On Distributed Scheduling in Wireless Networks Exploiting Broadcast and Network Coding

Tao Cui, Lijun Chen, and Tracey Ho, *Member, IEEE*

***Abstract*—In this paper, we consider cross-layer optimization in wireless networks with wireless broadcast advantage, focusing on the problem of distributed scheduling of broadcast links. The wireless broadcast advantage is most useful in multicast scenarios. As such, we include network coding in our design to exploit the throughput gain brought in by network coding for multicasting. We derive a subgradient algorithm for joint rate control, network coding and scheduling, which however requires centralized link scheduling. Under the primary interference model, link scheduling problem is equivalent to a maximum weighted hypergraph matching problem that is NP-complete. To solve the scheduling problem distributedly, locally greedy and randomized approximation algorithms are proposed and shown to have bounded worst-case performance. With random network coding, we obtain a fully distributed cross-layer design. Numerical results show promising throughput gain using the proposed algorithms, and surprisingly, in some cases even with less complexity than cross-layer design without broadcast advantage.**

***Index Terms*—Cross-layer design, rate control, wireless networks, broadcast advantage, network coding, hypergraph matching.**

I. Introduction

OPTIMIZATION-BASED cross-layer design for wireless networks has attracted much interest recently, see, e.g., [2]–[5] and the references therein. Joint optimization of multiple protocol layers can substantially increase the end-to-end throughput, or reduce power consumption. Most existing works on cross-layer design do not incorporate the exploitation of the wireless broadcast advantage where transmissions from an omnidirectional antenna can be received by any nodes that lie within its communication range. This broadcast advantage can result in throughput improvement and power saving especially with multicasting [6]. In this paper we consider distributed algorithms for wireless link scheduling that take the broadcast advantage into account. We apply this to a distributed joint optimization of multicast network coding, rate control, and channel access.

We model the wireless network as a directed hypergraph, with wireless broadcast being abstracted as a hyperarc. Scheduling with broadcast advantage is a hard problem in general. It forms a component of the algorithm proposed in [7], where it is assumed to be solved by a central controller. In this paper we focus on a simple, so-called primary interference model [8]. Under this interference model, any valid link schedule corresponds to a *hypergraph matching* and the optimal schedule corresponds to a *maximum weighted hypergraph matching*.

The maximum weighted hypergraph matching problem is, however, NP-complete [9]. We thus propose two classes of distributed approximation algorithms to solve the link scheduling problem under the primary interference model. The first class of algorithms is locally greedy algorithm, which chooses the locally heaviest hyperedge. We show that this algorithm returns a hypergraph matching with weight at least a constant factor of the maximum weighted hypergraph matching, giving a stability region for multi-hop communication within a constant factor of the region achievable with any hypergraph matching algorithm. The second class of algorithms is randomized algorithm, which always returns a maximal hypergraph matching. This gives a stability region for single hop communication that is at least $1/K$ of the region achievable with any hypergraph matching algorithm, where K is the maximum number of nodes in any hyperedge. The randomized algorithm can be readily turned into a constant-time algorithm.

We also provide a generalization of existing results in cross-layer optimization for multicast network coding in wireless networks. Our cross-layer design uses the framework of utility maximization, see, e.g., [5]. Our objective is to maximize the aggregate user utilities subject to flow conservation and schedulability constraint on the hypergraph. We then apply duality theory to decompose the problem vertically into rate control, network coding and session scheduling, and link scheduling subproblems, which interact through dual variables. Based on this decomposition, a distributed subgradient algorithm is proposed, whose session and link scheduling components are similar to the back-pressure algorithm in [7] which does not incorporate rate control.

The rest of this paper is organized as follows. In Section II, we briefly review some related work. Preliminaries including hypergraph formulation and network coding are presented in Section III. In Section IV, we present the cross-layer design algorithm. Link scheduling algorithms for the primary interference model are given in Section V. Simulation results are presented in Section VI, and finally, we conclude the paper in Section VII.

Paper approved by R. Fantacci, the Editor for Wireless Networks and Systems of the IEEE Communications Society. Manuscript received October 27, 2008; revised August 13, 2009.

The authors are with the Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA (e-mail: {taocui@, chen@cds., tho@}caltech.edu).

This work has been supported in part by DARPA grant N66001-06-C-2020, Caltech's Lee Center for Advanced Networking, the Okawa Foundation Research Grant and a gift from Microsoft Research. This paper has been presented in part at the IEEE Conference on Decision and Control, New Orleans, Louisiana USA, December 2007. An extended version can be found in [1].

Digital Object Identifier 10.1109/TCOMM.2010.04.080564

II. Related Work

Extensive research has been devoted to the cross-layer design for wireless networks but usually without considering network coding, see, e.g., [2]–[5]. Similar cross-layer design algorithm is proposed in [3], [4], [10], [11], and in particular, the impact of imperfect scheduling is also studied in [3]. In [10], the network capacity region is characterized, and a joint routing and power allocation policy is proposed to stabilize the system whenever the input rates are within this capacity region.

Network coding extends the functionality of network nodes from storing/forwarding packets to performing algebraic operations on received data. Starting with the work of [12], various potential benefits of network coding have been shown, including robustness to link/node failures and packet losses [13]. It is especially preferred in wireless networks, where the bandwidth is scarce. Distributed random linear coding schemes, see, e.g., [14], have made practical implementation of network coding possible.

There exist many works on optimization with network coding. Lun *et. al.* [15] propose a dual subgradient method for the problem of minimum cost multicasting with network coding. For rate control, the approach in [2] is extended to network coding in [16]. In [7], the rate stability region for a wireless network with and without correlated sources is characterized. In [17], medium access control and network coding is considered and broadcast advantage is also exploited. A set of conflict-free transmission schedules is predetermined, and the scheduling uses a time division mechanism, which is suboptimal.

The primary interference model was introduced in [8]. In [18], randomized algorithms are proposed, which achieve the capacity region with reduced complexity by comparing a random matching with the current matching. In [19], a distributed implementation of the algorithm in [18] is proposed. Scheduling algorithms based on maximal matching are also considered in works such as [3]. These matching-based algorithms do not consider the broadcast advantage.

III. Preliminaries

A. Network Model

A wireless network is modeled as a directed hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. A hyperarc is a pair (i, J), with $i \in \mathcal{N}$ the start node and $J \subseteq \mathcal{N}$ the set of end nodes, representing a broadcast link from node i to nodes in J. We assume that (i, J) is lossless, i.e., it does not experience packet erasures. When J only contains a single node j, the hypergraph reduces to the conventional graph model.

Let $\underline{S}(\tau) = \{S_{i,j}(\tau)\}$ denote the matrix process of channel states, where $S_{i,j}(\tau)$ represents the channel state from node i to node j at time τ. Every time slot, node i determines transmission rates on each hyperarc $(i, J) \in \mathcal{A}$ by allocating a power matrix $\underline{P} = \{P_{iJ}\}$ subject to a total power constraint

$$\sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \le P_i^{\text{tot}}, \ \forall i \in \mathcal{N}, \tag{1}$$

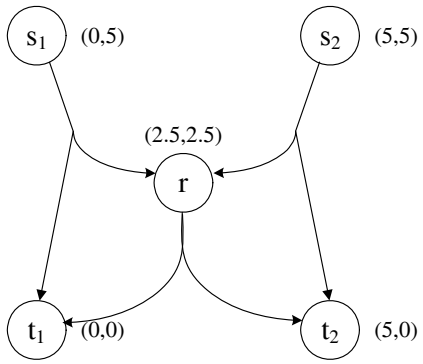


Fig. 1. Wireless butterfly network.

where P_i^{tot} is the maximal total power allowable at node i. Hyperarc rates are determined by a rate-power curve $\underline{r}(\underline{P}, \underline{S}) = \{r_{iJ}(\underline{P}, \underline{S})\}$, where $r_{iJ}(\underline{P}, \underline{S})$ is the rate at which packets are transmitted on hyperarc (i, J) and received by all the nodes in J. The rate-power curve is assumed to be given, either based on information theoretic capacity or on actual coding and modulation schemes. By time-sharing, the capacity region is the convex hull $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ of all achievable rate vectors $\underline{r}(\underline{P}, \underline{S})$.

B. Network Coding

A set $\mathcal{M}$ of multicast sessions is transmitted through the network. Each session $m \in \mathcal{M}$ is associated with a set $\mathcal{S}_m \subset \mathcal{N}$ of sources and a set $\mathcal{T}_m \subset \mathcal{N}$ of sinks. In session m, each source $s \in \mathcal{S}_m$ multicasts x^{ms} bits per second to all the sinks in $\mathcal{T}_m$. In session m, each source $s \in \mathcal{S}_m$ multicasts x^{ms} bits per second to all the sinks in $\mathcal{T}_m$. By the flow conservation condition,

$$\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} g_{iJj}^{mst} - \sum_{j\in\mathcal{N}} \sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst} = \sigma_i^{ms}, \tag{2}$$

$\forall i \in \mathcal{N},\ s \in \mathcal{S}_m,\ t \in \mathcal{T}_m,\ m \in \mathcal{M}$, where $\sigma_i^{ms} = x^{ms}$ if $i = s$, $\sigma_i^{ms} = -x^{ms}$ if $i = t$, $\sigma_i^{ms} = 0$ otherwise, and g_{iJj}^{mst} is the information rate from source s to sink t of session m over (i, J) and is intended to node $j \in J$.

In network coding, nodes can algebraically combine packets, increasing efficiency of information transfer. We assume that coding is done only across packets of the same multicast session. To achieve a fully distributed design, we use distributed random network coding, see, e.g., [20].

We define f_{iJ}^m as the physical flow of session m on hyperarc (i, J) as opposed to the virtual flow g_{iJj}^{mst} in (2). By the flow sharing property of network coding [12] and the rate constraint, we have the following two constraints

$$\sum_{s\in\mathcal{S}_m} \sum_{j\in J} g_{iJj}^{mst} \le f_{iJ}^m, \quad \forall (i, J) \in \mathcal{A},\, m \in \mathcal{M},\, t \in \mathcal{T}_m, \tag{3}$$

$$\sum_{m\in\mathcal{M}} f_{iJ}^m \le r_{iJ}, \qquad \forall (i, J) \in \mathcal{A}, \tag{4}$$

where $\{r_{iJ}\} \in \text{Co}(\underline{r}(\underline{P}, \underline{S}))$.

To illustrate this, consider the network in Figure 1. There is a single multicast session m with two source nodes and two sink nodes. The hyperarc $(s_1, \{r, t_1\})$ carries actual flow

$f^m_{s_1\{r,t_1\}}$ and virtual flows $g^{ms_1t_1}_{s_1\{r,t_1\}t_1}$ to receiver t_1, $g^{ms_1t_1}_{s_1\{r,t_1\}r}$ to receiver t_1 via r, and $g^{ms_1t_2}_{s_1\{r,t_1\}r}$ to receiver t_2 via r. The flow sharing condition (3) for this hyperarc is

$$g^{ms_1t_1}_{s_1\{r,t_1\}t_1} + g^{ms_1t_1}_{s_1\{r,t_1\}r} \leq f^m_{s_1\{r,t_1\}} \quad (5)$$

$$g^{ms_1t_2}_{s_1\{r,t_1\}r} \leq f^m_{s_1\{r,t_1\}} \quad (6)$$

IV. Cross-Layer Design with Broadcast Advantage and Network Coding

In this section, we derive a cross-layer design by using the utility maximization framework, which is an extension of [7], [16]. Each source s of session m is associated with a utility function $U_{ms}(x^{ms})$, which is assumed to be strictly concave, non-decreasing and twice continuously differentiable. We formulate network resource allocation as the following optimization problem

$$\begin{aligned}
\max_{\substack{x^{ms},g^{mst}_{iJj},\\ f^m_{iJ},r_{iJ},P_{iJ}}} \quad & \sum_{m\in\mathcal{M},s\in\mathcal{S}_m} U_{ms}(x^{ms}) \\
\text{s.t.} \quad & \sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} g^{mst}_{iJj} - \sum_{j\in\mathcal{N}}\sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g^{mst}_{jIi} = \sigma^{ms}_i, \\
& \qquad \forall i \neq t, s\in\mathcal{S}_m, t\in\mathcal{T}_m, m\in\mathcal{M}, \\
& \sum_{s\in\mathcal{S}_m,\, j\in J} g^{mst}_{iJj} \leq f^m_{iJ},\ \forall (i,J), t\in\mathcal{T}_m, m\in\mathcal{M}, \\
& \sum_{m\in\mathcal{M}} f^m_{iJ} \leq r_{iJ},\ \forall (i,J), \\
& \{r_{iJ}\} \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \leq P^{\mathrm{tot}}_i,\ \forall i,
\end{aligned} \quad (7)$$

where the constraints come from equations (2)-(4). Here we do not include flow conservation equation at destinations, which is automatically guaranteed by the flow conservation at the source and intermediate nodes. Problem (7) is strictly convex and has a unique solution with respect to source rates x^{ms}. The partial dual function to (7), by relaxing only the first set of constraints, can be decomposed into the following two subproblems

$$\phi_1(q) = \max_{x^{ms}} \sum_{m,s} U_{ms}(x^{ms}) - \sum_{m,s}\left(\sum_t q^{mst}_s\right) x^{ms}, \quad (8)$$

$$\begin{aligned}
&\phi_2(q) \\
&= \max_{\substack{g^{mst}_{iJj},f^m_{iJ},\\ r_{iJ},P_{iJ}}} \sum_{i,m,s,t} q^{mst}_i \Bigg(\sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} g^{mst}_{iJj} \\
&\qquad\qquad - \sum_{j\in\mathcal{N}}\sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g^{mst}_{jIi} \Bigg), \\
&\text{subject to} \sum_{s\in\mathcal{S}_m,\, j\in J} g^{mst}_{iJj} \leq f^m_{iJ},\ \sum_{m\in\mathcal{M}} f^m_{iJ} \leq r_{iJ}, \\
&\qquad \{r_{iJ}\} \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \leq P^{\mathrm{tot}}_i,
\end{aligned} \quad (9)$$

where q^{mst}_i is the Lagrange multiplier at node i for source s and sink t of session m, and will be interpreted as congestion price. The first subproblem is rate control. The second is joint network coding and scheduling. Thus, by dual decomposition, the flow optimization problem decomposes into separate "local" optimization problems of transport and network/link layers, respectively. The two subproblems interact through the dual variable q.

Rate Control: At time τ, given dual variable $q(\tau)$, each source adjusts its sending rate according to the aggregate dual variable $\sum_t q^{mst}_s$ that is generated locally at the source

$$x^{ms}(\tau+1) = U'^{-1}_{ms}\left(\sum\nolimits_t q^{mst}_s(\tau)\right). \quad (10)$$

Session Scheduling and Network Coding: Note that (9) is equivalent to the following problem

$$\begin{aligned}
&\max_{g^{mst}_{iJj},f^m_{iJ},r_{iJ},P_{iJ}} \sum_{(i,J),m,t}\sum_{s,j\in J} g^{mst}_{iJj}\left(q^{mst}_i - q^{mst}_j\right), \\
&\text{subject to} \sum_{s,\,j\in J} g^{mst}_{iJj} \leq f^m_{iJ},\ \sum_{m\in\mathcal{M}} f^m_{iJ} \leq r_{iJ}, \\
&\qquad \{r_{iJ}\} \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \leq P^{\mathrm{tot}}_i, \\
&= \max_{f^m_{iJ},r_{iJ},P_{iJ}} \sum_{(i,J),m} f^m_{iJ} \sum_t \max_{s,j\in J}\left[q^{mst}_i - q^{mst}_j\right]^+, \\
&\text{subject to} \sum_{m\in\mathcal{M}} f^m_{iJ} \leq r_{iJ},\ \{r_{iJ}\} \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})), \\
&\qquad \sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \leq P^{\mathrm{tot}}_i,
\end{aligned} \quad (11)$$

where $[\cdot]^+$ denotes the projection onto $\mathbb{R}^+$. The last equality in (11) comes from the fact that $\max_{g^{mst}_{iJj}} \sum_{s,j\in J} g^{mst}_{iJj}\left(q^{mst}_i - q^{mst}_j\right)$, subject to $\sum_{s,\,j\in J} g^{mst}_{iJj} \leq f^m_{iJ}$ is a linear program, so we can always choose an extreme point solution, i.e.,

$$g^{mst}_{iJj} = \begin{cases} f^m_{iJ}, & \text{if } s=\hat{s}^{mt},\ j=\hat{j}^{mt}, \\ & \text{and } q^{mst}_i - q^{mst}_j \geq 0, \\ 0, & \text{otherwise,} \end{cases} \quad (12)$$

where $\{\hat{s}^{mt},\hat{j}^{mt}\} = \arg\max_{s,j\in J}\left(q^{mst}_i - q^{mst}_j\right)$.

Let $\hat{m}_{iJ} = \arg\max_m \sum_t \max_{s,j\in J}\left[q^{mst}_i - q^{mst}_j\right]^+$ be the session with the maximum aggregate differential link prices over hyperarc (i,J). For each hyperarc (i,J), a random linear combination of packets from sources $\hat{s}^{\hat{m}_{iJ}t}$, $\forall t\in\mathcal{T}_{\hat{m}_{iJ}}$, in session $\hat{m}_{iJ}$ is broadcast to all nodes in J at the rate of r_{iJ}, where the packets received by node $j^{\hat{m}_{iJ}t}$ are intended for sink t in session $\hat{m}_{iJ}$. This is equivalent to solving (9) by

$$g^{mst}_{iJj}(q) = \begin{cases} r_{iJ}, & \text{if } m=\hat{m}_{iJ},\ s=\hat{s}^{mt},\ j=\hat{j}^{mt}, \\ & \text{and } \max\limits_{s,j\in J}\left[q^{mst}_i - q^{mst}_j\right]^+ > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

Link Scheduling and Power Control: Define $w_{iJ} = \max_m \sum_t \max_{s,j\in J}\left[q^{mst}_i - q^{mst}_j\right]^+$. The joint link scheduling and power control problem becomes

$$\begin{aligned}
&\max_{r,P} \sum_{(i,J)\in\mathcal{A}} w_{iJ} r_{iJ} \\
&\text{subject to } \{r_{iJ}\} \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{\{J|(i,J)\in\mathcal{A}\}} P_{iJ} \leq P^{\mathrm{tot}}_i.
\end{aligned} \quad (14)$$

The problem (14) is in general a difficult global optimization problem. In Section V, we will discuss a special interference model such that (14) can be solved distributedly in polynomial time.

Note that our scheduling problem (13)-(14) generalizes the back-pressure policy in [10], [21] by taking into account the differential backlog between node i and all nodes $j \in J$ instead of only a single node. Clearly, when $J = \{j\}$, our policy reduces to those in [10], [21]. The scheduling problem (13)-(14) is similar to that in [7], where the former is derived using the optimization framework while the latter is obtained by good intuition.

Dual Variable Update: At time $\tau+1$, each node i updates its dual variable q according to the subgradient algorithm

$$q_i^{mst}(\tau+1) = \begin{cases} q_i^{mst}(\tau) + \gamma_\tau \Bigg(x^{ms}(\tau) - \sum\limits_{\{J|(i,J)\in\mathcal{A}\}} \sum\limits_{j\in J} g_{iJj}^{mst}(q(\tau)) \\ \qquad + \sum\limits_{j\in\mathcal{N}} \sum\limits_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst}(q(\tau)) \Bigg), & \text{if } i = s, \\ q_i^{mst}(\tau) + \gamma_\tau \Bigg(\sum\limits_{j\in\mathcal{N}} \sum\limits_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst}(q(\tau)) \\ \qquad - \sum\limits_{\{J|(i,J)\in\mathcal{A}\}} \sum\limits_{j\in J} g_{iJj}^{mst}(q(\tau)) \Bigg), & \text{otherwise,} \end{cases} \tag{15}$$

where γ_τ is a positive stepsize. After node i updates q_i^{mst}, it passes the value to all its neighbors. Note that the algorithm (10)-(15) only requires nodes to communicate with neighbors.

Now, we discuss the convergence and optimality of this cross layer design. Let the primal function (i.e., the total achieved network utility) be $P(x)$ and achieve its optimum at x^*. Let q^* be a dual optimal. Define $\bar{x}(\tau) := \frac{1}{k}\sum_{k=1}^{\tau} x(k)$, the average data rate up to time τ, and $\bar{q}(\tau) := \frac{1}{k}\sum_{k=1}^{\tau} q(k)$, the average dual variable (congestion price) up to time τ. Let $g(q)$ be a subgradient of dual function $\phi(q)$. When the joint link scheduling and power control problem (14) is solved exactly, by using results on the convergence of the subgradient method, see, e.g., [4], [11], we can show the following result.

Theorem 1: If the norm of the subgradients is uniformly bounded, i.e., there exists $\mathfrak{g}$ such that $\|g(q)\|_2 \le \mathfrak{g}$ for all q, and a constant stepsize γ is adopted in (15), then the following inequalities hold

$$\limsup_{\tau\to\infty} \phi(\bar{q}(\tau)) \le \phi(q^*) + \frac{\gamma\mathfrak{g}^2}{2}, \tag{16}$$

$$\liminf_{\tau\to\infty} P(\bar{x}(\tau)) \ge P(x^*) - \frac{\gamma\mathfrak{g}^2}{2}. \tag{17}$$

Theorem 1 implies that the average source rate and congestion price approach the corresponding optima when the stepsize γ is small enough. We omit the proof here for brevity. We may also establish the convergence of our cross-layer design in a slightly different sense, by using the standard convergence results for the subgradient method [22].

V. Link Scheduling

In this section, we study the joint link scheduling and power control problem (14) for networks with primary interference. A system is stable if the queue lengths at all nodes remain finite all the time. Note that the queue length at node i can be written as q_i^{mst}/γ for a constant stepsize γ [3]. A rate vector $\vec{x} = \{x^{ms}\}$ is feasible if there exists a scheduling policy that stabilizes the system with $\vec{x}$. We are interested in those scheduling policies that can stabilize the system for any rate vector within $\eta\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ denotes the network capacity region characterized by the constraints in (7). $\eta \in (0,1]$ is a constant that characterizes the performance of the scheduling policy. For example, by Theorem 5 that will be presented later, $\eta = \max\{\frac{1}{K}, \frac{1}{\kappa}\}$ for Algorithm 1 and Algorithm 2. By following the same argument as in [3], we can show that the performance of the joint design with each of our proposed scheduling algorithms is not worse than the design specified by the following optimization problem

$$\max \sum_{m\in\mathcal{M}, s\in\mathcal{S}_m} U_{ms}(x^{ms}), \text{ subject to } \vec{x} \in \eta\mathbf{\Lambda}, \tag{18}$$

with appropriate η that is determined by the worst-case performance bound of the corresponding scheduling algorithm.

A. Problem Formulation

Under the primary interference model [8], only those links that do not share nodes can transmit at the same time. It models a situation where each node is equipped with a single transceiver and neighboring nodes can transmit simultaneously using orthogonal CDMA or FDMA channels. Under this interference model, without using broadcast advantage, any feasible schedule corresponds to a matching [21]. With the broadcast advantage, any feasible schedule corresponds to a hypergraph matching of the hypergraph $\mathcal{H}$, and (14) reduces to the maximum weighted hypergraph matching[1] problem.

Let $\mathbf{\Pi}$ denote the set of all hypergraph matchings of the hypergraph $\mathcal{H}$. Assume that if hyperarc (i,J) is active, it transmits at a given rate $r_{iJ}(P_i^{\text{tot}}, \underline{S})$. We can represent a hypergraph matching π as an $|\mathcal{A}|$-dimensional rate vector ξ^π

$$\xi_{iJ}^\pi = \begin{cases} r_{iJ}(P_i^{\text{tot}}, \underline{S}), & \text{if } (i,J) \in \pi, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

The achievable rate region $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ is then written as

$$\text{Co}(\underline{r}(\underline{P}, \underline{S})) \triangleq \left\{ \mathbf{r} : \mathbf{r} = \sum_{\pi\in\mathbf{\Pi}} \alpha_\pi \xi^\pi,\ \alpha_\pi \ge 0, \sum_{\pi\in\mathbf{\Pi}} \alpha_\pi = 1 \right\}. \tag{20}$$

Note that $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ is a polytope. So, we can always pick up an extreme point maximizer for the scheduling problem (14), which corresponds to a maximum weighted hypergraph matching in $\mathcal{H}$.

We first transform the directed hypergraph to an equivalent undirected hypergraph $\tilde{\mathcal{H}} = (\mathcal{V}, \mathcal{E}_h)$, where $\mathcal{H}$ and $\tilde{\mathcal{H}}$ have the same node set. Note that hyperarcs (i,J) and (j,I) mutually interfere and have the same interference/contention relations with other hyperarcs if $\{i\}\cup J = \{j\}\cup I$. Define an undirected hyperedge $e \subseteq \mathcal{V}$ in $\mathcal{E}_h$, which corresponds to all hyperarcs (i,J) such that $e = \{i\}\cup J$. The weight of hyperedge $e \in \mathcal{E}_h$ is

$$\tilde{w}_e = \max_{\{(i,J)\in\mathcal{A},\, \{i\}\cup J = e\}} w_{iJ} r_{iJ}(\underline{P}, \underline{S}). \tag{21}$$

[1]A hypergraph matching is defined as a set of hyperarcs with no pair incident to the same node.

The problem (14) is then equivalent to the maximum weighted hypergraph matching (or maximum weighted set packing) problem on the weighted hypergraph $\tilde{\mathcal{H}}$.

Different from the maximum weighted matching problem on graphs which can be computed in polynomial time, the maximum weighted hypergraph matching problem is NP-complete [9]. Also, we would like distributed algorithms. Both factors suggest that we should focus on approximation algorithms.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with the same node set as $\mathcal{H}$. We assume that there exists an edge between node i and node j if and only if $\min\{\mathrm{SNR}_{ij}, \mathrm{SNR}_{ji}\} \geq \lambda$, where λ is a predefined threshold. This means that if i can hear j, then j can hear i. Let $N(v)$ denote the neighbor node set of node v in $\mathcal{G}$. We call $\mathcal{G}$ *connectivity graph* in the following.

B. Local Optimal Algorithms

A linear time approximation algorithm with bounded worst-case performance for maximum weighted graph matching is proposed in [23], which adds a locally optimal edge into the matching at each step. Motivated by [23], our algorithm adds a locally heaviest hyperedge into the hypergraph matching at each step.

Definition 1 (**locally heaviest hyperedge**): A hyperedge e is a locally heaviest hyperedge if its weight is at least as large as the weight of all adjacent hyperedges, i.e., $\tilde{w}_e \geq \tilde{w}_f$ for all $f \in \mathcal{E}_h$ such that $f \cap e \neq \emptyset$.

The distributed local optimal hypergraph matching algorithm (DLOHMA) is given in Algorithm 1. In Algorithm 1, the set Γ_i keeps track of the set of neighbors of node i that are still not matched, which is initialized to be all its neighbors in $\mathcal{G}$. Node i also maintains, for each neighbor $j \in \Gamma_i$, the set $\Gamma_j^{(i)}$ which keeps track of the set of unmatched neighbors of j, and knows the queue lengths of its two hop neighbors. This allows node i to compute the weight $\tilde{w}_e$ of any edge e involving itself, as defined in (21). The vector C_i counts the number of matching e_i^* messages that have been received, which is initialized to be a null vector (line 3). Each node i broadcasts a matching e_i^* message, where $e_i^* = \{i\} \cup J^*$ is the maximum weight hyperedge in $\tilde{\mathcal{H}}$ containing i (lines 5-8). If node i receives $|J^*|$ matching e_i^* messages, hyperedge e_i^* is added to the hypergraph matching as e_i^* is a locally heaviest hyperedge. It broadcasts a drop e_i^* message to indicate that i is matched and unavailable, and at the same time to tell all nodes in e_i^* that they are matched (lines 26-28). If node i receives a drop e message and node i is not in e, it first checks whether some nodes of e are in Γ_i. If so, i is the direct neighbor of some nodes in e and i broadcasts the drop e message to let i's neighbors (two-hop neighbors of the nodes in e) know that all the nodes in e are matched. If not, i does not need to forward the drop message. Node i then removes the nodes in e from Γ_i and all $\Gamma_j^{(i)}$, $j \in \Gamma_i$. Furthermore, if some nodes in J^* are in e, the hyperedge e_i^* is dropped. Node i then finds another candidate set J^*, and it broadcasts a new matching e_i^* message (lines 16-23). If i receives a drop e message and node i is in e, i will broadcast a drop e message if it did not do so before, i.e., Γ_i is nonempty (line 23).

Note that some nodes in the locally heaviest hyperedge may not be able to hear each other. These nodes cannot receive

DLOHMA: ($\mathcal{G}$)

1 **for** *each node* $i \in \mathcal{V}$ **do**
2 Broadcast the set $\{\mathrm{SNR}_{ij} | j \in N(i)\}$ to all its neighbor nodes ;
3 Set $C_i = \emptyset$, $\Gamma_i = N(i)$, and $\Gamma_j^{(i)} = N(j)$, $\forall j \in N(i)$;
4 **end**
5 **for** *each node* $i \in \mathcal{V}$ **do**
6 Find a node set J^* by $J^* = \{j^*\} \cup L^* - i$ where j^*, L^* are obtained via

$$(j^*, L^*) = \arg\max_{j \in \Gamma_i \cup \{i\}, \left\{L | L \subseteq \Gamma_j^{(i)}, i \in L\right\}} w_{jL} r_{jL}(\underline{P}, \underline{S}), \tag{22}$$

and w_{jL}, r_{jL} are defined in (14) ;
7 **if** $J^* \neq \emptyset$ **then** Broadcast a *matching* $e_i^* = \{i\} \cup J^*$ message;
8 **end**
9 **while** $\exists i$, $\Gamma_i \neq \emptyset$ **do**
10 **if** *node* i *receives a message* m *which is has not received* **then**
11 **switch** m **do**
12 **case** *matching* e
13 $C_i(e) = C_i(e) + 1$;
14 **end**
15 **case** *drop* e
16 **if** $i \notin e$ **then**
17 **if** $e \cap \Gamma_i \neq \emptyset$ **then** Broadcast a *drop* e message;
18 Remove the nodes in e from Γ_i and all $\Gamma_j^{(i)}$, $j \in \Gamma_i$;
19 **if** $e \cap J^* \neq \emptyset$ **then**
20 Find a node set J^* by (22);
21 **if** $J^* \neq \emptyset$ **then** Broadcast a *matching* $e_i^* = \{i\} \cup J^*$ message;
22 **end**
23 **else if** $\Gamma_i \neq \emptyset$ **then** Broadcast a *drop* e message, and set $\Gamma_i = \emptyset$;
24 **end**
25 **end**
26 **if** $J^* \neq \emptyset$ ***and*** $C_i(e_i^*) = |J^*|$ **then**
27 Broadcast a *drop* e_i^* message, and set $\Gamma_i = \emptyset$;
28 **end**
29 **end**
30 **end**

Algorithm 1: Distributed local optimal algorithm.

$|J^*|$ matching e messages and conclude that e is the locally heaviest hyperedge. But at least one node can hear all the other nodes in the hyperedge. This is the reason why we broadcast a drop e message in line 27.

In Algorithm 1, we assume that all hyperedges have different weights. If they do not, we can always break ties by adding a small constant ϵ_e to w_e (different e has different ϵ_e). For example, we can change w_{iJ} or r_{iJ} by a small constant. In the following, we also assume that the cardinality of all hyperedges in $\mathcal{E}_h$ is bounded from above by a constant K. Let $\kappa = \max_{m \in \mathcal{M}} |\mathcal{T}_m| + 1$.

Proposition 1: The hyperedge e_i^* in line 26 is a locally heaviest hyperedge.

Proof: From (22), $\tilde{w}_e \geq \tilde{w}_f$ for any f that contains i. If node i receives $|J^*|$ matching e messages, we can conclude that $\tilde{w}_e \geq \tilde{w}_f$ for any f that contains i. Therefore, we have $\tilde{w}_e \geq \tilde{w}_f$ for any f such that $f \cap e \neq \emptyset$, and e is a locally heaviest hyperedge. □

Proposition 2: In Algorithm 1, each node i broadcasts at most $\sum_{j \in N(i)} |N(j)| + |N(i)|$ messages.

Proof: When the algorithm begins, node i first broadcasts a

matching message (line 7). After that it broadcasts a matching message only when it receives a drop e message from one of its neighbors and it is not in e. It initiates a drop message only when it gets matched. After that $\Gamma_i = \emptyset$ and no more messages will be sent. It forwards a drop e message only when $i \notin e$ and $e \cap \Gamma_i \neq \emptyset$. From Algorithm 1, node i can receive at most $N(i)$ drop messages initiated by its one-hop neighbors and $\sum_{j\in N(i)}(|N(j)|-1)$ drop messages initiated by its two-hop neighbors. Therefore, the worst case is broadcasting $\sum_{j\in N(i)}|N(j)|-1$ matching e messages and forwarding $|N(i)|$ drop e messages. This require broadcasting $\sum_{j\in N(i)}|N(j)|+|N(i)|$ messages. □

Different from [23] where node i only sends a message to node j, we make use of the broadcast property of wireless communication, which reduces the number of messages.

Theorem 2: The complexity of Algorithm 1 is $O\left(K^3|\mathcal{E}|\sum_{k=1}^{\min\{\kappa,K\}-1}\binom{K-1}{k}\right)$ time and the number of time-slots required to finish Algorithm 1 is $O(|\mathcal{V}|)$.

Proof: By Proposition 2, each node broadcasts at most $\sum_{j\in N(i)}|N(j)|+|N(i)|$ messages. Thus, there are at most $\sum_{i\in\mathcal{V}}\sum_{j\in N(i)}|N(j)|+|N(i)| \leq (2K+1)|\mathcal{E}|$ broadcasted messages. Each broadcasted message is received by at most $K-1$ neighbor nodes. Therefore, all the nodes receive at most $(2K+1)(K-1)|\mathcal{E}|$ messages. The **while** loop of Algorithm 1 (lines 9-30) has at most $(2K+1)(K-1)|\mathcal{E}|$ iterations. In each iteration, we need to perform (22) at most once. We can solve (22) by performing the inner max first with fixed j and then the outer max by varying j. By the definition of w_{iJ} and $r_{iJ}(\underline{P},\underline{S})$ in (14), we can write the inner max of (22) as

$$\max_{\{L|L\subseteq\Gamma_j,\, i\in L\}}\frac{1}{G}\log\left(1+P_j^{\mathrm{tot}}\min_{l\in L}\frac{|h_{j,l}|^2}{\sigma_l^2}\right)\times\left(\sum_t\max_{s,l\in L}\left[q_j^{mst}-q_l^{mst}\right]^+\right). \tag{23}$$

Clearly, given any set L with $|L|>\kappa-1$, we can always find a subset L' of L such that the weight of L' is at least that of L because $\sum_t\max_{s,l\in L}\left[q_j^{mst}-q_l^{mst}\right]^+$ in (23) contains at most $\kappa-1$ summands, and

$$\log\left(1+P_j^{\mathrm{tot}}\min_{l\in L}\frac{|h_{j,l}|^2}{\sigma_l^2}\right)\leq\log\left(1+P_j^{\mathrm{tot}}\min_{l\in L'}\frac{|h_{j,l}|^2}{\sigma_l^2}\right), \tag{24}$$

$\forall L'\subseteq L$. Therefore, we only need to consider those L with $|L|\leq\kappa-1$. The number of such L's is at most $\sum_{k=1}^{\min\{\kappa,K\}-1}\binom{K-1}{k}$. Also the number of j in $\Gamma_i\cup\{i\}$ is at most K. Thus, the complexity[2] of solving (22) is $O\left(K\sum_{k=1}^{\min\{\kappa,K\}-1}\binom{K-1}{k}\right)$ and the complexity of Algorithm 1 is $O\left(K^3|\mathcal{E}|\sum_{k=1}^{\min\{\kappa,K\}-1}\binom{K-1}{k}\right)$.

[2]We can sort $\left[q_j^{mst}-q_l^{mst}\right]^+$ and $|h_{j,l}|^2$ beforehand so that computing $\max_{l\in L}\left[q_j^{mst}-q_l^{mst}\right]^+$ and $\min_{l\in L}|h_{j,l}|^2$ takes $O(1)$ time.

On the other hand, Algorithm 1 is a parallel algorithm. We assume that every message broadcast takes one time-slot. It is easy to see that at least one locally heaviest hyperedge always exists. Let t denotes the time-slot that a locally heaviest hyperedge e is found through line 26. Note that at least one node in a hyperedge can hear all the other nodes. From this node, it takes at most one time-slot to let all the nodes in e know that they are matched. It takes at most two time-slots to have this drop e message propagate to all two-hop neighbors of the nodes in e. It takes one time-slot for all one-hop and two-hop neighbors of nodes in e to send a new matching message. Therefore, at the $t+4$ time-slot, we can find the next locally heaviest hyperedge. By removing the nodes in the locally heaviest hyperedge, the number of nodes in $\tilde{\mathcal{H}}$ is reduced at least by two. Therefore, by induction, the algorithm takes at most $O(|\mathcal{V}|)$ time-steps. □

If we do not consider the complexity of computing (22) as in [23], Algorithm 1 runs in linear time in the number of edges in the connectivity graph, i.e., $|\mathcal{E}|$ (not $|\mathcal{E}_h|$), which has the same complexity as the algorithms in [23] for finding a locally heaviest matching. This is because we use the broadcast advantage of the wireless communication.

Theorem 3: Algorithm 1 computes a hypergraph matching HM_{LO} with at least $\max\{\frac{1}{K},\frac{1}{\kappa}\}$ of the weight of a maximum weighted hypergraph matching HM_{MW}.

Proof: We show this by induction. Let HM_{LO}^i be the hypergraph matching set after the i-th hyperedge is added, and V_{LO}^i be the set of matched vertices in HM_{LO}^i. The total weight of all the hyperedges in M is denoted as $W(M)$. We need to show that for all i, the following is true

$$W(HM_{\mathrm{LO}}^i)\geq \max\left\{\frac{1}{K},\frac{1}{\kappa}\right\}W\left(\{e|e\in HM_{\mathrm{MW}},\, e\cap V_{\mathrm{LO}}^i\neq\emptyset\}\right). \tag{25}$$

Clearly, (25) is true for $i=0$ as $HM_{\mathrm{LO}}^i=\emptyset$. We assume (25) is true for $i=k-1$. Let e^k be the k-th hyperedge added into HM_{LO}. By Proposition 1 and the definition of locally heaviest hyperedge,

$$W(e^k)\geq W(e),\ \forall e\in\mathcal{E}_h,\ \text{and}\ e\cap V_{\mathrm{LO}}^{k-1}=\emptyset. \tag{26}$$

All the hyperedges adjacent to the nodes in V_{LO}^{k-1} have been excluded according to Algorithm 1. Therefore, we have

$$W(e^k)\geq W(e),\ \forall e\in HM_{\mathrm{MW}},\, e\cap V_{\mathrm{LO}}^{k-1}=\emptyset,\ \text{and}\ e\cap e^k\neq\emptyset. \tag{27}$$

Similar to the argument in Theorem 3, the size of e^k is at most $\min\{K,\kappa\}$. Thus, e^k intersects with at most $\min\{K,\kappa\}$ hyperedges in HM_{MW}, which indicates

$$W(e^k)\geq\max\left\{\frac{1}{K},\frac{1}{\kappa}\right\}\times W\left(\left\{e|e\in HM_{\mathrm{MW}},\, e\cap V_{\mathrm{LO}}^{k-1}=\emptyset,\ \text{and}\ e\cap e^k\neq\emptyset\right\}\right). \tag{28}$$

Adding both sides of (28) and (26) with $i=k-1$, we find (26) is still true for $i=k$. Therefore, (26) is true for any i, and the theorem is proved. □

In Algorithm 1, some matched nodes may not contribute much to a locally heaviest hyperedge. But when these nodes are matched in other hyperedges, they may contribute more,

which results in a hypergraph matching with higher weight. Instead of choosing the hyperedge according to its weight, we use the average hyperedge weight, i.e., $\bar{w}_e = \tilde{w}_e/|e|$. We modify Algorithm 1 to **Algorithm 2** by simply replacing $\tilde{w}_e$ with $\bar{w}_e$. Both the complexity and the approximation ratio of Algorithm 2 are identical to those of Algorithm 1.

Theorem 4: Algorithm 2 computes a hypergraph matching HM_{LO2} with at least $\max\{\frac{1}{K}, \frac{1}{\kappa}\}$ of the weight of a maximum weighted hypergraph matching HM_{MW}.

Proof: We show this as Theorem 3 by induction. Let HM^i_{LO2} be the hypergraph matching set after the i-th hyperedge is added, and V^i_{LO2} be the set of matched vertices in HM^i_{LO2}. The total weight of all the hyperedges in M is denoted as $W(M)$. We need to show that for all i, the following is true

$$W(HM^i_{\mathrm{LO2}}) \geq \max\left\{\frac{1}{K}, \frac{1}{\kappa}\right\} W\left(\{e | e \in HM_{\mathrm{MW}},\, e \cap V^i_{\mathrm{LO2}} \neq \emptyset\}\right). \tag{29}$$

Clearly, (29) is true for $i = 0$ as $HM^i_{\mathrm{LO2}} = \emptyset$. We assume (29) is true for $i = k-1$. Let e^k be the k-th hyperedge added into HM_{LO2}. By Proposition 1 and the definition of locally heaviest hyperedge, we have

$$\bar{w}_{e^k} = \frac{W(e^k)}{|e^k|} \geq \frac{W(e)}{|e|} = \bar{w}_e,\ \forall e \in \mathcal{E}_h,\ \text{and}\ e \cap V^{k-1}_{\mathrm{LO2}} = \emptyset. \tag{30}$$

Therefore, we have

$$\frac{W(e^k)}{|e^k|} \geq \frac{W(e)}{|e|},\ \forall e \in HM_{\mathrm{MW}},\ e \cap V^{k-1}_{\mathrm{LO}} = \emptyset,\ \text{and}\ e \cap e^k \neq \emptyset. \tag{31}$$

We then have

$$\begin{aligned} \sum_{\substack{\{e \in HM_{\mathrm{MW}},\, e \cap V^{k-1}_{\mathrm{LO}} = \emptyset,\\ \text{and } e \cap e^k \neq \emptyset\}}} W(e) &= \sum_{\substack{\{e \in HM_{\mathrm{MW}},\, e \cap V^{k-1}_{\mathrm{LO}} = \emptyset,\\ \text{and } e \cap e^k \neq \emptyset\}}} |e| \frac{W(e)}{|e|} \\ &\leq \sum_{\substack{\{e \in HM_{\mathrm{MW}},\, e \cap V^{k-1}_{\mathrm{LO}} = \emptyset,\\ \text{and } e \cap e^k \neq \emptyset\}}} |e| \frac{W(e^k)}{|e^k|} \leq \min\{K, \kappa\} W(e^k). \end{aligned} \tag{32}$$

Thus, (32) gives

$$W(e^k) \geq \max\left\{\frac{1}{K}, \frac{1}{\kappa}\right\} \times W\left(\{e | e \in HM_{\mathrm{MW}},\, e \cap V^{k-1}_{\mathrm{LO2}} = \emptyset,\ \text{and}\ e \cap e^k \neq \emptyset\}\right). \tag{33}$$

Adding both sides of (33) and (29) with $i = k-1$, we find (29) is still true for $i = k$. Therefore, (29) is true for any i, and the theorem is proved. □

Theorem 5: Both Algorithm 1 and Algorithm 2 stabilize the system for any rate vector $\vec{x}$ such that $\vec{x} + \epsilon \in \max\{\frac{1}{K}, \frac{1}{\kappa}\}\mathbf{\Lambda}$ for an arbitrarily small $\epsilon \succ 0$.

Proof: We only show the stability of Algorithm 1. Algorithm 2 can be shown similarly. Let $\vec{x}$ be any rate vector such that $\vec{x} + \min\{K, \kappa\}\epsilon \in \mathbf{\Lambda}$. Therefore, there exist flow variables $\check{g}^{mst}_{iJj}$, $\check{f}^m_{iJ}$ and rate variable $\check{r}_{i,J}$ such that the constraints in (7) are all satisfied with $x^{ms} = \check{x}^{ms} + \epsilon$ for an arbitrarily small $\epsilon \succ 0$. Let $\vec{\tilde{x}} = \max\{\frac{1}{K}, \frac{1}{\kappa}\}\vec{x}$, $\tilde{g}^{mst}_{iJj} = \max\{\frac{1}{K}, \frac{1}{\kappa}\}\check{g}^{mst}_{iJj}$, $\tilde{f}^m_{iJ} = \max\{\frac{1}{K}, \frac{1}{\kappa}\}\check{f}^m_{iJ}$ and $\tilde{r}_{i,J} = \max\{\frac{1}{K}, \frac{1}{\kappa}\}\check{r}_{i,J}$. Let $Q^{mst}_i(\tau)$ be the amount of session m data queued at node i for source s and sink t at time τ, and μ^{mst}_{iJj} be the rate offered to sink t of session m from source s for destination j over link (i, J). Define the Lyapunov function $L(\underline{Q}) = \sum_{i,m,s,t} (Q^{mst}_i)^2$. Suppose the input rates are $\tilde{x}^{ms}$. By following the same line of proof as in [7], [20], we obtain

$$\begin{aligned} &E\left\{L(\underline{Q}(\tau+T)) - L(\underline{Q}(\tau)) | \underline{Q}(\tau)\right\} \\ \leq& 2T^2 B|\mathcal{N}| - 2T \sum_{i,m,s,t} Q^{mst}_i(\tau) \\ &\times \left[E\left\{\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} \mu^{mst}_{iJj} - \sum_{j\in\mathcal{N}} \sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} \mu^{mst}_{jIi} \,\middle|\, \underline{Q}(\tau)\right\} - \tilde{\sigma}^{ms}_i\right] \\ \overset{(a)}{=}& 2T^2 B|\mathcal{N}| - 2T \sum_{(i,J),m,t} \sum_{s,j\in J} E\left\{\mu^{mst}_{iJj} | \underline{Q}(\tau)\right\}\left(Q^{mst}_i - Q^{mst}_j\right) \\ &+ 2T \sum_{(i,J),m,t} \sum_{s,j\in J} \tilde{g}^{mst}_{iJj}\left(Q^{mst}_i - Q^{mst}_j\right) - 2T \sum_m |\mathcal{S}_m|\epsilon, \end{aligned} \tag{34}$$

where B is a constant defined in [7]. Note that Q^{mst}_i is a scaled version of q^{mst}_i. Thus, the second term in (a) is equivalent to the objective function in the session scheduling problem (9). Let W_{MW} and W_{LO} be the values of the second term in (a) with maximum weighted hypergraph matching and Algorithm 1, respectively. From Theorem 3, we have $W_{LO} \geq \max\{\frac{1}{K}, \frac{1}{\kappa}\}W_{MW}$. On the other hand, as maximum weighted hypergraph matching solves (9) optimally, we have

$$W_{MW} \geq \sum_{(i,J),m,t} \sum_{s,j\in J} \check{g}^{mst}_{iJj}\left(Q^{mst}_i - Q^{mst}_j\right), \tag{35}$$

where $\check{g}^{mst}_{iJj}$ is also a feasible solution to (9). Multiply both sides of (35) by $\max\{\frac{1}{K}, \frac{1}{\kappa}\}$, we obtain

$$W_{LO} \geq \sum_{(i,J),m,t} \sum_{s,j\in J} \tilde{g}^{mst}_{iJj}\left(Q^{mst}_i - Q^{mst}_j\right). \tag{36}$$

Applying the Lyapunov drift lemma of [10] shows that Algorithm 1 stabilizes the system with rate vector $\vec{\tilde{x}}$ with $\vec{\tilde{x}} + \epsilon \in \max\{\frac{1}{K}, \frac{1}{\kappa}\}\mathbf{\Lambda}$. □

The complexity of algorithms 1 and 2 is also due to that we need to propagate drop e message to all two-hop neighbors of the nodes in e. If we assume that any node can receive drop e message from its two-hop neighbors, the complexity in Theorem 2 can be decreased by a factor of K.

C. Randomized Algorithm

In this subsection, we consider randomized algorithms to find a maximal hypergraph matching.

Definition 2 (maximal hypergraph matching): A hypergraph matching HM is maximal if for each hyperedge $e \in \tilde{\mathcal{H}}$, at least one of the following conditions is satisfied:

- $e \cap HM \neq \emptyset$, i.e., e has non-empty intersection with at least one hyperedge in HM.
- $\tilde{w}_e = 0$, i.e., the number of packets waiting to be transmitted over the hyperedge is zero.

DRHMA: ($\mathcal{G}'$)

1 **for** *each node* $i \in \mathcal{V}$ **do** Set $\Gamma_i = N(i)$;
2 **while** $\exists i$, $\Gamma_i \neq \emptyset$ **do**
3 **for** *each node* $i \in \mathcal{V}$ *and* $\Gamma_i \neq \emptyset$ **do**
4 Let p be a random number generated according to the uniform distribution on $[0, 1]$.
5 **if** $p < \frac{1}{|\Gamma_i|}$ **then**
6 For each node $j \in \Gamma_i$, with probability $\frac{1}{2}$ add j into set S_i;
7 **end**
8 **if** $S_i \neq \emptyset$ **then**
9 Node i decides to transmit and it broadcasts matching messages to all nodes in S_i;
10 Set $E_i = \emptyset$;
11 **end**
12 **end**
13 **for** *each node* $i \in \mathcal{V}$ *and node* i *does not transmit* **do**
14 **if** *node* i *receives matching messages from several neighbors* **then**
15 Node i chooses one of them uniformly at random, say j, and sets $\Gamma_i = \emptyset$;
16 Node i broadcasts a i matched j message;
17 **end**
18 **end**
19 **while** $\exists k$, k *receives a* i *matched* j *message* **do**
20 **if** $k = j$ **then** $E_k = E_k \cup \{i\}$;
21 **else** $\Gamma_k = \Gamma_k - \{i\}$;
22 **end**
23 **for** *each node* $i \in \mathcal{V}$, *and if* i *decides to transmit* **do**
24 **if** $E_i \neq \emptyset$ **then** E_i is added into the hypergraph matching, and set $\Gamma_i = \emptyset$;
25 **end**
26 **end**

Algorithm 3: Distributed randomized hypergraph matching algorithm.

A distributed randomized hypergraph matching algorithm (DRHMA) is given in Algorithm 3. The input of Algorithm 3 is a graph $\mathcal{G}'$, which is obtained after deleting all the directed edges (i, j) with $\max_{m,s,t} \left[q_i^{mst} - q_j^{mst}\right]^+ = 0$ from $\mathcal{G}$. This guarantees that all the hyperedges have positive weights. In Algorithm 3, the set Γ_i keeps track of the set of neighbors of node i that are still not matched, which is initialized to be all its neighbors in $\mathcal{G}$ (line 1). In each time slot, each unmatched node i attempts to transmit with probability $\frac{1}{|\Gamma_i|}$ (line 5). This choice of probability value is similar to that in [24] for the maximal independent sets problem. If i attempts to transmit, for each neighbor j, it sends a matching request to j with probability $1/2$ (line 6). If i sends request to at least one neighbor, i.e., $S_i \neq \emptyset$, it decides to transmit (line 9). E_i denotes the hyperedge to be added into the matching initialized by i (line 10). If node i does not transmit and it receives several matching requests from its neighbors, it chooses one of them uniformly at random, say j, sets $\Gamma_i = \emptyset$ (i is matched), and broadcasts an "i matched j" message (lines 13-18). Upon receiving an "i matched j" message, node k checks whether $k = j$. If $k = j$, this indicates that i got the matching request from k and it would like to join in the hyperedge initialized by k. Thus, k sets $E_k = E_k \cup \{i\}$ (line 20). If $k \neq j$, this indicates that i got matched to j and k should delete i from Γ_k (line 21). For each node i that decides to transmit, if finally $E_i \neq \emptyset$, E_i is added into the hypergraph matching, and we set $\Gamma_i = \emptyset$ (i is matched). Algorithm 3 returns a maximal hypergraph matching.

Theorem 6: The expected running time of Algorithm 3 is $O(\log|\mathcal{E}|)$.

Proof: We first give some definitions. A node $v \in \mathcal{V}$ is *bad* if more than $2/3$ of the neighbors of v are of higher degree than v. A node is *good* if it is not bad. An edge $e \in \mathcal{E}$ is *bad* if both of its endpoints are bad; otherwise the edge is *good*. To show the expected running time of Algorithm 3, we need to show the expected number executions of the **while** loop in Algorithm 3. Let $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ denote the graph after i executions of the while loop, where we only consider those nodes with $\Gamma_v \neq \emptyset$ in $\mathcal{V}_i$.

Let v be a good node of degree $d = |\Gamma_v| > 0$ in $\mathcal{G}_i$ and the neighbors of v be $u_1, \ldots, u_d$. The vertex v has $k \geq \lceil \frac{1}{3}d \rceil$ neighbors such that $d_j = |\Gamma_{u_j}| \leq d$, $j = 1, \ldots, k$. According to Algorithm 3, the probability that node u_j sends a matching request to v is $1/(2d_j) \geq 1/(2d)$. The probability that $u_1, \ldots, u_k$ do not broadcast a matching message to v is

$$\prod_{j=1}^{k}\left(1-\frac{1}{2d_j}\right) \leq \left(1-\frac{1}{2d}\right)^{d/3} < e^{-1/6}. \quad (37)$$

Therefore, the probability that v receives at least one matching message from its neighbors is greater than $1-e^{-1/6} > 0$. Note that ignoring the matching messages from the neighbors with degree greater than d only decreases this probability. Node v responds to received matching messages only when it decides not to transmit, whose probability is $1 - \frac{1}{d} + \frac{1}{d}\frac{1}{2^d}$. It is not hard to show that $1 - \frac{1}{d} + \frac{1}{d}\frac{1}{2^d}$ is an increasing function in d when $d \geq 1$. Therefore, the probability that node v decides not to transmit is at least $\frac{1}{2}$. Finally, node v is included in the hypergraph matching with probability at least $\frac{1}{2}(1 - e^{-1/6})$. The edges incident to v are either included in the hypergraph matching or deleted from $\mathcal{E}_i$. Note that every good edge is incident with at least one good node. According to Lemma 12.6 in [24], at least half the edges in $\mathcal{E}_i$ are good. Thus, the expected number of edges removed from $\mathcal{E}_i$ is at least $\frac{1}{4}(1 - e^{-1/6})|\mathcal{E}_i|$ or

$$E(|\mathcal{E}_i||\mathcal{E}_{i-1}) \leq |\mathcal{E}_{i-1}|(1-\alpha) \Rightarrow E(|\mathcal{E}_i|) \leq |\mathcal{E}|(1-\alpha)^i, \quad (38)$$

where $\alpha = \frac{1}{4}(1 - e^{-1/6})$. Therefore, the expected number executions of the **while** loop in Algorithm 3 is $O(\log|\mathcal{E}|)$. Each **while** loop requires 2 time-slots and the expected running time of Algorithm 3 is also $O(\log|\mathcal{E}|)$. □

Compared with Algorithm 1, Algorithm 3 not only reduces the time complexity from $O(|\mathcal{E}|)$ to $O(\log|\mathcal{E}|)$ but also it does not need to compute the weight of each hyperedge. If we assume that in each session all sinks are only one hop away from the source, by using similar approach as in [25], we can show the following theorem on the performance of the randomized algorithm.

Theorem 7: Algorithm 3 stabilizes the system for any rate vector within $\frac{1}{K}\mathbf{\Lambda}$ if in each session all sinks are only one hop away from the source.

Algorithm 3 can be readily turned into a constant-time algorithm by executing the while loop in Algorithm 3 only M times. We call this algorithm **Algorithm 4**.

Theorem 8: Algorithm 4 with M time-slots stabilizes the system for any rate vector within $\frac{1-(1-\alpha)^M}{K}\mathbf{\Lambda}$.

As maximal matching plays an important role in many scheduling algorithms, see, e.g., [18], [19], we expect that Algorithm 3 can also serve as a basis for other scheduling algorithms for our problem. Note that the approach in [19] cannot be trivially adopted as the connected components in the union of the new hypergraph matching and the old hypergraph matching may be very large. Also, the connected components are not simply cycles or paths as in [19].

D. Hybrid Algorithm

Algorithms 1 and 2 perform well but with high complexity, while Algorithms 3 and 4 have low complexity but with worse performance guarantee as they do not take into account the weight of hyperedge. We next consider combining these two types of algorithms to take advantage of both.

In the hybrid algorithm, we first run Algorithm 1 with T_{th} time slots. To speed up Algorithm 1, we execute the while loop of Algorithm 3 once at the end of T_{th} time slots. We then continue running Algorithm 1. The process continues until there does not exist a node i such that $\Gamma_i \neq \emptyset$. Clearly, if $T_{th} = 0$, the hybrid algorithm reduces to Algorithm 3, while if $T_{th} = \infty$, the hybrid algorithm reduces to Algorithm 1. T_{th} is used to control the tradeoff between complexity and performance. Similarly, Algorithm 2 can also be combined with Algorithm 3. We call this algorithm **Algorithm 5**. In Algorithm 1, each node needs to wait until all its neighbors are included in some local heaviest hyperarc or its neighbors response to the matching request. By running Algorithm 3, each node can directly construct a hyperarc with its neighbors without waiting for its neighbors' decision on the locally heaviest hyperarc. Thus, the hybrid algorithm can accelerate Algorithm 1. The running time of Algorithm 5 is between Algorithm 1 and Algorithm 3. Algorithm 5 also returns a maximal hypergraph matching, and thus Theorem 7 still applies.

Alternatively, we can apply the algorithms in [23] to find a maximum weighted matching first and then add the unmatched nodes into the hypergraph matching randomly.

Remark: Note that in the previous discussion of the scheduling algorithms, we do not consider possible collision of coordinating/signalling messages in carrying out these algorithms. This issue is particularly relevant when we come to the implementation of the scheduling algorithm in real systems. We usually assume to have a separate control channel to do message passing, or divide a time slot into control mini-slots and data slot and message passing happens in control slots. There are basically two ways to coordinate message passing and resolve collision. The first one is to have a "reservation" protocol to pre-specify who to talk and in what order. The second and popular one is to use random access scheme such as Aloha to coordinate message passing over the control channel or mini-slots. We will not elaborate on this, and leave it as an issue to consider in the practical implementation in real systems.

VI. Simulation Results

In this section, we provide numerical examples to complement the analysis in previous sections. We assume that node i's signal power is attenuated by a factor of $\rho_{i,j}^{-2}$ when the signal is received by node j, where $\rho_{i,j}$ is the Euclidean distance between i and j. All nodes have unit signal power and identical noise power 0.02. We assume the use of orthogonal spreading sequences and white Gaussian noise channels and compute r_{iJ} using

$$r_{iJ}(\underline{P}, \underline{S}) = \log\left(1 + \min_{j \in J} \mathrm{SNR}_{i,j}\right), \tag{39}$$

where $\mathrm{SNR}_{i,j} = P_i^{\mathrm{tot}} \frac{|h_{i,j}|^2}{\sigma_j^2}$ is the effective SNR from node i to node j, σ_j^2 is additive white Gaussian noise power at node j, and $h_{i,j}$ is the channel fading coefficient from node i to node j. We have neglected the spreading factor in (39). Two nodes i and j are considered to be connected if and only if the SNR is at least 1 over link (i,j) (i.e., SNR threshold $\lambda = 1$) or the distance between i and j is less than 7.07 meters. We choose log utility function $\log(x)$ for each source in all the experiments.

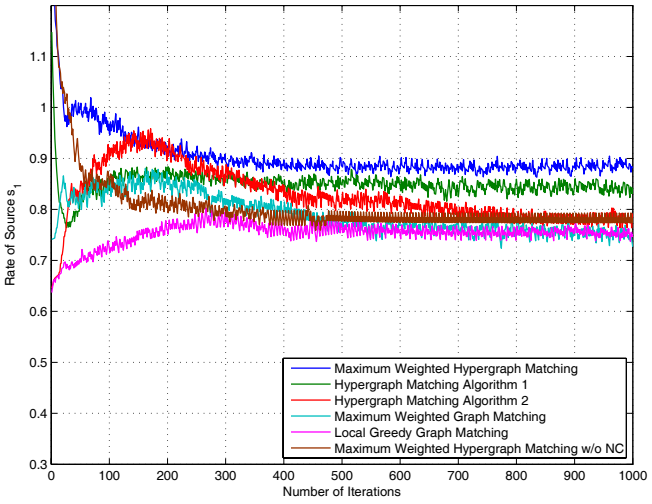


Fig. 2. The evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$ for the network in Fig. 1, where our cross-layer design with maximum weighted hypergraph matching, Algorithm 1 and Algorithm 2, and the algorithm in [16] with maximum weighted graph matching and local greedy matching are compared.

A. Wireless Butterfly Network

We first consider the wireless butterfly network in Fig. 1 with two sources s_1, s_2, two sinks t_1, t_2 and one relay node r. Each source multicasts data to both sinks. We thus only consider a single multicast session. We compare our cross-layer design with different scheduling algorithms in Section V to that in [1] with hypergraph matching but without network coding, that in [16] with maximum weighted matching algorithm in [26] and local greedy matching algorithm in [23]. As the network is small, we also show the performance of our cross-layer design with maximum weighted hypergraph matching by formulating the matching problem as an integer programming and solving it exactly.

Fig. 2 shows the evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$, where our cross-layer design with maximum weighted hypergraph matching, Algorithm 1 and Algorithm 2, the algorithm in [1]

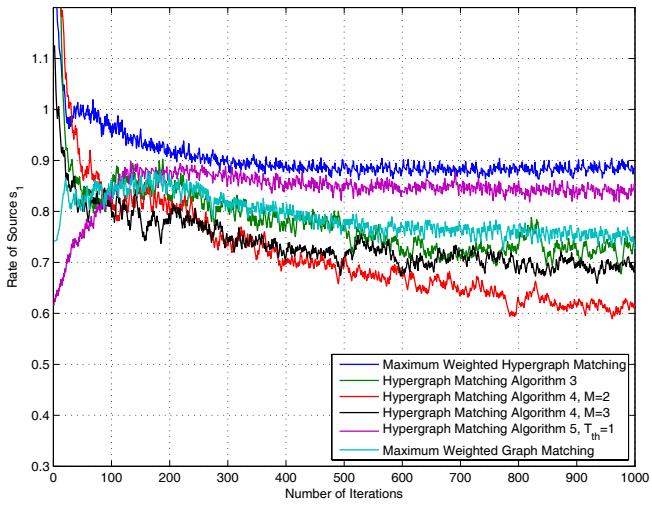


Fig. 3. The evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$ for the network in Fig. 1, where our cross-layer design with maximum weighted hypergraph matching, Algorithm 3, Algorithm 4, and Algorithm 5, and the algorithm in [16] with maximum weighted graph matching.

without network coding, the algorithm in [16] with maximum weighted graph matching and local greedy matching are compared. We observe that the rates of all algorithms converge to within a small neighborhood of the steady values after 500 steps as we have chosen a constant stepsize. Fig. 3 shows the evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$, where our cross-layer design with maximum weighted hypergraph matching, Algorithm 3, Algorithm 4, and Algorithm 5, and the algorithm in [16] with maximum weighted graph matching are compared. Compared with Fig. 2, the rates of Algorithm 3, Algorithm 4, and Algorithm 5 oscillate more severely as all the algorithms use randomized mechanism, which only guarantees the queue size at each node is finite all the time. We quantify the performance of different algorithms in Table I, where HM_{opt} denotes the maximum weighted hypergraph matching, $\text{HM}_{\text{alg}i}$ denotes Algorithm i in Section V, $\text{HM}_{\text{alg4},m}$ denotes Algorithm 4 with m time-slots, $\text{HM}_{\text{alg5},t}$ denotes Algorithm 5 with $T_{th} = t$, $\text{HM}_{\text{w/onc}}$ denotes the hypergraph matching algorithm without network coding in [1], M_{opt} denotes maximum weighted graph matching, and M_{lgd} denotes local greedy graph matching. The first row shows the average rate by averaging the rate of different algorithms in Figs. 2 and 3 from 700th step to 1000th step. Row two shows rate gains of different algorithms over the maximum weighted graph matching.

We can see that our design with broadcast advantage and HM_{opt} has about 17% gain over that without using broadcast advantage. Even with Algorithm 1, about 13% gain can still be achieved. The loss by using Algorithm 3, the randomized algorithm, is only 3.08% gain. A 11.81% gain can be realized by Algorithm 5. The third row compares expected ratio between the weight of different algorithms and that of HM_{opt}. It can be seen that HM_{alg2} has a greater ratio than both M_{opt} and M_{lgd} but they have the same throughput. This indicates that an algorithm that can return a heavier weight does not necessary achieve a higher throughput. Without network coding, the throughput gain over M_{lgd} is small, which is only 3.43%. Row four shows the average number of required time-slots by different algorithms. Surprisingly, both HM_{alg1} and HM_{alg2} require less time-slots than M_{lgd} does, but the former two have higher rates than the latter. This is because the broadcast advantage is exploited during scheduling, where one matching or drop message can reach several nodes. Also note that each hyperedge contains several nodes, which means that nodes are added faster into the hypergraph matching than graph matching. $\text{HM}_{\text{alg5},1}$ has a less number of time-slots than M_{lgd} but with a rate gain.

B. Random Networks

We next show the results on random networks. We assume N nodes are randomly and uniformly placed on a 20 meter by 20 meter square. Both source and sinks are randomly chosen from the 10 nodes. We consider only a single multicast session with one source and various number of sinks.

Tables II-IV show the simulation results with 2, 4, and 6 sinks and $N = 10$ nodes in the network, and table V shows the simulation results with 3 sinks and $N = 15$ nodes in the network. 1000 feasible network realizations are generated. As the number of hyperedges becomes large as the size of network increases, it is hard to find the maximum weighted hypergraph matching by solving the integer programming directly. For comparison purposes, we take a suboptimal approach by computing the linear programming relaxation of the integer program first. In the next phase, we only consider the hyperedges with nonzero solution by the linear programming and solve the integer program with only those hyperedges. This method is denoted as HM_{sub}. From the tables, we can see that even with this suboptimal solution we can achieve a rate gain from 9.45% to 23.47%. Gain increases as the number of sinks increases. The same observation holds for all the other algorithms. On average, Algorithm 1 performs better than Algorithm 2. Algorithm 5 performs close to Algorithm 1 but with reduced complexity. Algorithm 3 has the worst performance but with the lowest complexity among all our proposed algorithms and a comparable throughput as the matching solution. The average number of edges in the connectivity graph is 21.52. The number of time-slots required by Algorithms 1, 2 and 5 is on the order of this number. The locally optimal matching algorithm performs close to the optimal matching algorithm, and has lower complexity than the proposed hypergraph matching algorithms. However, hypergraph matching provides a throughput gain that increases with the number of sinks. Our results suggest that it is more advantageous to use hypergraph matching when the multicast group is large.

VII. Conclusion

We have studied the cross-layer optimization for multicasting in wireless networks with wireless broadcast advantage. By designing distributed approximation algorithms for broadcast link scheduling, we gave fully distributed algorithms for joint rate control, network coding and scheduling. Numerical results have shown promising throughput gain by using the proposed algorithms, and surprisingly, in some cases with

TABLE I
COMPARISON OF DIFFERENT ALGORITHMS IN THE WIRELESS BUTTERFLY NETWORK

	HM_{opt}	HM_{alg1}	HM_{alg2}	HM_{alg3}	$HM_{alg4,2}$	$HM_{alg4,3}$	$HM_{alg5,1}$	$HM_{w/onc}$	M_{opt}	M_{lgd}
Average rate (bits/s)	0.8831	0.8486	0.7972	0.7327	0.6411	0.7062	0.8453	0.7819	0.7594	0.7560
Rate gain over M_{lgd}	0.1681	0.1225	0.0545	-0.0308	-0.1520	-0.0659	0.1181	0.0343	0.0045	0
Average $w/w_{\mathrm{HM}_{opt}}$	1	0.9409	0.8928	0.8765	0.8829	0.8690	0.9250	0.9056	0.8666	0.8578
Average time-slots	-	4	3.9920	4.5500	3.3760	3.8532	5.0300	-	-	5

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS IN RANDOM NETWORKS WITH 10 NODES, 1 SOURCE AND 2 SINKS

	HM_{sub}	HM_{alg1}	HM_{alg2}	HM_{alg3}	$HM_{alg5,1}$	M_{opt}	M_{lgd}
Rate gain over M_{lgd}	9.45%	5.74%	4.32%	-6.46%	5.44%	2.30%	-
Average $w/w_{\mathrm{HM}_{sub}}$	1	0.9749	0.9593	0.8584	0.9714	0.9623	0.9593
Average time-slots	-	13.38	13.46	5.88	9.63	-	6.15

TABLE III
COMPARISON OF DIFFERENT ALGORITHMS IN RANDOM NETWORKS WITH 10 NODES, 1 SOURCE AND 4 SINKS

	HM_{sub}	HM_{alg1}	HM_{alg2}	HM_{alg3}	$HM_{alg5,1}$	M_{opt}	M_{lgd}
Rate gain over M_{lgd}	15.13%	8.78%	4.99%	-1.42%	8.23%	4.49%	-
Average $w/w_{\mathrm{HM}_{sub}}$	1	0.9737	0.9697	0.9310	0.9729	0.9707	0.9697
Average time-slots	-	10.98	11.83	5.90	9.33	-	6.26

TABLE IV
COMPARISON OF DIFFERENT ALGORITHMS IN RANDOM NETWORKS WITH 10 NODES, 1 SOURCE AND 6 SINKS

	HM_{sub}	HM_{alg1}	HM_{alg2}	HM_{alg3}	$HM_{alg5,1}$	M_{opt}	M_{lgd}
Rate gain over M_{lgd}	23.47%	13.05%	5.50%	1.18%	12.15%	6.47%	-
Average $w/w_{\mathrm{HM}_{sub}}$	1	0.9787	0.9776	0.9459	0.9785	0.9760	0.9776
Average time-slots	-	10.40	11.95	5.91	9.10	-	6.57

even lower complexity than the cross-layer design without broadcast advantage. Our results suggest that broadcast link scheduling can be a promising avenue of further research.

REFERENCES

[1] T. Cui, L. Chen, and T. Ho, "Cross-layer design in wireless networks by using broadcast advantage," Caltech, Tech. Rep., Mar. 2007.

[2] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for wireless ad hoc networks," in *Proc. IEEE Infocom*, Mar. 2005.

[3] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Networking*, vol. 14, no. 2, pp. 302-315, Apr. 2006.

[4] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. IEEE Infocom*, 2005.

[5] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition," in *Proc. IEEE*, Jan. 2007.

[6] J. Wieselthier, G. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. Infocom*, Mar. 2000, pp. 585-594.

[7] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," in *Proc. Allerton Conf. Commun., Contr. Comput.*, Sept. 2005.

[8] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 910-917, Sept. 1988.

[9] L. Lovasz and M. Plummer, *Matching Theory*. North Holland, 1986.

[10] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89-103, Jan. 2005.

[11] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proc. IEEE Infocom*, Apr. 2006.

[12] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204-1216, July 2000.

[13] D. S. Lun, M. Médard, and R. Koetter, "Efficient operation of wireless packet networks using network coding," in *Proc. International Workshop Convergent Technol.*, June 2005.

[14] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. Allerton Conf. Commun., Control, Computing*, Sept. 2003.

[15] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2608-2623, June 2006.

[16] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proc. IEEE Infocom*, 2007.

[17] Y. E. Sagduyu and A. Ephremides, "Crosslayer design for distributed MAC and network coding in wireless ad hoc networks," in *Proc. ISIT*, Sept. 2005, pp. 1863-1867.

[18] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radionetworks and input queued switches," in *Proc. Infocom*, Mar. 1998, pp. 533-539.

[19] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," *ACM SIGMETRICS Perf. Evaluation Rev.*, vol. 34, no. 1, pp. 27-38, June 2006.

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

[21] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

[22] N. Z. Shor, *Monimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985.

[23] R. Preis, "Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs," in *Proc. Symp. Theoretical Aspects Computer Science (STACS)*, 1999, pp. 259-269.

[24] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

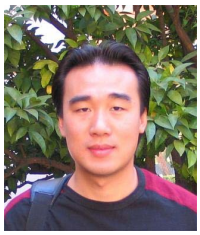
[25] P. Chaporkar, K. Kar, and S. Sarkar, "Fairness and throughput guarantees with maximal scheduling in multi-hop wireless networks," in *Proc. Allerton Conf. Commun., Contr. Comput.*, Sept. 2005.

[26] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 1990, pp. 434-443.

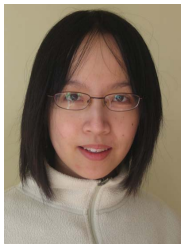
TABLE V
COMPARISON OF DIFFERENT ALGORITHMS IN RANDOM NETWORKS WITH 15 NODES, 1 SOURCE AND 3 SINKS

	HM_{sub}	HM_{alg1}	HM_{alg2}	HM_{alg3}	$HM_{alg5,1}$	M_{opt}	M_{lgd}
Rate gain over M_{lgd}	13.07%	7.79%	3.89%	1.39%	6.45%	4.73%	-
Average $w/w_{HM_{sub}}$	1	0.9813	0.9809	0.9769	0.9544	0.9806	0.9689
Average time-slots	-	15.75	15.41	6.99	11.73	-	7.32

Tao Cui (S'04) received the M.Sc. degree in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, in 2005, and the M.S. degree from the Department of Electrical Engineering, California Institute of Technology, Pasadena, USA, in 2006. He is currently working toward the Ph.D. degree at the Department of Electrical Engineering, California Institute of Technology, Pasadena. His research interests are in the interactions between networking theory, communication theory, and information theory.



Lijun Chen (M'05) received his B.S. from University of Science and Technology of China, M.S. from Institute of Theoretical Physics, Chinese Academy of Sciences and from University of Maryland at College Park, and Ph.D. from Caltech. He is a Research Scientist in the Control and Dynamical Systems Department at Caltech. His current research interests are in communication networks, optimization, game theory and their engineering application.



Tracey Ho (M'06) is an Assistant Professor in Electrical Engineering and Computer Science at the California Institute of Technology. She received a Ph.D. (2004) and B.S. and M.Eng degrees (1999) in Electrical Engineering and Computer Science (EECS) from the Massachusetts Institute of Technology (MIT). Her primary research interests are in information theory, network coding and communication networks.