G}) \leq P$. It suffices to show that $T_{min}(\mathcal{G}) \geq P$. Consider any mission-completing coding action that takes T slots. We show that $T \geq P$. Let $\mathcal{M}$ be the sequence of messages transmitted. Then, every node $n \in \mathcal{N}$ is able to decode its desired packets, being packets in the set $\mathcal{R}_n$, from the information $\{\mathcal{H}_n, \mathcal{M}\}$, being the information it has at the end of the coding action. That is, we have

$$\{\mathcal{H}_n, \mathcal{M}\} \iff \{\mathcal{H}_n, \mathcal{M}, \mathcal{R}_n\} \quad \forall n \in \{1, \ldots, N\}. \tag{12}$$

Because the graph is acyclic, there must be at least one node with no outgoing links (by Fact 1). Choose such a node, and label this node n_1. The node n_1 cannot be a packet node, because we have assumed that all packet nodes have outgoing links. Thus, $n_1 \in \mathcal{N}$. Because node n_1 has no outgoing links, it has $\mathcal{H}_{n_1} = \phi$ and thus has no initial side information about any of the packets. Thus, it is able to decode all packets in the set $\mathcal{R}_{n_1}$ by the messages $\mathcal{M}$ alone. That is

$$\mathcal{M} \iff \{\mathcal{M}, \mathcal{R}_{n_1}\}. \tag{13}$$

We want to show that this node n_1 can decode *all* packets in the set $\mathcal{P}$, so that

$$\mathcal{M} \iff \{\mathcal{M}, \mathcal{P}\}. \tag{14}$$

If we can show that (14) holds, then the sequence of messages $\mathcal{M}$ is also sufficient to deliver P independent packets to node n_1, and node n_1 did not have any initial side information about these packets. Thus, the number of slots T used in the coding action must be at least P by Fact 2, proving the result. Thus, it suffices to prove (14).

We prove (14) by induction on k, for $k \in \{1, \ldots, N-1\}$: assume that there is a labeling of k distinct user nodes $\{n_1, n_2, \ldots, n_k\}$ such that

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\}. \tag{15}$$

This property holds for the base case $k = 1$ by (13). We now assume that (15) holds for a general $k \in \{1, \ldots, N-1\}$, and prove it must also hold for $k+1$. Take the graph $\mathcal{G}$, and delete the user nodes $\{n_1, \ldots, n_k\}$, also deleting all links outgoing from and incoming to these nodes. This may create packet nodes with no outgoing links: delete all such packet nodes. Note that all deleted packet nodes (if any) must be in the set $\{\mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\}$, being the set of packets desired by the users that are deleted. The resulting subgraph must still be acyclic, and hence it must have a node n_{k+1} with no outgoing links. This node must be a user node, as we have deleted all packet nodes with no outgoing links.

Because the user node n_{k+1} has no outgoing links, it either had $\mathcal{H}_{n_{k+1}} = \phi$ (so that it never had any outgoing links), or all of its outgoing links were pointing to packet nodes that we have deleted, and so those packets were in the set $\{\mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\}$. That is, we must have $\mathcal{H}_{n_{k+1}} \subseteq \{\mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\}$. Therefore,

$$\{\mathcal{M}, \mathcal{H}_{n_{k+1}}\} \subseteq \{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\}. \tag{16}$$

However, at the end of the coding action, node n_{k+1} has exactly the information on the left-hand side of (16), and hence this information is sufficient to decode all packets in the set $\mathcal{R}_{n_{k+1}}$. Thus, the information on the right-hand side of (16) must *also* be sufficient to decode $\mathcal{R}_{n_{k+1}}$, so that

$$\{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}, \mathcal{R}_{n_{k+1}}\}.$$

But this together with (15) yields

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_k}, \mathcal{R}_{n_{k+1}}\}$$

which completes the induction step.

By induction over $k \in \{1, \ldots, N-1\}$, it follows that

$$\{\mathcal{M}\} \iff \{\mathcal{M}, \mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_N}\}. \quad (17)$$

However, by relabeling we have

$$\{\mathcal{R}_{n_1}, \ldots, \mathcal{R}_{n_N}\} = \{\mathcal{R}_1, \ldots, \mathcal{R}_N\} = \mathcal{P} \quad (18)$$

where the final equality holds by (1). Combining (17) and (18) proves (14). □

Appendix B
Proof of Necessity for Theorem 2

Let $\{\alpha[r]\}_{r=0}^{\infty}$ be a sequence of actions, chosen over frames, that makes all queues $Q_m[r]$ rate stable. We show there must exist probabilities $p(\alpha)$ that satisfy (6). For each positive integer R and each $m \in \{1, \ldots, M\}$, define $\overline{a}_m[R]$ and $\overline{\mu}_m[R]$ as the following averages over the first R frames:

$$\overline{a}_m[R] \triangleq \tfrac{1}{R}\sum_{r=0}^{R-1} arrivals_m[r]$$
$$\overline{\mu}_m[R] \triangleq \tfrac{1}{R}\sum_{r=0}^{R-1} \mu_m(\alpha[r])$$

where $arrivals_m[r]$ is defined in (5). Now define $\mathcal{F}(\alpha, R)$ as the set of frames $r \in \{0, \ldots, R-1\}$ that use action α, and define $|\mathcal{F}(\alpha, R)|$ as the number of these frames, so that $\sum_{\alpha \in \mathcal{A}} |\mathcal{F}(\alpha, R)| = R$. We then have

$$\overline{a}_m[R] = \sum_{\alpha \in \mathcal{A}} \frac{|\mathcal{F}(\alpha, R)|}{R} \times \frac{1}{|\mathcal{F}(\alpha, R)|} \sum_{r \in \mathcal{F}(\alpha, R)} arrivals_m[r] \quad (19)$$

$$\overline{\mu}_m[R] = \sum_{\alpha \in \mathcal{A}} \frac{|\mathcal{F}(\alpha, R)|}{R} \mu_m(\alpha). \quad (20)$$

The set $\mathcal{A}$ is finite. Thus, the values $\{(|\mathcal{F}(\alpha, R)|/R)\}_{R=1}^{\infty}$ can be viewed as an infinite sequence of bounded vectors (with entries indexed by α and dimension equal to the size of set $\mathcal{A}$) defined on the index $R \in \{1, 2, 3, \ldots\}$, and hence must have a convergent subsequence. Let R_k represent the sequence of frames on this subsequence, so that there are values $p(\alpha)$ for all $\alpha \in \mathcal{A}$ such that

$$\lim_{k \to \infty} |\mathcal{F}(\alpha, R_k)|/R_k = p(\alpha).$$

Further, by (20) we have for all $m \in \{1, \ldots, M\}$:

$$\lim_{k \to \infty} \overline{\mu}_m[R_k] = \sum_{\alpha \in \mathcal{A}} p(\alpha)\mu_m(\alpha). \quad (21)$$

Likewise, from (19) and the law of large numbers (used over each $\alpha \in \mathcal{A}$ for which $\lim_{k \to \infty} |\mathcal{F}(\alpha, R_k)| = \infty$, and noting that $arrivals_m[r]$ is i.i.d. with mean $T(\alpha)\lambda_m$ for all $r \in \mathcal{F}(\alpha, R)$) we have with probability 1:

$$\lim_{k \to \infty} \overline{a}_m[R_k] = \sum_{\alpha \in \mathcal{A}} p(\alpha)T(\alpha)\lambda_m. \quad (22)$$

Because $|\mathcal{F}(\alpha, R_k)|/R_k \geq 0$ for all $\alpha \in \mathcal{A}$ and all R_k, and $\sum_{\alpha \in \mathcal{A}} |\mathcal{F}(\alpha, R_k)|/R_k = 1$ for all R_k, the same holds for the limiting values $p(\alpha)$. That is, $p(\alpha) \geq 0$ for all $\alpha \in \mathcal{A}$, and $\sum_{\alpha \in \mathcal{A}} p(\alpha) = 1$. Because each queue $Q_m[r]$ is rate stable, we have with probability 1 that for all $m \in \{1, \ldots, M\}$:

$$\lim_{k \to \infty} \frac{Q_m[R_k]}{R_k} = 0. \quad (23)$$

However, from the queue update (4) we have for all $r \in \{0, 1, 2, \ldots\}$:

$$Q_m[r+1] \geq Q_m[r] - \mu_m(\alpha[r]) + arrivals_m[r].$$

Summing the above over $r \in \{0, 1, \ldots, R_k - 1\}$ and dividing by R_k yields

$$\frac{Q_m[R_k] - Q_m[0]}{R_k} \geq -\overline{\mu}_m[R_k] + \overline{a}_m[R_k].$$

Taking a limit as $k \to \infty$ and using (21)–(23) yields

$$0 \geq -\sum_{\alpha \in \mathcal{A}} p(\alpha)\mu_m(\alpha) + \lambda_m \sum_{\alpha \in \mathcal{A}} p(\alpha)T(\alpha). \quad (24)$$

This proves the result.

Appendix C
Proof of Theorem 3

We first prove rate stability for Algorithm 2, which uses a ratio rule. The proof for Algorithm 1 is simpler and is given after. We have the following preliminary lemma.

Lemma 5 (Sufficient Condition for Rate Stability [21]): Let $Q[r]$ be a nonnegative stochastic process defined over the integers $r \in \{0, 1, 2, \ldots\}$. Suppose there are constants B, C, D such that for all frames $r \in \{0, 1, 2, \ldots\}$ we have

$$\mathbb{E}\left[(Q[r+1] - Q[r])^2\right] \leq D \quad (25)$$
$$\mathbb{E}\left[Q[r]^2\right] \leq Br + C. \quad (26)$$

Then, $\lim_{r \to \infty} Q[r]/r = 0$ with probability 1.

The condition (25) is immediately satisfied in our system because second moments of queue changes over any frame are bounded. Thus, to prove rate stability, it suffices to show that (26) holds for all queues and all frames. That is, it suffices to prove the second moment of queue backlog grows at most linearly.

For each frame $r \in \{0,1,2,\ldots\}$, define the following quadratic function $L[r]$, called a *Lyapunov function*:

$$L[r] \triangleq \frac{1}{2}\sum_{m=1}^{M} Q_m[r]^2.$$

Define the *Lyapunov drift* $\Delta[r] \triangleq L[r+1] - L[r]$.

Lemma 6: Under any (possibly randomized) decision for $\alpha[r] \in \mathcal{A}$ that is *causal* (i.e., that does not know the future values of arrivals over the frame), we have for each frame r:

$$\begin{aligned}&\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right]\\ &\leq B + \sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha[r]) - \mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]\end{aligned}$$

where B is a finite constant that satisfies

$$B \geq \tfrac{1}{2}\sum_{m=1}^{M}\mathbb{E}\left[arrivals_m[r]^2 + \mu_m(\alpha[r])^2|\boldsymbol{Q}[r]\right].$$

Such a finite constant B exists because second moments of arrivals and service over a frame are bounded.

Proof: For simplicity of notation, define $b_m[r] \triangleq \mu_m(\alpha[r])$, and $a_m[r] \triangleq arrivals_m[r]$. The queue update equation is thus

$$Q_m[r+1] = \max[Q_m[r] - b_m[r], 0] + a_m[r].$$

For any nonnegative values Q, a, b we have

$$(\max[Q-b,0]+a)^2 \leq Q^2 + b^2 + a^2 + 2Q(a-b).$$

Using this and squaring the queue update equation yields

$$Q_m[r+1]^2 \leq Q_m[r]^2 + b_m[r]^2 + a_m[r]^2 + 2Q_m[r][a_m[r]-b_m[r]].$$

Summing over all m, dividing by 2, and taking conditional expectations yields

$$\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right] \leq B + \sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[a_m[r] - b_m[r]|\boldsymbol{Q}[r]\right]. \tag{27}$$

Now note that[2]

$$\begin{aligned}\mathbb{E}\left[a_m[r]|\boldsymbol{Q}[r]\right] &= \mathbb{E}\left[\sum_{\tau=t[r]}^{t[r]+T(\alpha[r])-1} A_m(\tau)|\boldsymbol{Q}[r]\right]\\ &= \mathbb{E}\left[\lambda_m T(\alpha[r])|\boldsymbol{Q}[r]\right]. \end{aligned}\tag{28}$$

Plugging this identity into (27) proves the result. □

We now prove that Algorithm 2 yields rate stability.

Proof (Theorem 3—Stability Under Algorithm 2): Suppose that Algorithm 2 is used, so that we choose $\alpha[r]$ in every frame r via (8). We first claim that for each frame r and for all possible $\boldsymbol{Q}[r]$ we have

$$\frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]} \geq \frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha^*[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha^*[r])|\boldsymbol{Q}[r]\right]} \tag{29}$$

[2]Equality (28) uses causality and the i.i.d. nature of the arrival process. It is formally proven by conditioning on $T(\alpha[r])$ and using iterated expectations.

where $\alpha^*[r]$ is any other (possibly randomized) coding action that could be chosen over the options in the set $\mathcal{A}$. This can be shown as follows: suppose we want to choose $\alpha[r] \in \mathcal{A}$ via a possibly randomized decision, to maximize the ratio of expectations in the left-hand side of (29). Such a decision would satisfy (29) by definition, since it would maximize the ratio of expectations over all alternative policies $\alpha^*[r]$. However, it is known that such a maximum is achieved via a *pure policy* that chooses a particular $\alpha \in \mathcal{A}$ with probability 1 (see [18, Ch. 7]). The best pure policy is thus the one that observes the queue backlogs $\boldsymbol{Q}[r]$ and chooses $\alpha[r] \in \mathcal{A}$ to maximize the deterministic ratio, which is exactly how Algorithm 2 chooses its action [see (8)].

Thus, (29) holds. We can rewrite (29) as follows:

$$\frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]} \geq \sum_{m=1}^{M} Q_m[r]\frac{\mathbb{E}\left[\mu_m(\alpha^*[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha^*[r])|\boldsymbol{Q}[r]\right]}. \tag{30}$$

We can thus plug any alternative (possibly randomized) decision $\alpha^*[r]$ into the right-hand side of (30). Consider the randomized algorithm that independently selects $\alpha \in \mathcal{A}$ every frame, independent of queue backlogs, according to the distribution $p(\alpha)$ in Theorem 2. Let $\alpha^*[r]$ represent the randomized decision under this policy. Then, from (6) we have for all $m \in \{1,\ldots,M\}$:

$$\lambda_m \leq \frac{\mathbb{E}\left[\mu_m(\alpha^*[r])\right]}{\mathbb{E}\left[T(\alpha^*[r])\right]} = \frac{\mathbb{E}\left[\mu_m(\alpha^*[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha^*[r])|\boldsymbol{Q}[r]\right]} \tag{31}$$

where the last equality holds because $\alpha^*[r]$ is chosen independently of $\boldsymbol{Q}[r]$. Using this in (30) yields

$$\frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]} \geq \sum_{m=1}^{M} Q_m[r]\lambda_m.$$

Rearranging terms above yields

$$\sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha[r]) - \mu_m(\alpha[r])|\boldsymbol{Q}[r]\right] \leq 0. \tag{32}$$

Plugging (32) into the drift bound of Lemma 6 yields

$$\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right] \leq B.$$

Taking expectations and using the definition of $\Delta[r]$ yields:

$$\mathbb{E}\left[L[r+1]\right] - \mathbb{E}\left[L[r]\right] \leq B \qquad \forall r \in \{0,1,2,\ldots\}.$$

Summing the above over $r \in \{0,1,\ldots,R-1\}$ yields

$$\mathbb{E}\left[L[R]\right] - \mathbb{E}\left[L[0]\right] \leq BR$$

and hence for all $R > 0$:

$$\sum_{m=1}^{M}\mathbb{E}\left[Q_m[R]^2\right] \leq 2\mathbb{E}\left[L[0]\right] + 2BR.$$

Thus, the second moments of all queues grow at most linearly, from which we guarantee rate stability by Lemma 5. □

We now prove that Algorithm 1 yields rate stability.

Proof (Theorem 3—Stability Under Algorithm 1): Note that Algorithm 1 is designed to observe queue backlogs $\boldsymbol{Q}[r]$ every frame r, and take a control action $\alpha[r] \in \mathcal{A}$ to minimize the right-hand side of the drift bound in Lemma 6. Therefore, we have

$$\begin{aligned}&\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right]\\ &\quad\le B + \sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r])|\boldsymbol{Q}[r]\right]\end{aligned}$$

where $\alpha^*[r]$ is any other (possibly randomized) decision. If $\alpha^*[r]$ makes a decision independent of $\boldsymbol{Q}[r]$ we have

$$\begin{aligned}&\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right]\\ &\quad\le B + \sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r])\right]. \end{aligned}\tag{33}$$

Consider again randomized algorithm $\alpha^*[r]$ that independently and randomly selects an action in $\mathcal{A}$ every frame, independent of queue backlogs, according to the distribution $p(\alpha)$ in Theorem 2. Then (31) again holds, so that for all $m \in \{1, \ldots, M\}$:

$$\mathbb{E}\left[\lambda_m T(\alpha^*[r]) - \mu_m(\alpha^*[r])\right] \le 0.$$

Substituting the above into the right-hand side of (33) gives

$$\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right] \le B$$

from which we then obtain rate stability in the same way as in the proof for Algorithm 2. □

Appendix D
Proof of the Queue Size Bound

Here we show that if $\boldsymbol{\lambda} \in \rho\Lambda_{\mathcal{A}}$, where $0 \le \rho < 1$, then both Algorithms 1 and 2 yield finite average backlog of size $O(1/(1-\rho))$.

Proof (Queue Bound for Algorithm 1): Because $\boldsymbol{\lambda} \in \rho\Lambda_{\mathcal{A}}$, we have

$$(\lambda_m/\rho) \in \Lambda_{\mathcal{A}}.$$

Thus, from Theorem 2 there is a randomized algorithm $\alpha^*[r]$ that makes decisions independent of queue backlogs to yield the following for all $m \in \{1, \ldots, M\}$:

$$\frac{\lambda_m}{\rho} \le \frac{\mathbb{E}\left[\mu_m(\alpha^*[r])\right]}{\mathbb{E}\left[T(\alpha^*[r])\right]}. \tag{34}$$

Define $T^* \triangleq \mathbb{E}\left[T(\alpha^*[r])\right]$. Using this and rearranging the above gives

$$\mathbb{E}\left[\mu_m(\alpha^*[r])\right] \ge \lambda_m T^*/\rho \qquad \forall m \in \{1, \ldots, M\}.$$

Substituting the above into (33) gives

$$\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right] \le B + \sum_{m=1}^{M} Q_m[r] T^* \lambda_m (1 - 1/\rho).$$

Taking expectations and using the definition of $\Delta[r]$ gives

$$\mathbb{E}\left[L[r+1]\right] - \mathbb{E}\left[L[r]\right] \le B + \sum_{m=1}^{M} \mathbb{E}\left[Q_m[r]\right] T^* \lambda_m (1 - 1/\rho).$$

Summing over $r \in \{0, \ldots, R-1\}$ (for any integer $R > 0$) gives

$$\mathbb{E}\left[L[R]\right] - \mathbb{E}\left[L[0]\right] \le BR + \sum_{r=0}^{R-1}\sum_{m=1}^{M} \mathbb{E}\left[Q_m[r]\right] T^* \lambda_m (1 - 1/\rho).$$

Using the fact that $\mathbb{E}\left[L[R]\right] \ge 0$ and $\mathbb{E}\left[L[0]\right] = 0$, dividing by R, and rearranging terms gives

$$\frac{1}{R}\sum_{r=0}^{R-1}\sum_{m=1}^{M} \lambda_m \mathbb{E}\left[Q_m[r]\right] \le \frac{B\rho}{T^*(1-\rho)}.$$

Because $T^* \ge 1$, the above bound can be simplified to $B\rho/(1-\rho)$. The above holds for all R, and so the expected queue backlog is $O(1/(1-\rho))$. Further, from [21] we can derive that the following holds with probability 1:

$$\limsup_{R\to\infty} \frac{1}{R}\sum_{r=0}^{R-1}\sum_{m=1}^{M} \lambda_m Q_m[r] \le \frac{B\rho}{T^*(1-\rho)}.$$

□

1) Proof (Queue Bound for Algorithm 2): Recall that (30) holds for every frame r and all possible $\boldsymbol{Q}[r]$ for Algorithm 2. Using $\alpha^*[r]$ as an algorithm that makes randomized decisions on frame r that are independent of $\boldsymbol{Q}[r]$ gives

$$\begin{aligned}&\frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]} \ge\\ &\qquad \frac{\sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\mu_m(\alpha^*[r])\right]}{\mathbb{E}\left[T(\alpha^*[r])\right]}\end{aligned}\tag{35}$$

where we have removed the conditional expectations on the right-hand side. Because $(\lambda_m/\rho) \in \Lambda_{\mathcal{A}}$, we know that there is an algorithm that makes independent and randomized decisions to yield (34). Plugging (34) into the right-hand side of (35) gives

$$\frac{\mathbb{E}\left[\sum_{m=1}^{M} Q_m[r]\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right]}{\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]} \ge \sum_{m=1}^{M} Q_m[r]\lambda_m/\rho.$$

Rearranging gives

$$\begin{aligned}&\sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\mu_m(\alpha[r])|\boldsymbol{Q}[r]\right] \ge\\ &\frac{1}{\rho}\sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha[r])|\boldsymbol{Q}[r]\right].\end{aligned}$$

Using this in the drift bound of Lemma 6 gives

$$\begin{aligned}&\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right]\\ &\quad\le B + \sum_{m=1}^{M} Q_m[r]\mathbb{E}\left[\lambda_m T(\alpha[r]) - \frac{\lambda_m}{\rho}T(\alpha[r])|\boldsymbol{Q}[r]\right].\end{aligned}$$

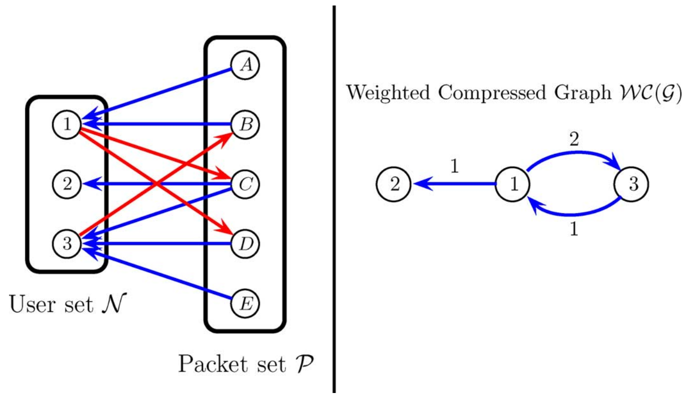

Fig. 10. Example graph $\mathcal{G}$ and its weighted compressed graph $\mathcal{WC}(\mathcal{G})$.

That is,

$$\begin{aligned}&\mathbb{E}\left[\Delta[r]|\boldsymbol{Q}[r]\right]\\ &\leq B+\sum_{m=1}^{M}Q_m[r]\lambda_m(1-1/\rho)\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right]\\ &\leq B+\sum_{m=1}^{M}Q_m[r]\lambda_m(1-1/\rho)\end{aligned}$$

where we have used the fact that $\mathbb{E}\left[T(\alpha[r])|\boldsymbol{Q}[r]\right] \geq 1$, and $1-1/\rho \leq 0$. We thus have by the same argument as in the previous proof that for any $R>0$:

$$\frac{1}{R}\sum_{r=0}^{R-1}\sum_{m=1}^{M}\lambda_m\mathbb{E}\left[Q_m[r]\right] \leq \frac{\rho B}{1-\rho}$$

and with probability 1:

$$\limsup_{R\to\infty}\frac{1}{R}\sum_{r=0}^{R-1}\sum_{m=1}^{M}\lambda_m Q_m[r] \leq \frac{\rho B}{1-\rho}.$$

□

Appendix E
General Compressed Graphs

The weighted compressed graph $\mathcal{WC}(\mathcal{G})$ was introduced in Section IV for the special case of broadcast relay networks. Such a graph is also useful for general index coding problems. Consider any directed bipartite demand graph $\mathcal{G}$ with user nodes $\mathcal{N}$ and packet nodes $\mathcal{P}$. Define $\mathcal{WC}(\mathcal{G})$ as a weighted graph on user nodes $\mathcal{N}$ with link weights P_{ij} defined as the (integer) number of distinct packets that user i has as side information that are wanted by user j. We say the graph has a link (i,j) if and only if $P_{ij}>0$. This general definition of $\mathcal{WC}(\mathcal{G})$ is consistent with the definition given for broadcast relay graphs in Section IV. An example of a graph $\mathcal{G}$ and its weighted compressed graph $\mathcal{WC}(\mathcal{G})$ is given in Fig. 10.

In the special case when $\mathcal{G}$ is the graph of a broadcast relay problem, $\mathcal{G}$ and $\mathcal{WC}(\mathcal{G})$ contain the same information. This is not true for general demand graphs $\mathcal{G}$. Indeed, the weight P_{ij} in the graph $\mathcal{WC}(\mathcal{G})$ tells us that node i has P_{ij} distinct packets that are wanted by node j, but does not specify which packets these are, or if these packets also take part in the weight count on other links of $\mathcal{WC}(\mathcal{G})$. In the example of Fig. 10, packet C affects the link weight for both links $(1,3)$ and $(1,2)$. Also, unlike broadcast relay problems, the sum of the weights of $\mathcal{WC}(\mathcal{G})$ is not necessarily equal to the number of packets P in $\mathcal{G}$.

Lemma 7: The original demand graph $\mathcal{G}$ is acyclic if and only if $\mathcal{WC}(\mathcal{G})$ is acyclic.

Proof: This is simple and omitted for brevity. □

Suppose that $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles, and let C be the number of such cycles. For each cycle $c \in \{1,\dots,C\}$, let $w_{min}^{(c)}$ represent the weight of the minimum weight link within the set of links in the cycle.

Theorem 5: Let $\mathcal{G}$ be a demand graph with N nodes and P packets. Suppose that $\mathcal{WC}(\mathcal{G})$ has edge-disjoint cycles, that there are C such cycles, and that all packets of these cycles are distinct and are unicast packets. Then,

$$T_{min}(\mathcal{G}) = P - \sum_{c=1}^{C} w_{min}^{(c)}. \tag{36}$$

Furthermore, the minimum clearance time can be achieved by performing cyclic coding $w_{min}^{(c)}$ times for each cycle $c \in \{1,\dots,C\}$, and then transmitting all the remaining packets without coding.

Proof: For each cycle $c \in \{1,\dots,C\}$, select a link with a link weight equal to $w_{min}^{(c)}$ (breaking ties arbitrarily). Let $\mathcal{P}^{(c)}$ be the set of all packets associated with this link. All packets in $\cup_{c=1}^{C}\mathcal{P}^{(c)}$ are distinct (by assumption), and the total number of these packets is

$$|\cup_{c=1}^{C}\mathcal{P}^{(c)}| = \sum_{c=1}^{C} w_{min}^{(c)}.$$

Now consider the subgraph $\mathcal{G}'$ formed from $\mathcal{G}$ by removing all packet nodes in the set $\cup_{c=1}^{C}\mathcal{P}^{(c)}$. The number of packets P' in this graph is

$$P' = P - \sum_{c=1}^{C} w_{min}^{(c)}.$$

Further, we have $T_{min}(\mathcal{G}') \leq T_{min}(\mathcal{G})$. Note that $\mathcal{WC}(\mathcal{G}')$ is formed from $\mathcal{WC}(\mathcal{G})$ by removing the min-weight link on each of the C cycles. Thus, $\mathcal{WC}(\mathcal{G}')$ is acyclic, so that $\mathcal{G}'$ is acyclic, and so $T_{min}(\mathcal{G}') = P'$ by Theorem 1. It follows that

$$T_{min}(\mathcal{G}) \geq P'.$$

Thus, any algorithm for clearing all packets in the graph $\mathcal{G}$ must use at least P' slots. However, a clearance time of P' can be achieved by cyclic coding over each of the C edge-disjoint cycles (noting that these packets are all distinct and unicast) and using direct transmission for the remaining packets. □

Corollary 1: Suppose all cycles of $\mathcal{WC}(\mathcal{G})$ are vertex-disjoint and involve only unicast packets. Then, $T_{min}(\mathcal{G})$ satisfies (36).

Proof: Vertex-disjoint implies edge-disjoint. Also, vertex-disjoint cycles that involve only unicast packets must involve distinct packets. The result then follows from Theorem 5. □

Corollary 2: If the demand graph $\mathcal{G}$ has P packets but only two users (so that $N = 2$), then

$$T_{min}(\mathcal{G}) = P - \min[P_{12}, P_{21}].$$

Proof: The graph $\mathcal{WC}(\mathcal{G})$ contains only two users and hence has at most one cycle, which is trivially edge-disjoint and involves distinct packets. Further, the cycle (if there is one) can only involve unicast packets. This is because any packet that participates in the cycle must be contained as side information at one of the two nodes, and hence is only desired by the single remaining node. The result then follows by Theorem 5. □

References

[1] M. J. Neely, A. S. Tehrani, and Z. Zhang, "Dynamic index coding for wireless broadcast networks," in *Proc. IEEE INFOCOM*, , CA, Mar. 2012, pp. 316–324.

[2] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," presented at the Conf. Inf. Sci. Syst., Mar. 2005.

[3] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," presented at the 43rd Annu. Allerton Conf. Commun., Control, Comput., Oct. 2005.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proc. Conf. Appl., Technol., Archit. Protocols Comput. Commun.*, 2006, pp. 243–254.

[5] Y. Birk and T. Kol, "Informed-source coding-on-demand (ISCOD) over broadcast channels," in *Proc. IEEE 17th Annu. Joint Conf. Comput. Commun. Soc.*, 1998, pp. 1257–1264.

[6] Y. Birk and T. Kol, "Coding-on-demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2825–2830, Jun. 2006.

[7] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.

[8] S. El Rouayheb, M. A. R. Chaudhry, and A. Sprintson, "On the minimum number of transmissions in single-hop wireless coding problems," in *Proc. IEEE Inf. Theory Workshop*, 2007, pp. 120–125.

[9] S. El Rouayheb, A. Sprintson, and C. Georghiades, "On the index coding problem and its relation to network coding and matroid theory," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3187–3195, Jul. 2010.

[10] M. Langberg and A. Sprintson, "On the hardness of approximating the network coding capacity," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1008–1014, Feb. 2011.

[11] N. Alon, E. Lubetzky, U. Stav, A. Weinstein, and A. Hasidim, "Broadcasting with side information," in *Proc. IEEE 49th Annu. Symp. Found. Comput. Sci.*, Oct. 2008, pp. 823–832.

[12] E. Lubetzky and U. Stav, "Nonlinear index coding outperforming the linear optimum," *IEEE Trans. Inf. Theory*, vol. 55, no. 8, pp. 3544–3551, Aug. 2009.

[13] M. A. R. Chaudhry and A. Sprintson, "Efficient algorithms for index coding," in *Proc. IEEE INFOCOM Workshops*, Apr. 2008, pp. 1–4.

[14] I. S. Reed and X. Chen, *Error-Control Coding for Data Networks*. Norwell, MA, USA: Kluwer, 2001.

[15] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," presented at the Inf. Theory Appl. Workshop, La Jolla, CA, USA, Feb. 2006.

[16] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[17] A. Saber Tehrani, A. G. Dimakis, and M. J. Neely, "Bipartite index coding," in *Proc. IEEE Int. Symp. Inf. Theory*, 2012, pp. 2246–2250.

[18] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. CA, USA: Morgan & Claypool, 2010.

[19] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.

[20] B. S. Vineeth and U. Mukherji, "Average-delay optimal policies for the point-to-point channel," in *Proc. 7th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw.*, Jun. 2009, pp. 1–10.

[21] M. J. Neely, "Stability and probability 1 convergence for queueing networks via Lyapunov optimization," *J. Appl. Math.*, vol. 2012, p. 831909, 2012.

Michael J. Neely received B.S. degrees in both Electrical Engineering and Mathematics from the University of Maryland, College Park, in 1997. He was then awarded a 3 year Department of Defense NDSEG Fellowship for graduate study at the Massachusetts Institute of Technology, where he received an M.S. degree in 1999 and a Ph.D. in 2003, both in Electrical Engineering. He joined the faculty of Electrical Engineering at the University of Southern California in 2004, where he is currently an Associate Professor. His research interests are in the areas of stochastic network optimization and queuing theory, with applications to wireless networks, mobile ad-hoc networks, and switching systems. Michael received the NSF Career award in 2008, the Viterbi School of Engineering Junior Research Award in 2009, and the Okawa Foundation Research Grant Award in 2012. He is a member of Tau Beta Pi and Phi Beta Kappa.

Arash Saber Tehrani received the B.S. degree in electrical engineering from Amirkabir University of Technology, Tehran, Iran, in 2001 and the M.Sc. degree in communication systems from the Technical University of Munich (TUM), Munich, Germany, in 2008. He is currently pursuing the Ph.D. degree in the Electrical Engineering at the University of Southern California, Los Angeles. His research interests include index coding, compressed sensing, and network optimization. Mr. Tehrani received the Viterbi School of Engineering Doctoral Fellowship award in 2009.

Zhen Zhang received the B.S. and M.S. degrees in mathematics from Nankai University, Tianjin, China, in 1969 and 1980, respectively, the Ph.D. degree in applied mathematics from Cornell University, Ithaca, NY, in 1984, and the Habilitation in mathematics from Bielefeld University, Bielefeld, Germany, in 1988. He served as a Lecturer, Department of Mathematics, Nankai University, from 1981 to 1982. He was a Postdoctoral Research Associate with the School of Electrical Engineering, Cornell University, from 1984 to 1985 and with the Information Systems Laboratory, Stanford University, in fall 1985. From 1986 to 1988, he was with the Mathematics Department, Bielefeld University, Bielefeld, Germany. He joined the faculty of University of Southern California, Los Angeles, in 1988, where he is currently a Professor of Electrical Engineering with the Department of Electrical Engineering-Systems. He has been a Visiting Professor with the Chinese University of Hong Kong in 1995, Shanghai Jiaotong University in 2002, University of Waterloo, Waterloo, Canada in 2008, and Nankai University in 2009. He holds an Adjunct Professor position with Shanghai Jiaotong University. His research interests include information theory, coding theory, data compression, network theory, communication theory, and applied mathematics. His current research is focused on network coding theory.