Multicast Packing for Coding across Multiple Unicasts

Chun Meng
INC, CUHK & EECS, UC Irvine
cmeng1@uci.edu

Hulya Seferoglu
LIDS, MIT
hseferog@mit.edu

Athina Markopoulou
EECS, UC Irvine
athina@uci.edu

Kenneth W. Shum, Chung Chan
INC, CUHK
{wkshum,cchan}@inc.cuhk.edu.hk

***Abstract*—We consider the problem of inter-session network coding for multiple unicast flows over directed acyclic graphs. Our approach consists of the following steps: (i) The unicast flows are partitioned into multiple disjoint subsets of unicast flows; (ii) each subset of unicast flows is mapped to a multicast flow, and linear network codes are constructed for these multicast flows. These linear network codes collectively serve as a linear network code for the original multiple unicast flows, which we refer to as *multicast-packing code (MPC)*. We formulate a linear program to evaluate the performance of MPC for a given partition of the unicast flows. We also propose a practical algorithm, using simulated annealing, for finding such a partition. Using simulations, we demonstrate the benefits of the multicast packing code in terms of achievable common rate and cost, as well as the efficiency of the partitioning algorithm in terms of running time.**

I. Introduction

Network coding was originally introduced to maximize the rate of a multicast flow over a network [1], [2]. Network code design for this scenario has been solved using polynomial-time deterministic algorithms [3] or a random approach [4]. Yet, network coding across different sessions, which includes multiple unicasts as a special case, remains a well-known open problem. It has been shown that even determining whether there exists a linear network coding solution to such a problem is NP-hard [6]. Thus, constructive and sub-optimal approaches, such as [7]–[12], have been proposed and shown to improve over routing for multiple unicasts.

In this paper, we introduce a constructive inter-session network coding scheme for multiple unicast flows, illustrated in the following example.

Example 1. Let us consider the network $\mathcal{N}$ shown in Fig. 1a. In this network, five unicast flows coexist in the network, where each edge has unit capacity. The ith unicast flow ($1 \leq i \leq 5$) is denoted by $\omega_i = (s_i, d_i)$, where s_i and d_i are the sender and the receiver of ω_i, respectively. The set of unicast flows are represented by $\Omega = \{\omega_i : 1 \leq i \leq 5\}$.

Let us partition Ω into two disjoint sets, $\Omega_1 = \{\omega_1, \omega_2, \omega_3\}$ and $\Omega_2 = \{\omega_4, \omega_5\}$. Also, $\mathcal{N}$ is partitioned into two sub-graphs $\mathcal{N}_1$ and $\mathcal{N}_2$. The flows in Ω_1 and Ω_2 are only transmitted over their respective subgraphs; *i.e.,* flows in Ω_1 (Ω_2) are transmitted only over $\mathcal{N}_1$ ($\mathcal{N}_2$). Then, we construct our codes to be the network codes for two multicast scenarios: In $\mathcal{N}_1$, d_1, d_2 and d_3 can decode all the source symbols transmitted by s_1, s_2 and s_3; in $\mathcal{N}_2$, d_4 and d_5 can decode all the source symbols transmitted by s_4 and s_5. These network codes also serve as network codes for the original multiple unicast flows.

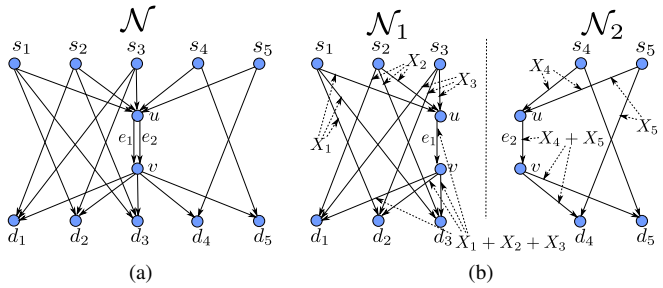

Fig. 1. A motivating example. In the network $\mathcal{N}$ shown in (a), five unicast flows coexist. In (b), these unicast flows are partitioned into two subsets, $\Omega_1 = \{\omega_1, \omega_2, \omega_3\}$ and $\Omega_2 = \{\omega_4, \omega_5\}$, and the network $\mathcal{N}$ is partitioned into two sub-graphs $\mathcal{N}_1$ and $\mathcal{N}_2$. Note that the unicast flows in Ω_1 are transmitted only over $\mathcal{N}_1$, while the unicast flows in Ω_2 are transmitted only over $\mathcal{N}_2$. Then, we construct linear network codes for Ω_1 and Ω_2 separately, as shown in (b), where X_i ($1 \leq i \leq 5$) denotes the source symbol transmitted by s_i. Note that the constructed network codes are network codes for two multicast scenarios over $\mathcal{N}_1$ and $\mathcal{N}_2$.

Note that several partitions of Ω, other than Ω_1 and Ω_2 discussed in this example, are also possible. Part of the contribution of this paper is how to find good partitions. ■

The example demonstrates the approach we follow in this paper. First, we partition the set of unicast flows into disjoint subsets of unicast flows. Second, we map each subset of unicast flows to a multicast flow with the same set of receivers, and linear network codes are constructed for these multicast flows by a deterministic [3] or a random approach [4]. These linear network codes collectively serve as a network code for the original unicast flows, which we refer to as *Multicast-Packing Code (MPC)*.

MPC has the following strengths. First, the MPC approach, *i.e.,* partitioning the unicast flows to subsets of unicast flows and mapping each subset to a multicast network coding problem, is general enough to be applied to any directed acyclic graph. Second, given a partition of the set of the unicast flows, we use a linear program to quickly analyze the performance, *e.g.,* maximum common rate and minimum cost, achieved by MPC. In contrast, previous constructive approaches are difficult to analyze due to the lack of succinct mathematical formulations. For example, the integer linear programming (ILP) approach [7] is difficult to analyze since it needs to consider all possible butterfly structures in the network. On the other hand, the evolutionary approach [8], does not have a mathematical formulation. Third, in order to find the best MPC, we only need to search the space of all partitions of the set of unicast flows, *independently of the network size.* This is clearly more efficient and scalable than other constructive

C. Meng and A. Markopoulou are also affiliated with the Center for Pervasive Communications and Computing at UC Irvine. C. Meng was visiting the Institute of Network Coding, in the Chinese University of Hong Kong, when this work was conducted.

This work has been partially supported by the following grants: NSF CAREER Award 0747110, AFOSR MURI Award FA9550-09-0643, AFOSR Award FA9550-10-1-0310, and the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

approaches, whose combinatorial optimization involved the network graph in addition to the set of flows. For example, the approach in [7] uses integer linear programming to select the best set of butterflies considering all pairs of flows, but also all possible coding points on the network topology. The evolutionary approach [8] involves a random walk in the space of local coding vectors, which does not scales well with the network size. Although independent of the network size, our search problem is still exponential in the number of unicast flows. This is why we utilize a suboptimal, yet efficient, simulating annealing technique to find good partitions of the unicast flows. Simulation results over appropriately chosen scenarios demonstrate the above points.

The structure of the rest of the paper is as follows. Section II presents related work. Section III describes the problem setup. Section IV presents the multicast packing code approach. Section V presents a practical flow partitioning algorithm using simulated annealing. Section VI presents simulation results. Section VII concludes the paper.

II. Related Work

The inter-session network coding problem across different flows, which includes multiple unicasts as a special case, is a well-known open problem. It has been shown that linear network codes may not be sufficient to achieve capacity region [5], and determining whether there exists a linear network coding solution is NP-hard [6]. Thus, constructive and sub-optimal approaches have been considered.

An integer linear programming (ILP) based approach is proposed in [7] for inter-session network coding across multiple unicasts. An evolutionary approach is proposed in [8] for the same problem. Both approaches have to face the scalability problem. For example, the ILP approach proposed in [7] needs to consider all possible butterflies in the network, the number of which increases exponentially with the network size; the evolutionary approach involves a random walk in the space of the local coding vectors in the network, which also scales up exponentially with the network size. Our main difference is that the search space in our problem is smaller: it depends only on the number of flows and not on the network size. However, the complexity is still exponential in the number of flows, and this is why we employ a suboptimal, yet efficient, simulated annealing-based partitioning algorithm.

Pairwise inter-session network coding is proposed in [9] and [10]. A game theoretic approach to inter-session network coding for two unicast sessions is considered in [11]. A practical one-hop inter-session network coding scheme is proposed in [12] over wireless networks. Our main difference is that our approach is general exploit network coding opportunities across more than two unicast flows and multi-hop network coding. An improved genetic algorithm for finding coding matrix for multi-user scenario is proposed in [13]. An interference alignment approach is considered in [14], [15], [16]. However, the interference alignment approach might be infeasible for some networks. Multicast packing were considered earlier, in [17], but this work doesn't consider network coding and its potential in improving rates.

III. Problem Set-up

We consider a weighted directed acyclic graph $\mathcal{N} = (V, E, h)$, where V and E denote the node set and the edge set, respectively. $h : E \to \mathbb{R}_{\geq 0}$ is a function such that for $e \in E$, $h(e)$ equals the capacity of e. We allow multiple edges between two nodes, and hence $E \subseteq V \times V \times \mathbb{Z}_+$, where the last integer enumerates edges between two nodes. The edges are denoted by (u, v, i). If no confusion arises, we simply use (u, v) to represent edges. The sets of incoming and outgoing edges at a node v are denoted by $\mathrm{In}(v)$ and $\mathrm{Out}(v)$, respectively. We denote the tail and the head of an edge e by $\mathrm{head}(e)$ and $\mathrm{tail}(e)$, respectively.

A unicast flow is represented by a tuple $\omega = (s, d) \in V \times V$, where s, d are the sender and receiver of ω, respectively. We consider a multiple-unicast scenario, represented by a set Ω of unicast flows that coexist in the network, *i.e.,* $\Omega = \{\omega_i = (s_i, d_i) : 1 \leq i \leq |\Omega|\}$.

The edge capacities can be any positive real number. In order to deal with possible non-integer capacities, we consider a time-extended version of the network as follows. We extend the length of a time slot by t times, where t is a positive integer. Hence, each edge $e \in E$ can transmit at most $k(e) = \lceil t \times h(e) \rceil$ symbols in an extended time slot. We use a network $\mathcal{N}^{(t)} = (V, E^{(t)}, h^{(t)})$ to represent this time-extended network: The node set of $\mathcal{N}^{(t)}$ is the same as $\mathcal{N}$; for each $e \in E$, we add $k(e)$ parallel edges from $\mathrm{tail}(e)$ to $\mathrm{head}(e)$ in $E^{(t)}$; each edge $e \in E^{(t)}$ has unit capacity, *i.e.,* $h^{(t)}(e) = 1$. We refer to $\mathcal{N}^{(t)}$ as the *t-extension network* of $\mathcal{N}$.

We define the following notations to facilitate our discussion. Given $\Omega' \subseteq \Omega$, $S(\Omega')$ and $D(\Omega')$ denote the set of senders and the set of receivers involved in Ω', respectively. For $(s_i, d_i) \in \Omega$, $\mathbf{X}(s_i)$ denotes the vector of source symbols that s_i transmits to d_i. For $v \in V$, $\mathbf{Z}^-(v)$ denotes the vector of the symbols transmitted along the incoming edges at v, and the source symbols injected by v if $v \in S(\Omega)$; $\mathbf{Z}^+(v)$ denotes the vector of the symbols transmitted along the outgoing edges at v. Given a vector $\mathbf{A}$, $|\mathbf{A}|$ denotes the dimension of $\mathbf{A}$.

We make the following assumptions to simplify our analysis. (i) The source symbols transmitted by all the senders in $S(\Omega)$ are mutually independent random variables; (ii) the symbols transmitted in the network take values from a finite field $\mathbb{F}_{2^m}$; (iii) each edge in $E^{(t)}$ represents an error-free and delay-free channel; (iv) the senders involved in Ω are all different, and so are the receivers involved in Ω.

We consider scalar linear network coding over $\mathcal{N}^{(t)}$. In this setup, at each node $v \in V$, $\mathbf{Z}^-(v)$ is mapped to $\mathbf{Z}^+(v)$ via a linear equation, $\mathbf{Z}^+(v) = \mathbf{Z}^-(v)\mathbf{E}(v)$, where $\mathbf{E}(v)$ is a $|\mathbf{Z}^-(v)| \times |\mathbf{Z}^+(v)|$ matrix on $\mathbb{F}_{2^m}$. For each $(s_i, d_i) \in \Omega$, $\mathbf{X}(s_i)$ can be decoded at d_i via a linear equation, $\mathbf{X}(s_i) = \mathbf{Z}^-(d_i)\mathbf{F}(d_i)$, where $\mathbf{F}(d_i)$ is a $|\mathbf{Z}^-(d_i)| \times |\mathbf{X}(s_i)|$ matrix on $\mathbb{F}_{2^m}$. $\mathbf{E}(v)$ and $\mathbf{F}(d_i)$ are referred to as the *encoding matrix* at v and the *decoding matrix* at d_i respectively. We group all the encoding matrices and decoding matrices into a set $\Phi^{(t)}$, and refer to it as a *scalar linear network code* for the multiple-unicast scenario Ω over $\mathcal{N}^{(t)}$.

The *transmission rate* of a unicast session (s_i, d_i) achieved by $\Phi^{(t)}$ is defined as $r(s_i) = \frac{|\mathbf{X}(s_i)|}{t}$. The *rate vector* achieved by $\Phi^{(t)}$ is defined as $\mathbf{r}(\Phi^{(t)}) = (r(s_i) : s_i \in S(\Omega))$. Given a vector $\mathbf{r}_0 \in \mathbb{R}_{\geq 0}^{|\Omega|}$, if there exists a sequence of scalar linear network codes, $\{\Phi^{(t_n)} : 1 \leq n < \infty\}$, where $\{t_n\}$ is a sequence of positive integers, and $\Phi^{(t_n)}$ is a scalar linear network code over the t_n-extension network $\mathcal{N}^{(t_n)}$, such that $\lim_{n\to\infty} \mathbf{r}(\Phi^{(t_n)}) = \mathbf{r}_0$, we say that $\mathbf{r}_0$ is *linearly achievable*.

IV. Packing Multicasts for Multiple Unicasts

A. Multicast-Packing Code

In this section, we present the idea of MPC, *i.e.,* mapping multiple unicast flows to multicast flows when the partitions of the original multiple unicast flows are given. The partitioning problem is considered in Section V. We introduce the following notations to facilitate our discussion. We use $\gamma = (s, D)$, where $s \in V$ and $D \subseteq V$, to represent a multicast flow such that the nodes in D all require the source symbols transmitted by s. A multicast scenario is represented by a set of multicast flows, *i.e.,* $\Gamma = \{(s_i, D) : 1 \leq i \leq |\Gamma|\}$, where the nodes in D require all the source symbols transmitted by all s_i's.

A *partition* of the multiple-unicast scenario Ω is a set of non-empty disjoint subsets of Ω, $\mathcal{G} = \{\Omega_i : 1 \leq i \leq |\mathcal{G}|\}$, such that $\Omega = \bigcup_{i=1}^{|\mathcal{G}|} \Omega_i$. Given a partition $\mathcal{G}$, an *allocation of network capacities* is represented by a set of functions $\mathcal{H} = \{h_i : E \to \mathbb{R}_{\geq 0} : 1 \leq i \leq |\mathcal{G}|\}$, which satisfies the following condition:

$$\sum_{i=1}^{|\mathcal{G}|} h_i(e) \leq h(e) \quad \forall e \in E. \tag{1}$$

Given $\mathcal{G}$ and $\mathcal{H}$, we can view each Ω_i as a multiple-unicast scenario of smaller scale that works "separately" in a network $\mathcal{N}_i = (V, E, h_i)$. For example, in Fig. 1, the allocation of network capacities are as follows. If e is an outgoing edge of s_1, s_2, s_3, an incoming edge of d_1, d_2, d_3, or $e = e_1$, $h_1(e) = 1$; otherwise, $h_1(e) = 0$. If e is an outgoing edge of s_4, s_5, an incoming edge of d_4, d_5, or $e = e_2$, $h_2(e) = 1$; otherwise, $h_2(e) = 0$.

Suppose $\mathcal{G}$ and $\mathcal{H}$ are already given. We construct a scalar linear network code for Ω over $\mathcal{N}^{(t)}$ as follows. For each $\Omega_i \in \mathcal{G}$, we construct a multicast scenario, $\Gamma_i = \{(s_j, D(\Omega_i)) : s_j \in S(\Omega_i)\}$, over the t-extension network $\mathcal{N}_i^{(t)} = (V, E_i^{(t)}, h_i^{(t)})$ of $\mathcal{N}_i$, where the receivers in $D(\Omega_i)$ can decode the source symbols transmitted by all the senders in $S(\Omega_i)$. A scalar linear network code $\Phi_i^{(t)}$ can then be constructed for this multicast scenario. Let $\mathbf{E}_i(v)$ and $\mathbf{F}_i(d_j)$ denote the encoding matrices and decoding matrices in $\Phi_i^{(t)}$, respectively. We then combine the above linear network codes $\Phi_i^{(t)}$ $(1 \leq i \leq k)$ into a scalar linear network code $\Phi^{(t)}$ for Ω over $\mathcal{N}^{(t)}$ as follows: Each encoding matrix $\mathbf{E}(v)$ in $\Phi^{(t)}$ is simply a concatenation of the encoding matrices $\mathbf{E}_i(v)$'s; however, since each receiver $d_j \in D(\Omega_i)$ only needs $\mathbf{X}(s_j)$, the decoding matrix $\mathbf{F}(d_j)$ in $\Phi^{(t)}$ is a sub-matrix of $\mathbf{F}_i(d_j)$ consisting of the columns corresponding to $\mathbf{X}(s_j)$.

We refer to the above scalar linear network code $\Phi^{(t)}$ as a *multicast-packing code* (MPC) for Ω over $\mathcal{N}^{(t)}$ with respect to $(\mathcal{G}, \mathcal{H})$. Given $\mathbf{r}_0 \in \mathbb{R}_{\geq 0}^{|\Omega|}$, if there exists a sequence of MPCs with respect to $(\mathcal{G}, \mathcal{H})$, $\{\Phi^{(t_n)} : 1 \leq n < \infty\}$, such that $\lim_{n\to\infty} \mathbf{r}(\Phi^{(t_n)}) = \mathbf{r}_0$, we say that $\mathbf{r}_0$ is *achievable through MPC* with respect to $(\mathcal{G}, \mathcal{H})$.

Remark: In the construction of MPC, we can use the following method. We add a super sender s and connect it to each $s_j \in S(\Omega_i)$ via $|\mathbf{X}(s_j)|$ parallel edges, each of which has unit capacity and carries a distinct source symbol in $\mathbf{X}(s_j)$. Thus, we transform the multicast scenario with multiple multicast flows into a multicast scenario with a single multicast flow. Hence, we can employ the random approach of [4] or the deterministic approach of [3] to construct linear network code for this multicast scenario.

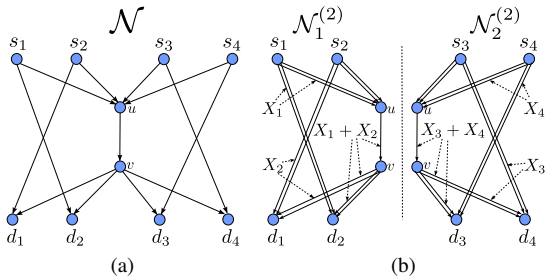

Fig. 2. An example of multicast-packing code over a 2-extension network. The network is shown in (a), where each edge has unit capacity, and 4 unicast flows coexist in the network. In (b), we show an MPC over a 2-extension network $\mathcal{N}^{(2)}$, which achieves one half rate for each unicast flow.

Example 2. Consider an example network as shown in Fig. 2. In this example, each edge has unit capacity, and four unicast flows coexist in the network, *i.e.,* $\Omega = \{\omega_i = (s_i, d_i) : 1 \leq i \leq 4\}$. We consider a partition $\mathcal{G} = \{\Omega_1 = \{\omega_1, \omega_2\}, \Omega_2 = \{\omega_3, \omega_4\}\}$. The allocation of network capacities is as follows. If e is an outgoing edge of s_1, s_2 or an incoming edge of d_1, d_2, $h_1(e) = 1$; if $e = (u, v)$, $h_1(e) = 0.5$; otherwise, $h_1(e) = 0$. If e is an outgoing edge of s_3, s_4 or an incoming edge of d_3, d_4, $h_2(e) = 1$; if $e = (u, v)$, $h_2(e) = 0.5$; otherwise, $h_2(e) = 0$. An MPC with respect to $(\mathcal{G}, \mathcal{H})$ over the 2-extension network $\mathcal{N}^{(2)}$ is shown in Fig. 2b. Clearly, this MPC achieves one half rate for each unicast flow. ■

Proposition 1. The multicast-packing codes as shown in Fig. 1b and Fig. 2b achieve the maximal common rate, which is the minimum rate each unicast flow must achieve simultaneously.

Proof. Technical report [19]. □

The choice of $\mathcal{G}$ and $\mathcal{H}$ is subject to various practical goals, which we explain in detail in Section IV-C.

B. Achievability of Multicast-Packing Code

In this section, we present the achievability properties of MPC for a given partition of the unicast flows. We first introduce the following concept. Given $S \subset V$ and $d \in V - S$, an $S - d$ flow over $\mathcal{N}$ is a function $f : E \to \mathbb{R}_{\geq 0}$ which satisfies the following conditions:

1) For each edge $e \in E$, $0 \leq f(e) \leq h(e)$.
2) For each node $v \in V - (S \cup \{d\})$, the following flow conservation law must be satisfied:

$$\sum_{e \in \mathrm{In}(v)} f(e) = \sum_{e \in \mathrm{Out}(v)} f(e).$$

The value of flow f at $v \in S$ is defined as $\mathrm{val}(f, v) = \sum_{e \in \mathrm{Out}(v)} f(e) - \sum_{e \in \mathrm{In}(v)} f(e)$.

The following theorem fully characterizes the rate region achieved by MPCs with respect to $(\mathcal{G}, \mathcal{H})$.

Theorem 1. Assume the size of finite field $\mathbb{F}_{2^m}$ is greater than $|\Omega|$. Given a vector $\mathbf{r} = (r(s_i) : s_i \in S(\Omega)) \in \mathbb{R}_{\geq 0}^{|\Omega|}$, $\mathbf{r}$ is achievable through MPC with respect to $(\mathcal{G}, \mathcal{H})$ if and only if for each $\Omega_i \in \mathcal{G}$ and each $d_j \in D(\Omega_i)$, there exists a $S(\Omega_i) - d_j$ flow f_{ij} over $\mathcal{N}_i = (V, E, h_i)$ such that $\mathrm{val}(f_{ij}, s_l) = r(s_l)$ for each $s_l \in S(\Omega_i)$.

Proof. Technical report [19]. □

Example 2 - continued. Let us consider again the example provided in Fig. 2a to explain Theorem 1. For $\Omega_i = \Omega_1$ and $d_j = d_1$, we construct a $\{s_1, s_2\} - d_1$ flow f_{11} over $\mathcal{N}_1 = (V, E, h_1)$ as follows: For $e \in \{(s_1, u), (u, v), (v, d_1), (s_2, d_1)\}$, $f_{11}(e) = 0.5$; otherwise, $f_{11}(e) = 0$. For $\Omega_i = \Omega_1$ and $d_j = d_2$, we construct a $\{s_1, s_2\} - d_2$ flow f_{12} as follows: For $e \in \{(s_2, u), (u, v), (v, d_2), (s_1, d_2)\}$, $f_{12}(e) = 0.5$; otherwise, $f_{12}(e) = 0$. It is easy to see that $\text{val}(f_{1i}, s_j) = 0.5$ for $i = 1, 2$ and $j = 1, 2$. Similarly, we can verify the case for $\Omega_i = \Omega_2$ and $s_j = d_3, d_4$. This indicates that the MPC can achieve a common rate $\frac{1}{2}$. ■

C. Linear Program for MPC

In this section, we formulate a linear program to calculate the performance achieved by MPCs for a given partition of the unicast flows. Theorem 1 yields a set of linear constraints to describe the rate region achieved by multicast-packing code for a given partition $\mathcal{G}$. In addition to Eq. (1), we add the following linear constraints:

- For each $\Omega_i \in \mathcal{G}$, $d_j \in D(\Omega_i)$ and $s_l \in S(\Omega_i)$, the value of the $S(\Omega_i) - d_j$ flow f_{ij} at s_l equals $r(s_l)$:

$$r(s_l) = \sum\nolimits_{e \in \text{Out}(s_l)} f_{ij}(e) - \sum\nolimits_{e \in \text{In}(s_l)} f_{ij}(e). \quad (2)$$

- For each $\Omega_i \in \mathcal{G}$ and $d_j \in D(\Omega_i)$, f_{ij} must satisfy the flow conservation law at each $v \in V - (S(\Omega_i) \cup \{d_j\})$:

$$\sum\nolimits_{e \in \text{Out}(v)} f_{ij}(e) = \sum\nolimits_{e \in \text{In}(v)} f_{ij}(e). \quad (3)$$

- For each $\Omega_i \in \mathcal{G}$, $d_j \in D(\Omega_i)$ and $e \in E$,

$$0 \leq f_{ij}(e) \leq h_i(e). \quad (4)$$

Remark: It can be easily seen that if each Ω_i only contains one unicast flow, the above linear constraints are reduced to those of a routing scheme, in which each node only forwards the symbols it receives. Hence, routing can be viewed as a special case of MPC.

In practice, the above constraints can be combined with additional constraints and various objectives to form a linear program. In this paper, we consider the following two objectives:

- *Maximum common rate*: We require that the transmission rate of each unicast flow must be at least a common rate $r(\mathcal{G})$. In addition to Eq. (1) and Eq. (2)-(4), we add the following linear constraint for each $s_l \in S(\Omega)$,

$$r(s_l) \geq r(\mathcal{G}). \quad (5)$$

The objective is simply:

$$\text{Maximize} \quad r(\mathcal{G}). \quad (6)$$

- *Minimum cost*: We require that the transmission rate of each unicast flow ω_l must be at least a fixed value q_l. In addition to Eq. (1) and Eq. (2)-(4), we add the following constraint for each $s_l \in S(\Omega)$,

$$r(s_l) \geq q_l. \quad (7)$$

Let $a : E \to \mathbb{R}_{\geq 0}$ be a function such that $a(e)$ denotes the cost of occupying unit capacity along e. The objective is simply:

$$\text{Minimize} \quad \sum\nolimits_{e \in E} \sum\nolimits_{i=1}^{|\Omega|} a(e) h_i(e). \quad (8)$$

Remark: As it is seen, the allocation of network capacities $\mathcal{H}$ are decision variables in the above linear programs (see Eq. (1)). Thus, the solution to these linear programs not only allows us to evaluate the performance achieved by multicast-packing code for a given partition $\mathcal{G}$, but also includes $\mathcal{H}$ as part of the LP solution. From practical perspective, we only need to find the best partition $\mathcal{G}$ such that the MPC constructed from the LP solution achieves the best performance among all MPCs for Ω. Yet, when $|\Omega|$ becomes large, finding such partition as an LP solution is computationally expensive. Therefore, we present a practical partitioning algorithm based on simulated annealing techniques in the next section.

V. Searching For Good Partitions

In this section, we present a practical partitioning algorithm to approximate the best partition of Ω by employing simulated annealing technique [18]. The running process of the algorithm is divided into stages, each of which is associated with a positive value T (also called *temperature* [18]). During each stage, it performs a random walk in the space of partitions of Ω. The probability that it moves from the current partition $\mathcal{G}$ to another partition $\mathcal{G}_1$ is

$$Pr(r, r_1, T) = \begin{cases} 1 & \text{if } r_1 \text{ is better than } r; \\ \exp(-\frac{\kappa |r_1 - r|}{T}) & \text{otherwise.} \end{cases}$$

where r, r_1 denote the values of the objective function corresponding to $\mathcal{G}$ and $\mathcal{G}_1$, respectively. At the end of each stage, we reduce T by a constant factor. Note that in case $\mathcal{G}_1$ is worse than $\mathcal{G}$, there is still probability that the algorithm will move to $\mathcal{G}_1$. This strategy prevents the algorithm from being stuck at a sub-optimal partition, which is typical of a greedy strategy.

The algorithm consists of the following parts:

Initialization (lines 1-4): The algorithm starts with a trivial partition $\mathcal{G}$, in which each Ω_i contains only one unicast flow ω_i (line 1). An LP solver is invoked to compute the solution $(r, \mathcal{H})$ to the linear program constructed from $\mathcal{G}$ (line 2), where r denotes the value of the objective function, and $\mathcal{H}$ the allocation of network capacities included in the solution. Then, T is initialized to T_0 (line 4).

The for-loop (lines 5-22): The major body of the algorithm is the for-loop. Each iteration of the for-loop corresponds to a stage. The major body of the for-loop is a while-loop (lines 7-19). At the beginning of the while-loop, the algorithm calls a function get to generate a new partition $\mathcal{G}_1$ from $\mathcal{G}$ (line 8). The LP solver is invoked to compute the solution of the linear program constructed from $\mathcal{G}$ (line 9). If r_1 is better than the best objective found previously, the algorithm records this better solution (lines 10-13). A function oracle is then called to decide if the algorithm moves to the new partition $\mathcal{G}_1$ (lines 14-17). At the end of each stage, T is reduced by a factor η (line 20).

In the function get, the algorithm first randomly picks up two distinct subsets Ω_i, Ω_j, and a unicast session $\omega_l \in \Omega_i$. Then, it moves ω_l from Ω_i to Ω_j, and returns the final

Algorithm 1: Algorithm to find good partition

$\mathcal{G} \leftarrow \{\{\omega_i\} : 1 \leq i \leq |\Omega|\}$; // Initialize partition
$(r, \mathcal{H}) \leftarrow \text{solve}(\mathcal{G})$; // Solve LP
$(r_{opt}, \mathcal{H}_{opt}) \leftarrow (r, \mathcal{H})$; $\mathcal{G}_{opt} \leftarrow \mathcal{G}$; // Store the result
$T \leftarrow T_0$; // Setting initial temperature
for $i \leftarrow 1$ **to** α **do**
  $j \leftarrow 1, k \leftarrow 1$;
  while $j \leq \beta$ **and** $k \leq \zeta$ **do**
    $\mathcal{G}_1 \leftarrow \text{get}(\mathcal{G})$; // Get a new partition from $\mathcal{G}$
    $(r_1, \mathcal{H}_1) \leftarrow \text{solve}(\mathcal{G})$; // Solve LP
    if r_1 is better than r_{opt} **then**
      $(r_{opt}, \mathcal{H}_{opt}) \leftarrow (r_1, \mathcal{H}_1)$;
      $\mathcal{G}_{opt} \leftarrow \mathcal{G}_1$;
    end
    if $\text{oracle}(r, r_1, T)$ = **true then**
      $r \leftarrow r_1, \mathcal{G} \leftarrow \mathcal{G}_1$; // Move to the new partition
      $k \leftarrow k + 1$; // Record successful moves
    end
    $j \leftarrow j + 1$;
  end
  $T \leftarrow T * \eta$; // Decrease temperature by a factor
end
return $(r_{opt}, \mathcal{H}_{opt}, \mathcal{G}_{opt})$;

function $\text{get}(\mathcal{G})$
  Select Ω_i randomly from $\mathcal{G}$; select ω_l randomly from Ω_i;
  Select Ω_j randomly from $\mathcal{G} \cup \{\emptyset\}$ such that $\Omega_i \neq \Omega_j$;
  $\Omega_i \leftarrow \Omega_i - \{\omega_l\}$, $\Omega_j \leftarrow \Omega_j \cup \{\omega_l\}$;
  if Ω_i is empty **then** $\mathcal{G} \leftarrow \mathcal{G} - \{\Omega_i\}$;
  return $\mathcal{G}$;

function $\text{oracle}(r, r_1, T)$
  if r_1 is better than r **then return true**;
  Randomly select a number δ in the range $[0, 1]$;
  if $\delta < \exp(-\frac{\kappa|r - r_1|}{T})$ **then return true**;
  else return false;

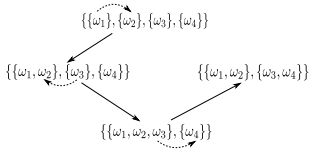

Fig. 3. An example of the running process of the algorithm. The dashed lines denote the operations performed by the get function. For example, for the initial partition, the get function moves ω_1 from $\{\omega_1\}$ to $\{\omega_2\}$, resulting in the partition $\{\{\omega_1, \omega_2\}, \{\omega_3\}, \{\omega_4\}\}$. The algorithm reaches the best partition $\{\{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}\}$ from the initial partition in three steps.

partition. Fig. 3 shows an example of the running process of the algorithm for the example of Fig. 2a. The algorithm starts from $\{\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4\}\}$, and reaches the best partition $\{\{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}\}$ in three steps.

To deal with large scenarios, we divide the space of partitions of Ω into disjoint sub-spaces, and assign each of them to a dedicated processor. Then, these processors run the algorithm in parallel by randomly moving in the assigned sub-spaces. At last, we choose the best partition returned by these processors.

VI. Evaluation

A. Simulation Setup

We evaluate the performance of our approach via simulations. We used a network (see Fig. 4), which has been used by other researchers [8] [7], for our simulations. It has been shown by previous work [8] [7] that network coding exhibits better performance than routing only when shared bottlenecks are present. Thus, in our simulations, we focus on communication scenarios, where senders are separated from receivers by bottleneck links, *e.g.,* e_5, e_6 and e_7, that

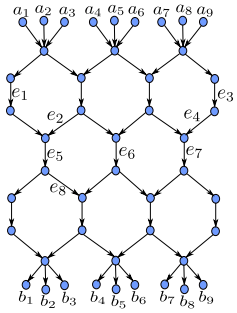

Fig. 4. The network used for simulation.

have lower bandwidths and higher costs than other links. By changing the positions of senders and receivers, this network allows us to investigate the influence of the positions of bottlenecks on the performance of MPCs. The outgoing edges of a_i's ($1 \leq i \leq 9$) and the incoming edges of b_i's all have infinite capacities and zero costs. Each multiple-unicast scenario Ω is a subset of $\{(a_i, b_j) : 1 \leq i, j \leq 9\}$ such that the senders and the receivers are all distinct. We considered the cases where $3 \leq |\Omega| \leq 7$. For each setting of $|\Omega|$, we randomly constructed 50 multiple-unicast scenarios. We considered two objectives, maximum common rate and minimum cost. For the first objective, we considered two capacity settings for the edges other than the outgoing edges of a_i's and the incoming edges of b_i's:

Scenario 1: The edges all have unit capacities.

Scenario 2: $h(e_2) = h(e_5) = h(e_7) = 0.5, h(e_6) = 0.1$. The other edges have unit capacities. In this network, for the second objective, each edge has infinite capacity. We required that each unicast session must achieve at least unit rate. We considered two cost settings for the edges other than the outgoing edges of a_i's and the incoming edges of b_i's:

Scenario 3: $a(e_1) = a(e_2) = a(e_3) = a(e_4) = 10, a(e_6) = 100$. The other edges have unit costs.

Scenario 4: $a(e_3) = a(e_5) = a(e_7) = 10$, $a(e_6) = 100$. The other edges have unit costs.

Scenarios 2-4 model many practical transmission scenarios, where the end users are communicating with each other through long-distance links, which usually have limited bandwidths and higher costs than local links. The parameters for the simulated annealing algorithm were: $T_0 = 0.5$, $\alpha = 7$, $\beta = 92$, $\zeta = 46$, $\eta = 0.9$, $\kappa = 7$. We used GLPK 4.47 as the LP solver. All the simulations were run on a desktop computer with Intel core i3 CPU and 2GB memory.

B. Simulation Results

Let q_m denote the objective obtained by the partitioning algorithm, q_r the optimal objective achieved by routing, and λ_{succ} the number of scenarios in which q_m is better than q_r. We define the following metrics to evaluate the results: i) Performance gain, $\lambda = \frac{|q_m - q_r|}{q_r} \times 100\%$; ii) ratio of scenarios with gains, $\delta = \frac{\lambda_{succ}}{50} \times 100\%$; iii) the average running time τ of the partitioning algorithm. We averaged the performance gains over all scenarios with gains. The simulation results are shown in Table I. We make the following observations:

Except for Scenario 1, the ratio of scenarios for which MPC achieves gains over routing all increases with $|\Omega|$. Moreover, under Scenarios 2-4, MPC outperforms routing in almost half of the scenarios when $|\Omega| = 6, 7$. Under Scenario 3, MPC

TABLE I
SIMULATION RESULTS.

$\lvert\Omega\rvert$	δ (%)	λ (%)	τ (sec)
3	36	27.78	5.07
4	20	100	4.74
5	10	28.75	11.23
6	46	89.13	9.75
7	34	18.37	13.08

(a) Max. common rate, Scenario 1

$\lvert\Omega\rvert$	δ (%)	λ (%)	τ (sec)
3	18	94.44	3.23
4	20	100	4.74
5	30	96.67	6.71
6	46	89.13	9.75
7	50	84	15.32

(b) Max. common rate, Scenario 2

$\lvert\Omega\rvert$	δ (%)	λ (%)	τ (sec)
3	26	27.94	3.47
4	38	21.46	6.64
5	44	26.04	10.20
6	60	26.73	14.01
7	66	25.36	17.78

(c) Min. cost, Scenario 3

$\lvert\Omega\rvert$	δ (%)	λ (%)	τ (sec)
3	18	30.59	3.91
4	20	29.51	7.11
5	30	27.22	10.8
6	46	24.69	12.16
7	50	23.5	17.76

(d) Min. cost, Scenario 4

outperforms routing for more than 60% of the scenarios when $|\Omega| = 6, 7$. For these scenarios, MPC is more scalable than routing in the sense that the chance of obtaining performance gains through MPC increases with $|\Omega|$.

Under Scenarios 1-2, we observed that MPC achieves considerably better performance than routing for some scenarios. Under Scenario 2, MPC nearly doubles the performance of routing for the scenarios with gains. Under Scenarios 3-4, MPC still achieves a performance gain over routing ranging from 23% to 30% for the scenarios with gains. Since the ILP-based approach [7] converges very slow even for four unicast flows, we compare MPC with the evolutionary approach [8]. We consider the particular scenario presented in [8], where $\Omega = \{(a_1, b_7), (a_2, b_1), (a_7, b_2), (a_8, b_5)\}$ and the cost setting is the same as Scenario 3. For this scenario, MPC achieves a cost of 148, whereas the best cost achieved by the evolutionary approach over 30 runs of simulations is 156 [8].

The simulated annealing algorithm is efficient in finding good partitions. For most scenarios, the running time of the algorithm never exceeds 17 seconds. This is mainly because the algorithm only needs to search in the space of the partitions of Ω. Note that, even for $|\Omega| = 5$, the integer linear program in [7] contains around 68700 and 1400 variables, and around 67500 and 1700 constraints. This makes the converging speed of the integer linear program very slow, and may fail to converge in a reasonable time. In contrast, for $|\Omega| = 7$ and $|\mathcal{G}| = 6$, our linear program contains only 750 variables and 791 constraints, and each stage of the algorithm takes no more than 2 seconds for most scenarios. The evolutionary approach in [8] performs a random walk in the space of the coding vectors in the network. With each generation taking around one second, the total running time is around 100 seconds for 100 generations. In contrast, the simulated annealing algorithm preforms a random walk in the space of the partitions of Ω, which is much smaller than the space of the coding vectors. This greatly reduces the random steps the algorithm takes. The simulation results fully demonstrate the efficiency of the simulated annealing algorithm in finding good partitions.

VII. CONCLUSION

In this paper, we propose a novel approach, MPC, to construct linear network code for multiple-unicast scenario. We propose a set of linear constraints to describe the rate region achieved by MPC for a given partition of the multiple-unicast scenario. These linear constraints can be combined with various objectives and additional constraints to form linear programs to calculate the performance achieved by MPC. The succinct formulation of these linear programs allow us to quickly analyze the performance of MPC. We further present a practical partitioning algorithm to find good partitions such that the resulting MPC approximates the best performance among all MPCs. Simulation results demonstrate the performance of MPC and the efficiency of the partitioning algorithm. As part of our ongoing work, we are investigating how to apply MPC to other problems related to network coding.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, pp. 1204–1216, 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Information Theory*, vol. 49, pp. 371–381, 2003.

[3] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. on Information Theory*, vol. 51, 2005.

[4] 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. on Information Theory*, vol. 52, pp. 4413–4430, 2006.

[5] R. Dougherty, C. Freiling, K Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. on Information Theory*, vol. 51(8), pp. 2243–2256, Aug. 2005.

[6] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 142–150, 2004.

[7] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *ISIT*, pp. 1758–1762, 2006.

[8] M. Kim, M. Médard, U.-M. O'Reilly, and D. Traskov, "An evolutionary approach to inter-session network coding," in *IEEE INFOCOM*, pp. 450–458, 2009.

[9] C. C. Wang and N. B. Shroff, "Pairwise intersession network coding on directed networks," *IEEE Trans. on Information Theory*, vol. 56, pp. 3879–3900, 2010.

[10] A. Khreishah, C.-C. Wang, and N. B. Shroff, "Optimization based rate control for communication networks with inter-session network coding," in *IEEE INFOCOM*, pp. 81–85, 2008.

[11] J. Price and T. Javidi, "Network coding games with unicast flows," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 1302–1316, 2008.

[12] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: practical wireless network coding," *IEEE/ACM Trans. on Networking (TON)*, vol. 16, pp. 497–510, 2008.

[13] S. Zhang, Y. Liu, M. Zhao, and W. Zhou, "Improved GA solution on LNC coefficient matrix for multi-user cooperative communication," *IEEE PIMRC*, pp. 809-814, 2012.

[14] A. Das, S. Vishwanath, S. Jafar, and A. Markopoulou, "Network Coding for Multiple Unicasts: An Interference Alignment Approach," *ISIT*, pp. 1878–1882, 2010.

[15] A. Ramakrishnan, A. Das, H. Maleki, A. Markopoulou, S. A. Jafar, and S. Vishwanath, "Network Coding for Three Unicast Sessions: Interference Alignment Approaches," *Allerton Conference*, pp. 1033-1040, 2011.

[16] C. Meng, A. Ramakrishnan, A. Markopoulou, and S. A. Jafar,, "On the Feasibility of Precoding-Based Network Alignment for Three Unicast Sessions," *ISIT*, pp. 1907–1911, 2012.

[17] S. Chen, O. Gunluk, and B. Yener, "The multicast packing problem," *IEEE/ACM Transactions on Networking*, vol. 8, June 2000.

[18] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[19] C. Meng, A. Markopoulou, H. Seferoglu, K. W. Shum, C. Chan, "Packing multicasts for multiple-unicast scenarios," Tech. Rep., available on http://odysseas.calit2. uci.edu/wiki/doku.php/public:publication