Distributed Optimization in Wireless Networks Using Broadcast Advantage

Tao Cui, Lijun Chen, and Tracey Ho *Member, IEEE*
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA, USA 91125
Email: {taocui, tho}@caltech.edu, chen@cds.caltech.edu

Abstract— **In this paper, we study cross-layer design for multicasting in wireless networks by exploiting broadcast advantage. With network coding, a distributed subgradient algorithm for joint congestion control, session scheduling, and network coding was proposed, which requires a centralized scheduling algorithm in general. Under the primary interference model, we find that any valid link schedule corresponds to a hypergraph matching. To solve the scheduling problem distributedly, local greedy, randomized, and hybrid algorithms were proposed. We also modify the randomized algorithm into a constant-time algorithm. With random network coding, our obtain a fully distributed cross-layer design. Our experimental results have shown promising throughput gain by using our proposed framework but surprisingly in some cases with less complexity than cross-layer design without broadcast advantage. In the end, we also extend our framework to the case without network coding.**

I. Introduction

Optimization-based cross-layer design for wireless networks has attracted much recent interest (see, e.g., [1]–[5] and the references therein). Joint optimization of multiple protocol layers can substantially increase the end-to-end throughput, or reduce interference and power consumption. Further performance improvements can be achieved using network coding [6].

In this paper we consider distributed joint optimization of multicast network coding, congestion control, packet scheduling and medium access. An important issue relates to the broadcast nature of wireless networks. Most existing work on cross-layer design only takes into account the interference property of the wireless channel. However, when omnidirectional antennas are used, a node's transmissions can be received by any nodes that lie within its communication range. This broadcast advantage can result in power saving and throughput improvement especially with multicasting [7]. Our cross-layer design takes advantage of both network coding and the wireless broadcast advantage. It extends also to the more complicated case of multicasting without network coding.

We assume that a data session is a multicast session; a unicast session is a special case. Our cross-layer design uses the framework of utility maximization. To use broadcast advantage, we model the network as a directed hypergraph. Our framework maximizes the utility subject to the flow conservation on the hypergraph, whose capacity is constrained by the interference model. We then apply duality theory to decompose the problem vertically into congestion control, network coding, and link scheduling subproblems, which interact through dual variables. Based on this decomposition, a distributed subgradient algorithm is proposed, which is similar to the back-pressure algorithm in [8].

For a general interference model, this requires a centralized scheduling algorithm. As such, we study a simpler primary interference model [1], [2]. Under this interference model, we find that 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 NP-complete [9]. We propose three classes of distributed approximation algorithms to solve the link scheduling problem under primary interference model. The first class of algorithms is local greedy algorithm, which chooses the locally heaviest hyperedge. We show this algorithm returns a hypergraph matching with weight at least a constant factor of the maximum weighted hypergraph matching. The second class of algorithms is randomized algorithm, which always returns a maximal hypergraph matching. We show that the rate stability region with randomized algorithm is at least $1/K$ of that with the maximum weighted hypergraph matching, where K is the maximum number of nodes in any hyperedge. The randomized algorithm can be readily turned into a constant-time algorithm. To compromise between the complexity and performance of the first two classes of algorithms, we propose a hybrid algorithm which takes advantage of both. Experimental results show promising performance of our proposed algorithms. Our results in this paper open a new avenue of research in the scheduling problem for wireless networks.

II. Related Work

Extensive research has been devoted to the cross-layer design for wireless networks. Without network coding, joint congestion control and media access control is studied in [1]. Similar cross-layer design algorithm is proposed in [2], where the impact of imperfect scheduling is also studied. In [3], 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.

With network coding, in [10], Lun *et. al.* proposed a dual subgradient method for the problem of minimum cost multicasting routing with network coding. For rate control, the approach in [1] is extended to wireline networks in [11]. In [12], the rate stability region for a wireless network with and without correlated sources is characterized. In [4], 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 works using a time division mechanism, which is clearly suboptimal.

For link scheduling, Hajek and Sasaki [13] introduced the primary interference model. Reference [8] considered stochastic arrivals in general interference models. In the case of primary interference, their algorithm boils down to finding maximum weight matchings. Tassiulas [14] studied randomized algorithms that achieve the capacity region with reduced complexity by comparing a random matching and the current matching. In [15], the authors proposed a distributed implementation of the algorithm in [14] to achieve the entire capacity region. Maximal matching policy is used in [16]. All of the above work only do not use the broadcast advantage.

III. Preliminaries

A. Network Model

Wireless networks are considered in this paper. The 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 hypergraph is a generalization of a graph, where, rather than arcs, we have hyperarcs. A hyperarc is a pair (i, J), where i, the start node, is an element of $\mathcal{N}$, and J, the set of end nodes, is a non-empty subset of $\mathcal{N}$. Each hyperarc (i, J) represents 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 used in [1], [2]. A set of multicast sessions $\mathcal{M}$ 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 of $\mathcal{T}_m \subset \mathcal{N}$ of sinks. In session m, each source $s \in \mathcal{S}_m$ multicasts x^{ms} bits to all the sinks in $\mathcal{T}_m$. By the flow conservation condition, we have

$$\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} g_{iJj}^{mst} - \sum_{j\in\mathcal{N}} \sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst} = \sigma_i^{ms},\ \forall i \in \mathcal{N}, \tag{1}$$

where

$$\sigma_i^{ms} = \begin{cases} x^{ms}, & \text{if } i = s \\ -x^{ms}, & \text{if } i = t \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

and g_{iJj}^{mst} is the flow rate from source $s \in \mathcal{S}_m$ to sink $t \in \mathcal{T}_m$ in session m over hyperarc (i, J) and is intended to node $j \in J$.

Let $\underline{S}(t) = (S_{i,j}(t))$ denote the matrix process of channel states, where $(S_{i,j}(t))$ represents the current state of channel from node i to node j. 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

$$\sum_{(i,J)\in\mathcal{A}} P_{iJ} \le P_i^{\text{tot}},\ \forall i \in \mathcal{N}, \tag{3}$$

where P_i^{tot} is the total power 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 $r_{iJ}(\underline{P}, \underline{S})$ determines the rate at which packets, injected into hyperarc (i, J), are received by all the nodes in J.

B. Network Coding

In conventional packet networks, each node's functions are limited to the forwarding or replication of received packets, while each node is allowed to perform algebraic operations on received packets in network coding. It has been shown that the ability of the network to transfer information can be significantly improved [6]. In this paper, we assume that coding is done only across packets of the same session. With this setting, 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 (1). By the flow sharing property of network coding and rate constraints, we have the following two constraints

$$\sum_{s\in\mathcal{S}_m} \sum_{j\in J} g_{iJj}^{mst} \le f_{iJ}^m, \quad \forall (i,J) \in \mathcal{A},\, m \in \mathcal{M},\, t \in \mathcal{T}_m, \tag{4}$$

$$\sum_{m\in\mathcal{M}} f_{iJ}^m \le r_{iJ}, \qquad \forall (i,J) \in \mathcal{A}, \tag{5}$$

where r_{iJ} belongs to $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$, and the convex hull operator $\text{Co}(\cdot)$ is due to a standard time-averaging argument. To ensure fully distributed cross-layer design, we use distributed random network coding [17]. Interested readers in random network coding are referred to [17].

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

In this section, we derive the cross-layer design framework by using hop-by-hop mechanism. Our algorithms can be readily adapted to end-to-end cross-layer design as in [1], [2], [11]. 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. Our objective is to choose source rates x^{ms} so as to solve the following problem

$$\begin{aligned} \max_{x,g,f,r,P} \ & \sum_{m\in\mathcal{M}, s\in\mathcal{S}_m} U_{ms}(x^{ms}) \\ \text{s.t.} \ & \sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} g_{iJj}^{mst} - \sum_{j\in\mathcal{N}} \sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst} = \sigma_i^{ms}, \\ & \sum_{s\in\mathcal{S}_m,\, j\in J} g_{iJj}^{mst} \le f_{iJ}^m,\ \forall (i,J),\ m,\ t, \\ & \sum_{m\in\mathcal{M}} f_{iJ}^m \le r_{iJ},\ \forall (i,J), \\ & (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})),\ \sum_{(i,J)\in\mathcal{A}} P_{iJ} \le P_i^{\text{tot}},\ \forall i, \end{aligned} \tag{6}$$

where σ_i^{ms} is defined in (2), and the constraints come from (1)-(5). Problem (6) is strictly convex and has a unique solution with respect to source rates x^{ms}. The partial dual function to (6), by relaxing only the first set of constraints in (6), can be decomposed into two subproblems

$$\phi_1(q) = \max_x \sum_{m,s} U_{ms}(x^{ms}) - \sum_{m,s} \left(\sum_t q_s^{mst} \right) x^{ms}, \tag{7}$$

$$\begin{aligned} \phi_2(q) = \max_{g,f,r,P} \sum_{i,m,s,t} q_i^{mst} & \left(\sum_{\{J|(i,J)\in\mathcal{A}\}} \sum_{j\in J} g_{iJj}^{mst} \right. \\ & \left. - \sum_{j\in\mathcal{N}} \sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst} \right), \\ \text{s.t.} \ & \sum_{s,\, j} g_{iJj}^{mst} \le f_{iJ}^m,\ \sum_m f_{iJ}^m \le r_{iJ}, \\ & (r_{iJ}) \in \text{Co}(\underline{r}(\underline{P}, \underline{S})),\ \sum_{(i,J)} P_{iJ} \le P_i^{\text{tot}}, \end{aligned} \tag{8}$$

where q_i^{mst} is the Lagrange multiplier at node i for source s and sink t in session m. The first subproblem is rate control. The second one is the joint network coding and scheduling. Thus, by dual decomposition, the flow optimization problem decomposes into separate "local" optimization problems of transport, and network/data 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 variables $\sum_t q_s^{mst}$ that is generated locally at the source. As utility function is assumed to be strictly concave, we find that

$$x^{ms}(\tau+1) = U_{ms}'^{-1}\left(\sum_t q_s^{mst}(\tau)\right). \tag{9}$$

Session Scheduling and Network Coding: Note that (8) is equivalent to the following problem

$$\begin{aligned}
&\max_{g,f,r,P} \sum_{(i,J),m,t}\sum_{s,j\in J} g_{iJj}^{mst}\left(q_i^{mst}-q_j^{mst}\right),\\
&\quad\text{s.t.} \sum_{s,\,j\in J} g_{iJj}^{mst} \le f_{iJ}^m,\ \sum_m f_{iJ}^m \le r_{iJ},\\
&\quad (r_{iJ}) \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{(i,J)} P_{iJ} \le P_i^{\mathrm{tot}}, \qquad\qquad (10)\\
&= \max_{f,r,P} \sum_{(i,J),m} f_{iJ}^m \sum_t \max_{s,j\in J}\left[q_i^{mst}-q_j^{mst}\right]^+,\\
&\quad\text{s.t.} \sum_m f_{iJ}^m \le r_{iJ},\ (r_{iJ}) \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{(i,J)} P_{iJ} \le P_i^{\mathrm{tot}},
\end{aligned}$$

where $[\cdot]^+$ denotes the projection onto $\mathbb{R}^+$. The last equality in (10) comes from the fact that $\sum_{s,j\in J} g_{iJj}^{mst}\left(q_i^{mst}-q_j^{mst}\right)$, subject to $\sum_{s,\,j\in J} g_{iJj}^{mst} \le f_{iJ}^m$ is a linear programming, we can always choose an extreme point solution, i.e.,

$$g_{iJj}^{mst} = \begin{cases} f_{iJ}^m & \text{if } s=\hat{s}^{mt},\ j=\hat{j}^{mt},\ \text{and } q_i^{mst}-q_j^{mst}\ge 0,\\ 0 & \text{otherwise,}\end{cases} \tag{11}$$

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

For each hyperarc (i,J), let $\hat{m}_{iJ}$ be the multicast session, which has the maximum aggregate differential link prices over the hyperarc, i.e., $\hat{m}_{iJ} = \arg\max_m \sum_t \max_{s,j\in J}\left[q_i^{mst}-q_j^{mst}\right]^+$. 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 (8) by the following assignment

$$g_{iJj}^{mst}(q) = \begin{cases} r_{iJ} & \begin{array}{l}\text{if } m=\hat{m}_{iJ},\ s=\hat{s}^{mt},\ j=\hat{j}^{mt},\\ \text{and } \max_{s,j\in J}\left[q_i^{mst}-q_j^{mst}\right]^+ > 0,\end{array}\\ 0 & \text{otherwise.}\end{cases} \tag{12}$$

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

$$\max_{r,P} \sum_{(i,J)\in\mathcal{A}} w_{iJ} r_{iJ},\ \text{s.t. } (r_{iJ}) \in \mathrm{Co}(\underline{r}(\underline{P},\underline{S})),\ \sum_{(i,J)} P_{iJ} \le P_i^{\mathrm{tot}}. \tag{13}$$

Depending on the rate-power function $r(\cdot,\cdot)$, the joint link scheduling and power control problem (13) is usually a difficult global optimization problem. In some cases, this optimization problem does not have a polynomial-time solution. In Section V, we will discuss a special interference model such that (13) can be solved distributedly in polynomial time.

Dual Variable Update: Let $q(\tau)$ denote the dual variable q at time τ. By the subgradient method [18], each node i updates its dual variable q with respect to source s and sink t in session m according to

$$q_i^{mst}(\tau+1) =$$
$$\begin{cases} q_i^{mst}(\tau) + \gamma_\tau \left(x^{ms}(\tau) - \sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} g_{iJj}^{mst}(q(\tau)) \right. \\ \qquad \left. + \sum_{j\in\mathcal{N}}\sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst}(q(\tau))\right), & \text{if } i=s,\\ q_i^{mst}(\tau) + \gamma_\tau \left(\sum_{j\in\mathcal{N}}\sum_{\{i|(j,I)\in\mathcal{A},\, i\in I\}} g_{jIi}^{mst}(q(\tau)) \right. \\ \qquad \left. - \sum_{\{J|(i,J)\in\mathcal{A}\}}\sum_{j\in J} g_{iJj}^{mst}(q(\tau))\right), & \text{otherwise,}\end{cases} \tag{14}$$

where γ_τ is positive stepsize. After node i updates the value of q_i^{mst}, it passes the value $q_i^{mst}(\tau+1)$ to all its neighbors for next time slot rate control, scheduling and network coding. Note that our algorithm (9)-(14) only requires nodes to communicate with neighbors. Our design is a hop-by-hop control mechanism.

By using results on the convergence of the subgradient method [18], we can show that, for constant stepsize, our algorithm is guaranteed to converge to within a small neighborhood of the optimal value as in [1], [2]. We omit the proof here for brevity. We can also extend our algorithm to multicasting without network coding. Due to space limitation, we omit it here.

V. Link Scheduling

In this section, we solve the joint link scheduling and power control problem (13). Motivated by the results of [2] on imperfect scheduling, we only consider approximation algorithms for solving (13) under primary interference model. We say that a system is stable if the queue lengths (or dual variable q) at all nodes remain finite. A rate vector $\vec{x}$ with entries x^{ms} is feasible if there exists a scheduling policy that can stabilize the system with this rate vector. We define the capacity region $\mathbf{\Lambda}$ to be the set of feasible rate vectors $\vec{x}$. We would like to show whether our scheduling policy stabilizes the system for any rate vector from $\gamma\mathbf{\Lambda}$, where $\gamma \in (0,1]$ is a constant determined by our algorithms.

A. Problem Formulation

Under the primary interference model, we assume that each node is equipped with only a single transceiver. Therefore, links that share a common node cannot be active simultaneously. If we further assume that nodes use orthogonal CDMA or FDMA, links that do not share nodes can transmit at the same time. Under this interference model, any feasible schedule in [1], [2] without using broadcast advantage corresponds to a matching. By using the broadcast advantage, (13) reduces to the maximum weighted hypergraph matching[1] problem.

We assume the use of CDMA in the following. The spreading factor or processing gain of the CDMA is G. With the

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

primary interference model, assuming orthogonal spreading sequences and white Gaussian noise channel, the maximum achievable data rate per unit bandwidth on (i, J) is

$$\begin{aligned} r_{iJ}(\underline{P}, \underline{S}) &= \min_{j \in J} \frac{1}{G} \log\left(1 + P_i^{\text{tot}} \frac{|h_{i,j}|^2}{\sigma_j^2}\right) \\ &= \frac{1}{G} \log\left(1 + \min_{j \in J} \text{SNR}_{i,j}\right), \end{aligned} \quad (15)$$

where σ_j^2 is additive white Gaussian noise power at node j, $h_{i,j}$ is the channel fading coefficient from node i to node j, and $\text{SNR}_{i,j} = P_i^{\text{tot}} \frac{|h_{i,j}|^2}{\sigma_j^2}$ is the effective SNR from node i to node j. Note that at each time slot only one hyperarc per node is active. Therefore, any feasible schedule corresponds to a hypergraph matching of the hypergraph $\mathcal{H}$, and the achievable rate set $\underline{r}(\underline{P}, \underline{S})$ corresponds to the set of hypergraph matchings in $\mathcal{H}$. Let $\mathbf{\Pi}$ denote the set of all hypergraph matchings. We represent a hypergraph matching π as an $|\mathcal{A}|$-dimensional rate vector ξ^π

$$\xi_{iJ}^\pi = \begin{cases} r_{iJ}(\underline{P}, \underline{S}), & \text{if } (i, J) \in \pi \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

The achievable rate region $\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ becomes the convex hull of all the matching rate vectors or equivalently

$$\text{Co}(\underline{r}(\underline{P}, \underline{S})) \triangleq \left\{ \mathbf{r} : \mathbf{r} = \sum_{\pi \in \mathbf{\Pi}} \alpha_\pi \xi^\pi,\ \alpha_\pi \geq 0, \sum_{\pi \in \mathbf{\Pi}} \alpha_\pi = 1 \right\}. \quad (17)$$

$\text{Co}(\underline{r}(\underline{P}, \underline{S}))$ in (17) is a polytope. Thus, (13) is maximized at an extreme point, which corresponds to a maximum weighted hypergraph matching in $\mathcal{H}$. Note that there may exist multiple maximizers, but we always pick an extreme point maximizer.

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 is

$$\tilde{w}_e = \max_{\{(i,J) \in \mathcal{A},\, \{i\} \cup J = e\}} w_{iJ} r_{iJ}(\underline{P}, \underline{S}). \forall e \in \mathcal{E}_h, \quad (18)$$

The problem (13) is then equivalent to the maximum weighted hypergraph matching (or maximum weighted set packing) problem on the weighted hypergraph $\tilde{\mathcal{H}}$.

Different from maximum weighted matching problem on graphs which can be computed in polynomial time, 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. When the cardinality of all hyperedges in $\mathcal{E}_h$ is bounded from above by a constant $K \geq 3$, the problem is approximable within $K - 1 + \epsilon$ [19] for any $\epsilon > 0$. However, the algorithm in [19] is hard to be decentralized. We propose three distributed algorithms in the following.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with the same node set as $\mathcal{H}$. There exists an undirected edge $\{i, j\} \in \mathcal{E}$ between node i and node j if and only if each of them can successfully receive from the other, provided no other node in the network transmits at the same time. Physically, node i can hear the signals from all the other nodes in the network, which makes $\mathcal{G}$ a complete graph. However, the capacity of some edges may be very small. We assume that there exists an edge between node i and node j when $\min\{\text{SNR}_{ij}, \text{SNR}_{ji}\} \geq \gamma$, where γ 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 Algorithm

A linear time approximation algorithm for maximum weighted matching is proposed in [20], and its distributed version appears in [21]. Both algorithms in [20], [21] add a locally optimal edge into the matching at each step, while a global greedy algorithm adds a globally optimal edge into the matching, which makes it hard to be decentralized. Motivated by [20], [21], our algorithm adds a locally heaviest hyperedge into the hypergraph matching at each step.

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

```
DLOHMA: ($\mathcal{G}$)
1  for each node $i \in \mathcal{V}$ do
2      Broadcast the set $\{\text{SNR}_{ij} | j \in N(i)\}$ 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
           $(j^*, L^*) = \arg\max_{j \in \Gamma_i \cup \{i\}} \max_{\{L | L \subseteq \Gamma_j^{(i)},\, i \in L\}} w_{jL} r_{jL}(\underline{P}, \underline{S})$,   (19)
       and $w_{jL}$, $r_{jL}$ are defined in (13) ;
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 $\Gamma_i$ and all $\Gamma_j^{(i)}$, $j \in \Gamma_i$;
19                     if $e \cap J^* \neq \emptyset$ then
20                         Find a node set $J^*$ by (19);
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.

The distributed local optimal hypergraph matching algorithm is given in Algorithm 1. In Algorithm 1, the set Γ_i

maintains 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 the neighbor set $\Gamma_j^{(i)}$ for each neighbor j to facilitate the computation of $\tilde{w}_e$ in (19). 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 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 yes, i is the direct neighbor of some nodes in e and i broadcasts drop e message to let i's neighbors (two-hop neighbor of the nodes in e) know that all the nodes in e are matched. If not, some nodes in e are two-hop neighbors of i and we do 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 node i receives a drop e message and node i is in e, node 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 $|J^*|$ 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 26.

In Algorithm 1, we assume that all hyperedges have different weights. If they do not, we can always break the tie by adding a small constant ϵ_e to the weight of hyperedge e (different e has different ϵ_e). For example, we can change w_{iJ} or r_{iJ} by a small constant. By running Algorithm 1 with this weight, we obtain an ϵ approximation algorithm, where ϵ depends on ϵ_e and the number of hyperedges with the same weight. 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.

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

Different from [21] 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 1: 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 can broadcast 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 has at most $(2K+1)(K-1)|\mathcal{E}|$ iterations. In each iteration, we need to perform (19) at most once. We can solve (19) 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 (13) and (15), 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 (13) contains at most $\kappa - 1$ summands. 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 of Algorithm 1 is $O\left(K^3|\mathcal{E}| \sum_{k=1}^{\min\{\kappa,K\}-1} \binom{K-1}{k}\right)$.

On the other hand, Algorithm 1 is a parallel algorithm. We assume that every message takes one time-slot. It is easy to see that at least one locally heaviest hyperedge can be found within 4 time-slots. By removing the nodes in the locally heaviest hyperedge, the number of nodes in $\tilde{\mathcal{H}}$ is reduced at least by two. Therefore, the algorithm takes at most $O(|\mathcal{V}|)$ time-steps. $\square$

By using the definition of locally heaviest hyperedge, we can readily prove the following theorem.

Theorem 2: 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}.

Algorithm 1 always chooses the locally heaviest hyperedge. Some matched nodes may not contribute much to this 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 $\bar{w}_e$. The complexity of Algorithm 2 is identical to that of Algorithm 1.

Theorem 3: 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}.

Theorem 4: Both Algorithm 1 and Algorithm 2 stabilize the system for any rate vector $\vec{x}$ such that $\vec{x} + \epsilon \in \max\{\frac{1}{K}, \frac{1}{\kappa}\}\mathbf{\Lambda}$.

This theorem can be shown by following the proof in [12]. As in [1], [2], we do not consider the issue of medium access, i.e., packet collision, for the scheduling phase. Since the control messages exchanged during this phase are very short, we assume medium access can be resolved as with a backoff mechanism. We note that additional benefit for medium access in the scheduling phase may be obtained by exploiting the broadcast nature of wireless channels. The details are beyond the scope of this paper.

The high complexity in Theorem 1 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 1 can be decreased by a factor of K.

C. Randomized Algorithm

From Section V-B, the time needed to find a locally heaviest hypergraph matching increases linearly with the number of edges in the connectivity graph. In this subsection, we consider randomized algorithms to find a maximal hypergraph matching, which is motivated by the Luby's parallel maximal independent set algorithm [22]. We start with the definition of the maximal hypergraph matching.

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

- $e \cap HM \neq \emptyset$ or e has non-empty intersection with at least one hyperedge in the maximal matching.
- $\tilde{w}_e = 0$ or 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 $i$ 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 algorithm.

The distributed randomized hypergraph matching algorithm is given in Algorithm 3. The input of Algorithm 3 is a graph $\mathcal{G}'$ is after deleting all the edges $\{i, j\}$ with both $\max\limits_{m,s,t} \left[q_i^{mst} - q_j^{mst}\right]^+ = 0$ and $\max\limits_{m,s,t} \left[q_j^{mst} - q_i^{mst}\right]^+ = 0$ from $\mathcal{G}$, which guarantees that all the hyperedges have positive weights. In Algorithm 3, the set Γ_i maintains 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). Each unmatched node i attempts to transmit with probability $\frac{1}{|\Gamma_i|}$ (line 5). If i attempts to transmit, for each neighbor j, it sends a matching request to j with probability $1/2$ (line 6), which is because we would like to pick any hyperedge containing i with equal probability. 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 a "i matched j" message (lines 13-18). On receiving a "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 just indicates that i got matched to j and k should delete i from Γ_k (line 21). For all the nodes that decide 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 according to Definition 2.

Theorem 5: The expected running time of Algorithm 3 is $O(\log |\mathcal{E}|)$.

Proof: We first give some definitions before proving the theorem. 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*. 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

$$\prod_{j=1}^{k} \left(1 - \frac{1}{2d_j}\right) \leq \left(1 - \frac{1}{2d}\right)^{d/3} < e^{-1/6}. \tag{20}$$

Therefore, the probability that v receives at least a 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 a increasing function in d when $d \geq 1$. Therefore, the probability that node v decides not to transmit is at least $\frac{1}{2}$, and it 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 [22], at least half the edges in $\mathcal{E}_i$ are good. Thus, we get

$$E(|\mathcal{E}_i| |\mathcal{E}_{i-1}) \leq |\mathcal{E}_{i-1}|(1 - \alpha) \Rightarrow E(|\mathcal{E}_i|) \leq |\mathcal{E}|(1 - \alpha)^i, \tag{21}$$

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 $O(\log |\mathcal{E}|)$. $\square$

Compared with Algorithm 1, Algorithm 3 not only reduces the time complexity from $O(|\mathcal{E}|)$ to $O(\log |\mathcal{E}|)$ but it also 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 [16], we can show the following theorem.

Theorem 6: Algorithm 3 stabilizes the system for any rate vector from $\frac{1}{K}\mathbf{\Lambda}$.

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**. Again, by using similar approach as in [16], Theorem 7 in the following guarantees the performance of this constant-time algorithm.

Theorem 7: Algorithm 4 with M time-slots stabilizes the system for any rate vector from $\frac{1-(1-\alpha)^M}{K}\mathbf{\Lambda}$.

As in Section V-B, we do not consider packet collision during hypergraph matching. Algorithms 3 and 4 can be readily adapted to this case. By following the approach in the proof of Theorem 5, we can also bound α in (21).

As maximal matching takes an important role in many scheduling algorithms, e.g., [14], [15], we expect that our Algorithm 3 can also serve as a basis for other scheduling algorithms for our problem. For example, the algorithm in [14] extends naturally to our problem, where a random maximal hypergraph matching is generated by Algorithm 3 and schedule is switched to this new matching if and only if it represents a larger weight. It is not hard to show that this algorithm can also achieve the capacity region $\mathbf{\Lambda}$. The main difficulty to implement this algorithm is to compare two hypergraph matchings distributedly. Note that the approach in [15] 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 [15].

D. Hybrid Algorithm

Our experimental results in Section VI indicate that Algorithms 1 and 2 perform well but with high complexity, while Algorithms 3 and 4 have low complexity but with poor performance as they do not take into account the weight of hyperedge. We next combine these two algorithms to take the 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 as **Algorithm 5**. In fact, we can consider that graph $\mathcal{G}$ is gradually partitioned into isolated components by running the hypergraph matching algorithm. Algorithm 1 is applied on each component, which has a running time proportional to the size of the maximum size component. With Algorithm 3, this partitioning process is accelerated. This is why the hybrid algorithm can speed up Algorithm 1. Clearly, the running time of Algorithm 5 is between Algorithm 1 and Algorithm 3. Algorithm 5 also returns a maximal hypergraph matching. Thus, Theorem 6 still holds.

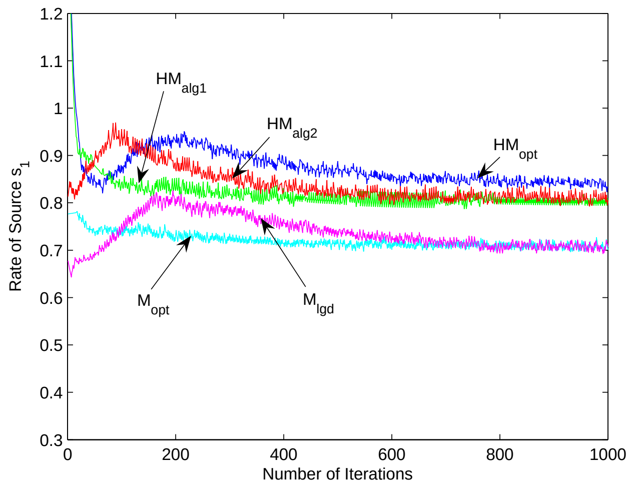

Fig. 1. The evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$ for the wireless butterfly network, where maximum weighted hypergraph matching, Algorithms 1 and 2, and maximum weighted graph matching and local greedy matching are compared.

VI. Simulation Results

In this section, all the nodes are equipped with omnidirectional antennas. The node i's signal power is attenuated by a factor of $\rho_{i,j}^{-\delta}$ when the signal is received by node j, where $\rho_{i,j}$ is the Euclidean distance between i and j, and δ is the path-loss exponent. We set $\delta = 1$. In all our simulations, we assume that all nodes have unity signal power and identical noise power 0.1. We adopt (15) for computing r_{iJ}. We neglect the factor $1/G$ in (15). Two nodes i and j are considered to be connected if and only if link (i,j)'s capacity is at least 1.

We first consider the wireless butterfly network with two sources s_1, s_2, two sinks t_1, t_2 and one relay node r, where the coordinates (in meters) of s_1, s_2, t_1, t_2, r are $(0,5)$, $(5,5)$, $(0,0)$, $(5,0)$, and $(2.5,2.5)$. Each source multicasts data to both sinks. We thus only consider a single multicast session. Fig. 1 shows the evolution of source s_1's rate versus the number of iterations with fixed stepsize $\gamma = 0.01$, where maximum weighted hypergraph matching, Algorithms 1 and 2, maximum weighted graph matching in [23], and local greedy matching in [21] are compared, where the maximum weighted hypergraph matching is obtained by formulating the matching problem as an integer programming and solving it optimally. In Fig. 1, HM_{opt} denotes the maximum weighted hypergraph matching, $\text{HM}_{\text{alg}i}$ denotes Algorithm i in Section V,M_{opt} denotes maximum weighted graph matching, and M_{lgd} denotes local greedy graph matching. We observe that the rates of all algorithms converge within a small neighborhood of the optimal values after 500 steps as we have chosen a constant stepsize. Similar figure is also plotted for Algorithms 3, 4, 5, which is omitted due to lack of space. We have observed that the rates of Algorithms 3, 4, 5 oscillate more severely than Algorithms 1 and 2 as the former algorithms use randomized mechanism, which only guarantees that the queue size at each node is finite all the time. We quantify the performance of different algorithms in Table I, where $\text{HM}_{\text{alg4},m}$ denotes Algorithm 4 with m time-slots, and $\text{HM}_{\text{alg5},t}$

Table I: Comparison of Different Algorithms in the Wireless Butterfly Network.

	$\mathrm{HM}_{\mathrm{opt}}$	$\mathrm{HM}_{\mathrm{alg1}}$	$\mathrm{HM}_{\mathrm{alg2}}$	$\mathrm{HM}_{\mathrm{alg3}}$	$\mathrm{HM}_{\mathrm{alg4},2}$	$\mathrm{HM}_{\mathrm{alg4},3}$	$\mathrm{HM}_{\mathrm{alg5},1}$	$\mathrm{M}_{\mathrm{opt}}$	$\mathrm{M}_{\mathrm{lgd}}$
Average rate (bits/s)	0.8424	0.8030	0.8114	0.7145	0.6104	0.6618	0.8029	0.7061	0.7054
Rate gain over $\mathrm{M}_{\mathrm{opt}}$	19.30%	13.72%	14.91%	1.19%	-13.55%	-6.27%	13.71%	0%	-1%
Expected $w/w_{\mathrm{HM}_{\mathrm{opt}}}$	1	0.9759	0.9763	0.8385	0.6806	0.7638	0.9756	0.7364	0.7290
Expected time-slots	-	4	3.9980	4.6400	3.2480	4.0460	5.0340	-	5

denotes Algorithm 5 with $T_{th} = t$. The first row shows the average rate by averaging the rate of different algorithms in Fig. 1 from 700th step to 1000th step. Row two shows rate gains of different algorithms over the maximum weighted graph matching. Our design with broadcast advantage and $\mathrm{HM}_{\mathrm{opt}}$ has about 20% gain over that without using broadcast advantage. Even with our Algorithms 1 and 2, about 14%-15% gain can still be achieved. Algorithm 3, the randomized algorithm, can also achieve a 1.19% gain. A 13.71% gain can be realized by Algorithm 5. The third row compares expected ratio between the weight of different algorithms and that of $\mathrm{HM}_{\mathrm{opt}}$. Row four shows the expected number of required time-slots by different algorithms. Surprisingly, both $\mathrm{HM}_{\mathrm{alg1}}$ and $\mathrm{HM}_{\mathrm{alg2}}$ require less time-slots than $\mathrm{M}_{\mathrm{lgd}}$ does, but the former two have higher rates than the latter. This is due to the use of the broadcast advantage during scheduling and that each hyperedge contains several nodes. $\mathrm{HM}_{\mathrm{alg5},1}$ requires almost the same time-slots as $\mathrm{M}_{\mathrm{lgd}}$. By changing M in Algorithm 4, we see a trade-off between complexity and performance. Even though Algorithm 4 has from 6% to 14% rate losses, they have lower and constant complexity, and can handle packet collision easily.

We next consider randomly generated networks. 10 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 2, 4, and 6 sinks. 1000 feasible network realizations are generated. We observe that maximum weighted hypergraph matching can achieve rate gains 11.86%, 14.07%, and 16.33% for 2, 4, and 6 sinks. Gain increases as the number of sinks increases. The same observation holds for all the other algorithms. On average, Algorithm 2 performs better than Algorithm 1 as the former takes into account the size of the hyperedge. Our results suggest that it is better to use hypergraph matching when the multicast group is large.

VII. Conclusion

We proposed a cross-layer design framework for multicasting in wireless networks by exploiting broadcast advantage. A distributed subgradient algorithm for joint congestion control, scheduling and network coding was proposed, which requires a centralized scheduling algorithm in general. Under the primary interference model, we find that any valid link schedule corresponds to a hypergraph matching. Distributed local greedy, randomized, and hybrid algorithms were proposed. Our experimental results have shown promising throughput gain by using our framework over that without broadcast advantage but surprisingly in some cases with even lower complexity. We expect more benefit when power cost is also considered. It is also interesting to investigate the achievable rate ratio between multicasting with and without broadcast advantage. For complete proof of theorems and more simulation results, please refer to our technical report [24].

References

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

[2] 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, April 2006.

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

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

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

[6] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

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

[8] 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.

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

[10] 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. Inform. Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.

[11] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Rate control for multicast with network coding," to appear in *IEEE Infocom*, 2007.

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

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

[14] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radionetworks and input queued switches," in *Proc. of Infocom*, March 1998, pp. 533–539.

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

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

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

[18] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[19] E. M. Arkin and R. Hassin, "On local search for weighted packing probems," *Math. Oper. Res.*, vol. 23, pp. 640–648, 1998.

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

[21] J.-H. Hoepman, "Simple distributed weighted matchings," eprint cs.DC/0410047, Oct. 2004.

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

[23] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *Proc. of ACM-SIAM Symposium on Discrete algorithms*, 1990, pp. 434–443.

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