Joint Congestion Control and Distributed Scheduling for Throughput Guarantees in Wireless Networks

GAURAV SHARMA
D. E. Shaw & Co., L.P.
CHANGHEE JOO and NESS B. SHROFF
The Ohio State University
and
RAVI R. MAZUMDAR
University of Waterloo

We consider the problem of throughput-optimal cross-layer design of wireless networks. We propose a joint congestion control and scheduling algorithm that achieves a fraction $1/d_I(G)$ of the capacity region, where $d_I(G)$ depends on certain structural properties of the underlying connectivity graph G of the wireless network, and also on the type of interference constraints. For a wide range of wireless networks, $d_I(G)$ can be upper bounded by a constant, independent of the number of nodes in the network. The scheduling element of our algorithm is the maximal scheduling policy. Although this scheduling policy has been considered in several previous works, the challenges underlying its practical implementation in a fully distributed manner while accounting for necessary message exchanges have not been addressed in the literature. In this paper, we propose two algorithms for the distributed implementation of the maximal scheduling policy accounting for message exchanges, and analytically show that they still can achieve the performance guarantee under the 1-hop and 2-hop interference models. We also evaluate the performance of our cross-layer solutions in more realistic network settings with imperfect synchronization under the signal-to-interference-plus-noise ratio (SINR) interference model, and compare with the standard layered approaches such as TCP over IEEE 802.11b DCF networks.

Categories and Subject Descriptors: C.2.1 [**Computer-communication networks**]: Network architecture and design; C.4 [**Performance of systems**]: Design studies; I.6.4 [**Simulation and modeling**]: Model validation and analysis

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Cross-layer design, wireless communication systems simulation and modeling: general, distributed algorithm, maximal scheduling

1. INTRODUCTION

Wireless networks have become a ubiquitous part of all modern day communication systems. Unlike wireline networks, where bandwidth and other resources are

plentiful, wireless networks are highly resource constrained, thus underscoring the need for efficient utilization of the wireless resources. A seminal contribution in this direction was made by Tassiulas and Ephremides [1992b]. They characterized the *capacity region* of constrained queuing systems, such as a wireless network, and developed a queue length based scheduling scheme that is *throughput-optimal*, i.e., it stabilizes the network provided the user rates fall within the capacity region of the network, where the capacity region is defined to be the set of user arrival rates under which the network is stable (i.e., average queue lengths of all the links are kept finite; See Definition 3 for a formal definition).

Unlike wireline networks, where each link has a fixed capacity, the capacity of a wireless link varies with channel variations due to fading; changes in power allocation, link scheduling, or routing; and changes in network topology, etc. These types of dependencies make the traditional layered design approaches less effective in the case of wireless networks and demand a new cross-layer design approach, which has spurred the recent interest in developing cross-layer optimization algorithms for wireless networks (see, for example, [Xiao et al. 2004], [Neely et al. 2003], [Tassiulas and Ephremides 1992a]).

Motivated by the work of Kelly et al. [1998] on fair resource allocation in wireline networks, researchers have incorporated congestion control and link scheduling into the cross-layer optimization framework. The congestion control component determines the rates at which users inject data into the network so as to ensure that they fall within the capacity region of the network, and the scheduling component decides which links should be active at what time and in what sequence to accommodate the rates allocated by the congestion control. In this case, it turns out that the most complex part of cross-layer optimization is the scheduling that has to solve a very difficult global optimization problem of the form:

$$\begin{aligned} &\textbf{maximize} \sum_{l\in\mathcal{L}} q_l r_l \\ &\textbf{subject to } \mathbf{r} \in \Delta, \end{aligned} \tag{1}$$

where $\mathcal{L}$ denotes the set of wireless links, $\mathbf{r}$ is the vector of rates r_l of link $l \in \mathcal{L}$, q_l is the congestion price or possibly some function of the backlog at link $l \in \mathcal{L}$, and Δ is the stability region of the network. The main difficulty in solving the problem is that the stability region Δ depends on the complete network topology and has no simple representation in terms of the power constraints on the individual links or nodes.

A scheduling scheme is said to be *throughput-optimal* if it stabilizes the network for all rate vectors strictly within Δ. A throughput-optimal scheduling scheme has been known, but it requires a centralized coordinator with high computational complexity, and is, in general, NP-Hard. Since distributed algorithms are highly preferred in ad hoc settings, a considerable amount of effort has been put forth in devising the simple distributed schemes that can achieve a certain fraction of the stability region.

To this end, the scheduling problem has been studied under several special cases of particular interest, e.g., under a simplified interference model with no power control, including the node-exclusive (or primary) interference model, which can be used for

Bluetooth and FH-CDMA networks (see [Hajek and Sasaki 1988] and [Sarkar and Tassiulas 2005]), and the IEEE 802.11 DCF type (or secondary) interference model (see [Chaporkar et al. 2005] and [Wu et al. 2007]). The works of [Sharma et al. 2006a] and [Sharma et al. 2006b] considered the two models as an instance of the class of K-hop interference models, under which no two links within a K-hop distance can transmit simultaneously. It captures some of the essential properties of the signal-to-interference-plus-noise ratio (SINR) based interference model and includes the node-exclusive model as the 1-hop interference model and the IEEE 802.11 DCF type as the 2-hop interference model. By increasing K, one can model even more stringent interference constraints. It has been known that the scheduling problem (1) is polynomial time solvable under the 1-hop interference model, but it is NP-Hard and Non-Approximable[1] under K-hop interference models for $K > 1$. [Sharma et al. 2006b] showed that the optimal solution can be approximated within a constant factor in a geometric network graph, such as unit-disk graph and (r, s)-civilized graph. These results are encouraging as a wide variety of wireless networks can be modeled using these families of graphs.

There are several recent works in the literature that propose cross-layer optimization solutions with provable throughput guarantees. Sarkar and Tassiulas [2005] developed a distributed congestion control scheme with centralized scheduling to achieve maxmin fair end-to-end performance. Stolyar [2005] reformulated the cross-layer throughput-optimality problem to a utility maximization problem, and Neely et al. [2008] extended the idea to time-varying channels. Bui et al. [2006] has shown that using regulator, a distributed scheduling algorithm can operate in conjunction with an asynchronous congestion control scheme. Lin and Shroff [2005] studied the cross-layer interactions with the imperfect scheduling scheme that achieves a fraction of the optimal weighted rate sum (1). Eryilmaz et al. [2009] generalized the notion of imperfectness in routing, scheduling, and congestion control algorithms.

Due to high complexity involved in the scheduling component, *maximal scheduling policy*[2] has been widely studied in the context of the cross-layer design since it is amenable to implement in a distributed manner. A schedule is said to be *maximal* if it cannot add a link without violating the underlying interference constraints. The maximal scheduling policy chooses a maximal schedule at every time slot. Wu et al. [2007] have studied the performance of the maximal scheduling policy under a joint congestion control and scheduling framework with multi-hop traffic in a static setting. Lin and Shroff [2005] have extended the results to the case that users arrive departure dynamically. Chaporkar et al. [2005] and Sharma et al. [2006b] considered a slightly more restrictive setting of single-hop traffic and no congestion control under the matrix based interference model and the K-hop interference model, respectively. Other distributed scheduling policies have been also developed to improve the throughput performance with low complexity. They include Greedy Maximal Scheduling studied by McKeown [1995] and Hoepman [2004], Pick-and-Compare type scheduling by Tassiulas [1998] and Modiano et al.

[1]A problem is said to be Non-Approximable if it does not admit any constant factor polynomial time approximation algorithm.

[2]Note that the terminologies used in these works and some minor details of the schemes differ slightly from each other, but the main idea is essentially the same.

[2006], and constant-time random access scheduling by Lin and Rasool [2006] and Joo and Shroff [2007]. However, these scheduling schemes suffer from lack of the corresponding congestion control counterpart that can deliver the performance improvement of scheduling to the end user.

Most previous cross-layer solutions are limited to specific interference models (i.e., the 1-hop or 2-hop interference model), or require that the scheduling component guarantee a fraction of the optimal value of (1), which is stronger than guarantee a fraction of throughput performance. Moreover, they require information exchanges among neighboring links and nodes, which is often assumed to be provided. However, in practice, these message exchanges are non-trivial especially under the presence of wireless interference, and becomes a major obstacle for practical use.

In this paper, we propose a cross-layer solution for the joint congestion control and scheduling problem that achieves provable throughput guarantees under general interference models, and provide randomized and fully distributed implementations of the maximal scheduling policy taking into account message exchanges. Next, we highlight the main contributions of this paper.

—We propose a cross-layer solution based on the maximal scheduling policy under a multi-hop setting with *general interference constraints described by a matrix*, and show that it achieves a fraction $1/d_I(G)$ of the capacity region, where $d_I(G)$ denotes the largest number of links that can be activated simultaneously in the interfering neighborhood of a link (see Definition 2). It extends earlier results in [Chaporkar et al. 2005] and [Sharma et al. 2006b] under a single-hop (MAC layer) setting with no congestion control.

—We develop randomized distributed algorithms that *fully account for control message exchanges* in wireless settings for the maximal scheduling policy under 1-hop and 2-hop interference models. They have $\Theta(\delta \log^2 |V|)$ and $\Theta(\delta^2 \log^2 |V|)$ complexity, respectively, for scheduling computation and local message exchanges, where $|V|$ is the number nodes in the network and δ is the maximum node degree. We show that the proposed practical algorithms still achieve the same performance guarantee.

—We evaluate the performance of our cross-layer solution through simulations. We show that our algorithms work well with a small amount of overhead, and that the performance can improve further if more information on the queue lengths is available at the scheduling component. We also simulate our cross-layer solution under more realistic settings, i.e., *under an asynchronous system with SINR-based interference model*, and clarify its advantage over the conventional layered solutions such as TCP over IEEE 802.11b DCF.

The rest of the paper is organized as follows. We describe our system model in Section 2. The joint solution of congestion control and scheduling is developed with analysis providing a lower bound on the performance in Section 3. We develop randomized distributed algorithms for the maximal scheduling policy accounting for message exchanges in Section 4. We numerically evaluate our joint cross-layer solution under realistic network settings, and compare with the layered solution of TCP over IEEE 802.11b DCF in Section 5. Finally, we present concluding remarks in Section 6.

2. SYSTEM MODEL AND RELATED WORK

We consider a set V of nodes, labeled $1, 2, ..., |V|$, communicating with each other using wireless means. We say that link (u, v) joining node u to node v exists if node u can successfully transmit to node v, provided no other node in the network transmits at the same time. The set of *undirected* links (or edge) so formed is denoted by E. We consider a time-slotted system where a single frequency is used for the whole system. Then due to wireless interference, two links cannot transmit simultaneously if they interfere with each other. We model the interference constraints using a contention matrix $C := [C_{ij}]_{i,j\in E}$, where $C_{ij} = 1$ if link i interferes with link j, and $C_{ij} = 0$ otherwise. If link i interferes with link j, it cannot be scheduled with link j at the same time. All diagonal entries of C are set to 1. The contention matrix based interference model captures some of the essential properties of the SINR-based interference model and includes several interference models studied in the literature as special cases (see, for example, [Chaporkar et al. 2005], [Sharma et al. 2006a], [Sharma et al. 2006b], [Lin and Shroff 2005], [Lin et al. 2008], [Wu et al. 2007], and [Bui et al. 2006]), which can be obtained by imposing certain constraints on the matrix. The only constraint we impose on $[C_{ij}]$ in the paper is that it should be symmetric, i.e., $C_{ij} = C_{ji}$ for all $i, j \in E$. We denote a schedule by a set $\mathcal{S} \in \{0, 1\}^{|E|}$ of link, where $|\cdot|$ denote the cardinality of the set. We have $\mathcal{S}_l = 1$ if link l is active and $\mathcal{S}_l = 0$ if it is not. Slightly abusing the notation, we also denote the set of active links by $\mathcal{S}$. A schedule is said to be *feasible* if it does not violate the interference constraints defined by C. We denote the set of all feasible schedules by $\mathbb{S} := \{\mathcal{S}^1, \mathcal{S}^2, \dots\}$.

We assume that link l can transmit at rate c_l during a time slot if no other interfering link transmits at the same time slot. Note that in wireless environments, the link rate may depend on many factors (e.g., noise variance at the receiving node, coding and modulation scheme used by the nodes), but in certain network scenarios, e.g., wireless mesh networks, wireless channels are quite stable and the link rates do not change frequently. However, the system dynamics across layers are still very complex due to wireless interference. Note that our model can be also incorporated with techniques dealing with time-varying channels, for which we refer the interested reader to [Neely et al. 2003] and [Lin et al. 2008].

We consider $\mathcal{K}$ types of users, labeled $1, 2, \dots, \mathcal{K}$, sending data via multiple hops over the network. We assume that type k users arrive into the network according to a Poisson process with rate λ_k. Each user of type k, brings with it a file, whose size is exponentially distributed with mean $1/\mu_k$. We assume that all users of any given type send their data over a predetermined loop-free route. The extension to multi-route case is straightforward and discussed in Section 3.2. The user routes are stored in an incidence matrix $[H_k^l]$, where $H_k^l = 1$ if link l belongs to the route of type k users; and 0 otherwise.

Let $\vec{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_{\mathcal{K}})$ be the vector of user arrival rates. Let $n_k(t)$ and $Q_l(t)$ denote the number of type k users and queue backlog at link l in the network at time t, respectively. As in [Lin and Shroff 2005] and [Neely et al. 2003], we say that the network is *stable* if

$$\limsup_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{1}_{\left\{\sum_{k=1}^{\mathcal{K}} n_k(t) + \sum_{l\in E} Q_l(t) > \eta\right\}} dt \to 0, \text{ as } \eta \to \infty, \tag{2}$$

where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function. The *capacity region*[3] of the network is defined as the set of user arrival rate vectors, for which the network can be stabilized by some scheduling policy. Tassiulas and Ephremides [1992b] characterized the capacity region of a constrained queuing system, such as a wireless network. For our model, the capacity region Ω can be presented as

$$\Omega := \left\{ \vec{\lambda} \;\middle|\; \left[\sum_{k=1}^{\mathcal{K}} \frac{H_k^l \lambda_k}{\mu_k c_l} \right]_{l \in E} \in Co(\mathbb{S}) \right\}, \tag{3}$$

where $Co(\mathbb{S})$ denotes the convex hull of $\mathbb{S}$, i.e.,

$$Co(\mathbb{S}) := \{ \textstyle\sum_i w_i \mathcal{S}^i \mid \mathcal{S}^i \in \mathbb{S}, w_i \geq 0, \sum_i w_i = 1 \}.$$

In the following, we derive an upper bound on the capacity region under the contention matrix based interference model. The bound will be used to analyze the performance of the maximal scheduling policy. We start with some definitions related to the underlying network graph.

We define the *interference set* of link e as the set of links that interfere with link e, and denote by $I(e) := \{l \in E \mid C_{el} = 1\}$. Note that $e \in I(e)$ from $C_{ll} = 1$ for all $l \in E$. We define the interference degree of a link as follows.

DEFINITION 1. *The* interference degree $d_I(e)$ of link e *is the maximum number of links can be active at the same time, i.e.,*

$$d_I(e) := \max_{\mathcal{S} \in \mathbb{S}} |\mathcal{S} \cap I(e)|.$$

Note that since the singleton $\{e\} \in \mathbb{S}$, we have $d_I(e) \geq 1$ for all $e \in E$. We now characterize the upper bound of the capacity region.

THEOREM 1. *All user arrival rate vectors $\vec{\lambda}$ within the capacity region Ω specified by (3) must satisfy*

$$\sum_{l \in I(e)} \sum_{k=1}^{\mathcal{K}} \frac{H_k^l \lambda_k}{\mu_k c_l} \leq d_I(e) \text{ for all } e \in E.$$

PROOF. Let us consider a link $e \in E$ and a feasible schedule $\mathcal{S}$ that contains e. Since all links in $\mathcal{S}$ do not interfere with each other, $\mathcal{S}$ contains at most $d_I(e)$ links from $I(e)$. Thus, the link rate vectors $[x_l]_{l \in E}$ under $\mathcal{S}$ must satisfy that

$$\sum_{l \in I(e)} \frac{x_l}{c_l} \leq d_I(e) \text{ for all } e \in E. \tag{4}$$

Note that the constraint (4) holds for all feasible link schedules $\mathcal{S} \in \mathbb{S}$, and thus, any feasible link rate vectors in $Co(\mathbb{S})$ must also satisfy the constraint (4) due to the convexity. Since a user arrival rate vector $\vec{\lambda} \in \Omega$ induces a feasible link rate vector $[\sum_{k=1}^{\mathcal{K}} H_k^l \lambda_k / \mu_k]_{l \in E}$ from (3), the result follows. □

[3]It is worth noting the difference between the capacity region Ω and the stability region Δ in (1). In brief, the stability region Δ can be interpreted as the capacity region with single-hop traffics only.

Using the following definition, we can also obtain a network-wide bound on the capacity region from Theorem 1.

DEFINITION 2. *The* interference degree $d_I(G)$ of graph $G(V,E)$ *is the maximum interference degree across its constituent links, i.e.,*

$$d_I(G) := \max_{e \in E} d_I(e). \tag{5}$$

COROLLARY 1. *All user arrival rate vectors* $\vec{\lambda}$ *that belongs to the capacity region* Ω *must satisfy that*

$$\sum_{l \in I(e)} \sum_{k=1}^{\mathcal{K}} \frac{H_k^l \lambda_k}{\mu_k c_l} \leq d_I(G) \text{ for all } e \in E.$$

3. JOINT CONGESTION CONTROL AND SCHEDULING FOR THROUGHPUT GUARANTEES

3.1 Algorithm

We now propose a joint congestion control and scheduling (CCS) algorithm that is guaranteed to achieve a fraction $1/d_I(G)$ of the capacity region under the contention matrix based interference model. Each link $l \in E$ maintains *congestion price* $q_l(t)$ to estimate the level of congestion at time t. Both congestion control and scheduling algorithms make use of these congestion prices. Time is divided into slots of unit duration. Congestion prices and user rates are updated at the beginning of each time slot.

The proposed algorithm is similar in spirit to the joint congestion control and scheduling algorithm proposed by [Lin and Shroff 2005] under the 1-hop interference model. However, our algorithm is significantly more general and works under the contention matrix based interference model, which includes the 1-hop interference model. To this end, we have to involve more information to control the user rates and use different congestion prices. The detailed description of the algorithm is as follows.

CCS Algorithm:

—*Congestion price update:* The congestion prices are updated as

$$q_l(t+1) = \left(q_l(t) + \alpha \Delta q_l(t)\right)^+, \tag{6}$$

where $(q)^+ := \max\{q, 0\}$ represents a projection, and

$$\Delta q_l(t) = \sum_{j \in I(l)} \left(\sum_{k=1}^{\mathcal{K}} H_k^j \int_t^{t+1} \frac{n_k(t) x_k(t)}{c_j} dt - \mathbb{1}_{\{j \in \mathcal{S}(t)\}} \right), \tag{7}$$

where $\mathcal{S}(t)$ denotes the set of links scheduled to transmit during time slot t. Note that the congestion price is strongly connected to backlogs of links since $\sum_{k=1}^{\mathcal{K}} H_k^j \int_t^{t+1} \frac{n_k(t)x_k(t)}{c_j} dt - \mathbb{1}_{\{j \in \mathcal{S}(t)\}} =: \Delta Q_j(t)$ is the change of backlog of link j during $[t, t+1]$. Different from the previous work by [Lin and Shroff 2005], the

congestion price of link l depends not only on the backlog of link l, but also on the backlogs of its neighboring links $j \in I(l)$. Hence, if the network is stabilized and the backlogs converge, we will have $q_l(t) = \alpha \sum_{j \in I(l)} Q_j(t)$. Clearly, the price update can be implemented in a distributed manner.

—*User rate update:* The data rate of type-k users are updated as

$$x_k(t+1) = \min \left\{ \frac{1}{\sum_{l \in E} q_l(t+1) \sum_{j \in I(l)} \frac{H_k^j}{c_j}}, M_k \right\}, \tag{8}$$

where M_k is the maximum data rate of type-k users. The data rate of type-k users depends on the congestion prices of all the links l that either belong to the route of type-k users (i.e., $H_k^l = 1$) or interfere with such a link (i.e., $H_k^j = 1$ for some $j \in I(l)$).

—*Transmission scheduling:* The link transmissions are scheduled in accordance with the maximal scheduling policy. Specifically, at each time slot t, the set $\mathcal{S}(t)$ of links chosen for transmission satisfies that for all links $l \in E$, either $I(l) \cap \mathcal{S}(t) \neq \emptyset$ or $q_l(t) \leq 1$. For the sake of concreteness, we assume that link l can be scheduled when its congestion price $q_l(t)$ is greater than 1. Clearly, a schedule that satisfies the condition is maximal, and no other link l with $q_l(t) > 1$ can be added to $\mathcal{S}(t)$ without violating the interference constraints.

These settings of the user rates and congestion prices allow CCS to achieve a fraction $1/d_I(G)$ of the capacity region, as stated in the following theorem:

THEOREM 2. *If the stepsize α is chosen to be small enough, CCS stabilizes the network for all user arrival rate vectors $\vec{\lambda} \in \Omega^o/d_I(G)$, where Ω^o denotes the interior of the capacity region Ω.*

PROOF. We use the Lyapunov technique to show the system stability as Georgiadis et al. [2006] and Neely et al. [2003]. In queueing processes that evolve according to ergodic Markov chains, if there is a Lyapunov function whose drift is negative whenever the queue state is outside of a bound region of the state space, the Markov chain has a well defined steady state, and the system is stable.

The main difficulty is to construct an appropriate Lyapunov function that exhibits the required negative drift for all $\vec{\lambda} \in \Omega^o/d_I(G)$. We shall use the Lyapunov function

$$V(\vec{n}, \vec{q}) := V_n(\vec{n}) + V_q(\vec{q}), \text{ where } V_n(\vec{n}) := \sum_{k=1}^{\mathcal{K}} \frac{\beta n_k^2}{2\lambda_k}, \text{ and } V_q(\vec{q}) := \sum_{l \in E} \frac{q_l^2}{2\alpha}.$$

In the following, we show that the Lyapunov function has a negative drift, i.e., $\mathbb{E}[V(\vec{n}(t+1), \vec{q}(t+1)) - V(\vec{n}(t), \vec{q}(t)) \mid \vec{n}(t), \vec{q}(t)] < 0$ for a large $\vec{n}(t)$ and $\vec{q}(t)$. Then, the result follows using Theorem 2 of [Neely et al. 2003].

Let $\Delta V_q := \mathbb{E}[V_q(\vec{q}(t+1)) - V_q(\vec{q}(t))|\vec{n}(t), \vec{q}(t)]$, and $\Delta V_n := \mathbb{E}[V_n(\vec{n}(t+1)) - V_n(\vec{n}(t))|\vec{n}(t), \vec{q}(t)]$. Also, let $\mathbb{E}_t[\cdot]$ denote the conditional expectation $\mathbb{E}[\cdot|\vec{n}(t), \vec{q}(t)]$ for notational convenience. We first obtain intermediate results for ΔV_q and ΔV_n, and then combine them. For ΔV_q, since all scheduled links l must have a congestion

price $q_l > 1$, the projection operator in (6) can be removed if $\alpha \leq 1$, and we have

$$\begin{aligned}\Delta V_q &= \sum_{l\in E} \mathbb{E}\left[q_l(t)\Delta q_l(t) + \frac{\alpha}{2}(\Delta q_l(t))^2 \mid \vec{n}(t), \vec{q}(t)\right] \\ &\leq \sum_{l\in E} q_l(t) \sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{H_k^j}{c_j} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)x_k(\tau)\right] d\tau \\ &\quad - \sum_{l\in E} q_l(t)\mathbb{1}_{\{q_l(t)>1\}} + \frac{\alpha}{2}\mathbb{E}_t\left[(\Delta q_l(t))^2\right].\end{aligned} \tag{9}$$

For the last term of (9), we can obtain the following from (7) after some algebraic manipulations:

$$\mathbb{E}_t[(\Delta q_l(t))^2] \leq |E| + \theta_a\theta_b \int_t^{t+1} \mathbb{E}_t\left[n_k^2(\tau)x_k^2(\tau)\right] d\tau, \tag{10}$$

where $\theta_a := \max_{l\in E} \sum_{k=1}^{\mathcal{K}} \sum_{j\in I(l)} \frac{H_k^j}{c_j}$, and $\theta_b := \max_{k=1,2,\ldots,K} \sum_{l\in E} \sum_{j\in I(l)} \frac{H_k^j}{c_j}$. Combining (9) and (10), we get

$$\begin{aligned}\Delta V_q \leq{}& \frac{\alpha\theta_a\theta_b}{2} \sum_{k=1}^{\mathcal{K}} \int_t^{t+1} \mathbb{E}_t\left[n_k^2(\tau)x_k^2(\tau)\right] d\tau + \frac{\alpha}{2}|E| - \sum_{l\in E} q_l(t)\mathbb{1}_{\{q_l(t)>1\}} \\ &+ \sum_{l\in E} q_l(t) \sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{H_k^j}{c_j} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)x_k(\tau)\right] d\tau.\end{aligned} \tag{11}$$

For ΔV_n, we can follow the line of analysis used by Lin and Shroff [2005]. That is, using the user rate update (8), we can obtain that

$$\begin{aligned}\Delta V_n \leq{}& \beta \sum_{l\in E} q_l(t) \sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{\lambda_k H_k^j}{\mu_k c_j} \\ &- \sum_{l\in E} q_l(t) \sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{H_k^j}{c_j} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)x_k(\tau)\right] d\tau \\ &- (\beta-1) \sum_{k=1}^{\mathcal{K}} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)\right] d\tau + \theta_1 \\ &- \sum_{k=1}^{\mathcal{K}} \frac{\mu_k}{4\lambda_k M_k} \int_t^{t+1} \mathbb{E}_t\left[n_k^2(\tau)x_k^2(\tau)\right] d\tau,\end{aligned} \tag{12}$$

where θ_1 is a constant, which results from bounded change of the congestion prices and the user rates during a time slot. We omit the detailed derivations and refer the interested reader to the proof of Theorem 7 of [Lin and Shroff 2005].

Now, we add (11) and (12). Since the third term of (11) can be bounded by

$\sum_{l\in E} q_l(t)\mathbb{1}_{\{q_l(t)>1\}} \geq \sum_{l\in E} q_l(t) - |E|$, we have that

$$\begin{aligned}\Delta V_n + \Delta V_q \leq & \sum_{k=1}^{\mathcal{K}} \left(\frac{\alpha\theta_a\theta_b}{2} - \frac{\mu_k}{4\lambda_k M_k}\right) \int_t^{t+1} \mathbb{E}_t\left[n_k^2(\tau)x_k^2(\tau)\right] d\tau \\ & + \sum_{l\in E} q_l(t) \left(\beta \sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{\lambda_k H_k^j}{\mu_k c_j} - 1\right) + \left(1+\frac{\alpha}{2}\right)|E| \\ & - (\beta-1)\sum_{k=1}^{\mathcal{K}} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)\right] d\tau + \theta_1.\end{aligned}$$

Note that in the second term, we have $\sum_{j\in I(l)} \sum_{k=1}^{\mathcal{K}} \frac{\lambda_k H_k^j}{\mu_k c_j} < 1$ for all $l \in E$ and any $\lambda \in \Omega^o/d_I(G)$ from Corollary 1. Hence, given any $\epsilon > 0$, we can choose a small $\alpha(>0)$ and a $\beta(>1)$ close to 1 such that

$$\Delta V_n + \Delta V_q \leq \theta_2 - \epsilon\left(\sum_{l\in E} q_l(t) + \sum_{k=1}^{\mathcal{K}} \int_t^{t+1} \mathbb{E}_t\left[n_k(\tau)\right] d\tau\right), \tag{13}$$

where $\theta_2 = \theta_1 + \left(1+\frac{\alpha}{2}\right)|E|$. □

Theorem 2 provides a lower bound on the performance of CCS in wireless networks under arbitrary (symmetric) interference constraints.

In the following, we provide an example analysis for the performance bounds of CCS by applying Theorem 2 to more restricted network settings, i.e., a geometric network where connectivity between nodes is determined by geometric properties. If all transmissions in the network employ the same power level and the statistical properties of the noise are the same at each node, then the connectivity graph of the network is indeed a geometric graph. Let us restrict our attention to the K-hop interference models, which enforce further structure on the graph.

Proposition 1. *In a wireless network, whose connectivity graph is a geometric graph and the interference constraints are described by the K-hop interference model for some $K \geq 1$, CCS can stabilize the network for all user arrival rate vectors $\vec{\lambda} \in \frac{1}{2}\Omega^o$ for $K = 1$, and for $\vec{\lambda} \in \frac{\lfloor K/2\rfloor^2}{(2K+1)^2}\Omega^o$ for $K \geq 2$ by choosing the stepsize α sufficiently small.*

Proof. Sharma et al. [2006b] have showed that if the underlying connectivity graph is a geometric graph and interference constraints correspond to the K-hop interference model, then the interference degree of the graph satisfies that $d_I(G) \leq 2$ for $K = 1$ and $d_I(G) \leq \frac{(2K+1)^2}{\lfloor K/2\rfloor^2}$ for $K \geq 2$. Then, the result directly follows from Theorem 2. □

Similar results can be obtained for disk graphs and (r, s)-civilized graphs.

3.2 Implementation issues and extensions

In this section, we identify some implementation issues related to CCS including local message exchanges, time synchronization, multi-scale decomposition, multi-route extension, and distributed implementation. We discuss how they can be dealt with.

The congestion price update in (6) requires each link to track the changes in the backlogs at all its interfering links from the summation over $I(l)$. This can be done with the help of local message exchanges: Each link l floods relevant information (e.g., whether the link was active in the previous time slot) within its local neighborhood including all links $j \in I(l)$. However, under an arbitrary contention matrix based interference model, in which two interfering links can be several hops away from each other, the congestion price update would incur a considerable amount of overhead. In practice, however, the scope of interference is limited to a small number of hops, typically 1 or 2. In such settings, the overhead due to congestion price update is expected to be small. A detailed discussion of this issue in the context of specific interference models can be found in Section 4.

Further, the congestion price update assumes that a user rate change is instantaneously informed to all links on its route. In practice, however, there is some delay for the information to propagate to the links on the route. In addition, if user clocks are not perfectly synchronized with each other, their rate changes would be asynchronous. Bui et al. [2006] have dealt with both these issues by introducing a regulator at each link, in the context of a joint congestion control and scheduling under the 1-hop interference model. It is straightforward to extend the techniques to our case. The regulators smooth incoming traffic to queues, and make the packet arrivals in a good shape. Since they effectively average the rates, we can utilize stochastic tools as before to design the cross-layer system achieving the same performance guarantee.

The rate update for type-k users requires the knowledge of congestion prices of all the links $j \in I(l)$ for each link l on the route. Note that this kind of behavior is common to many end-to-end congestion control protocols, e.g., TCP, and can be seen even under a wireline setting. In view of the results by Yaiche et al. [2000], Low and Lapsley [1999] and Bui et al. [2006], one would expect that the stability properties of CCS should be preserved even when the rate updates are performed at a much slower time scale as compared to the congestion price updates. This time-scale decomposition issue will be addressed in our future work.

Our results can be also extended to multi-route case. Note that (8) is a solution to the optimization problem that maximizes $U_k(x_k) - \sum_{l \in E} q_l \sum_{j \in I(l)} \frac{H_k^j x_k}{c_j}$ for $0 \leq x_k \leq M_k$, where $U_k(x_k)$ is the logarithmic utility function. The idea for multi-route extension is to divide the traffic belongs to a type into multiple routes and solve a modified optimization problem with an addition of a quadratic form of rates. The new term (i.e., the quadratic form of rates) helps the traffic assignment to be a continuous function across time. The resultant solution will be still amenable to implement in a distributed manner with local information. The techniques have been developed by Tassiulas and Ephremides [1992a] and further exploited by Neely et al. [2003] and Lin et al. [2008].

Another important issue is distributed implementations of the scheduling component of CCS. Generating a maximal schedule accounting for the local message exchanges might be computationally expensive under the contention matrix based interference model. The problem becomes more complex if two end nodes of a link have a different view on the network. To this end, we provide in the next section two randomized distributed algorithms that are described by nodes' behavior, and

that compute a maximal schedule accounting for message exchanges. Assuming a time-slotted system, we show that the proposed distributed implementations for the maximal scheduling still achieve the provable performance guarantee with low complexity. Note that the assumption on the system synchronization is for the analytical purpose. In Section 5, we show through simulations that in realistic network environments, i.e., with imperfect synchronization and under the SINR-based interference model, the proposed cross-layer solution significantly outperforms the layered solution like TCP over IEEE 802.11b network.

4. RANDOMIZED DISTRIBUTED ALGORITHMS FOR MAXIMAL SCHEDULING POLICY

The maximal scheduling policy has been studied in [Lin and Shroff 2005], [Chaporkar et al. 2005], and [Sharma et al. 2006b]. However, the difficulties that arise in its distributed implementation (i.e., local message exchanges and operations based on the node behavior) under the presence of interference have not previously been dealt with. In this section, we provide randomized distributed algorithms of the maximal scheduling policy under the 1-hop and 2-hop interference models. They are inspired by the classical algorithm by [Peleg 2000] for constructing a maximal independent set.

We start with describing our distributed computing model. As in [H. Balakrishnan et al. 2004] and other related works, we assume a *synchronous message passing distributed computing model*: We describe the network $G(V, E)$ as a set V of nodes and a set E of *directed*[4] links (or edges), where a link is placed between two nodes if one can make a direct transmission to the other. We assume that links are bidirectional (i.e., if $(u, v) \in E$, then $(v, u) \in E$) and the interference between links can be specified by the K-hop interference model for $K = 1$ or 2. The clocks at all the nodes are synchronized, and time is divided into slots of unit length. A transmission from node u can be heard by each node v with $(u, v) \in E$, unless node v or some of v's neighbors transmits at the same time. Hence, a node can broadcast a packet to all its connected neighbors, but the interference between simultaneous transmissions prevents the packet from successfully being delivered, and becomes a major obstacle in the implementation of the maximal scheduling policy.

A time slot consists of two periods for scheduling and data transmission, respectively. The scheduling period is used to choose a set of non-interfering links, and the data transmission period is used to transmit data packets over the chosen links. The scheduling period is further divided into mini-slots. We assume that the transmission of control message takes place in rounds, each occupying a mini-slot. We focus on the scheduling period and develop a distributed algorithm that yields a maximal schedule $\mathcal{S}(t)$ at each time slot t (i.e., for all links $l \in E$, the schedule $\mathcal{S}(t)$ satisfies that $I(l) \cap \mathcal{S}(t) \neq \emptyset$ or $q_l(t) < 1$).

4.1 Maximal scheduling under the 1-hop interference model

We first propose a randomized distributed algorithm, namely MaxScheduleOneHop, that implements the maximal scheduling policy under the 1-hop interference

[4]We use directed links for price estimation of a link at each node. Once all the prices are calculated, the behavior of each node can be described using undirected links.

model. MaxScheduleOneHop has two levels of iterations. We denote the lower level by *iteration* and the higher level by *phase*, respectively, and label an instance as iteration i of phase p.

We start with some definitions of terminology that will be used in the sequel. New notations are node-based and deal with direct links. Though some of them are conceptually similar to the previously defined (link-based) notations, they will greatly simplify algorithm descriptions and performance analysis.

—$N(u)$: Set of nodes v directly connected with node u, i.e., $N(u) := \{v \in V \mid (u,v) \in E\}$. Note that $u \notin N(u)$, and the set denotes the connected neighboring nodes.

—$d(u) := \max_{v \in N(u) \cup \{u\}} |N(v)|$.

—$\delta := \max_{u \in V} |N(u)|$.

—$\mathcal{S}_p^i$: Set of links that are included in the schedule *before* iteration i of phase p. Slightly abusing the notation, we also denote it as the set of nodes that are connected to one of those links. The meaning should be clear by context. Note that this set monotonically grows as iterations and phases proceed.

—$\mathcal{S}_p$: Set of links (or nodes) *after* all iterations in phase p. After all the phases end, the final value will be used as the schedule $\mathcal{S}(t)$ for data transmission.

—$b(u)$: Indicator that denotes whether node u is eligible for a schedule or not, i.e., at iteration i of phase p, $b(u) = 1$ if $u \in \mathcal{S}_p^i$, and $b(u) = 0$ otherwise.

—$N_p(u)$: Knowledge of node u about its neighborhood during phase p. Specifically, $N_p(u) = \{v \in N(u) \mid b(v) = 0 \text{ at the beginning of phase } p, \text{ and } q_{(u,v)}(t) > 1\}$. This information is updated at the end of each phase p (which implies that each node u obtains its local knowledge about $\mathcal{S}_p$), and fixed during the next phase.

We assume that the information $N(u)$, $d(u)$, and δ has been known to each node u after initial network setup. The congestion price of a directed link $l = (u,v)$ can be defined as in (6) and (7), where $I(l)$ denotes the set of all directed links that interfere with the link. Then it is clear that $q_{(u,v)}(t) = q_{(v,u)}(t)$ from (7).

The algorithm of MaxScheduleOneHop is provided in Algorithm 1, which has two subroutines: UpdatePrices and UpdateAndDistNeighborhoods. At the beginning of the procedure, **UpdatePrices** allows each node to update the congestion prices of its outgoing links. At the end of each phase p, **UpdateAndDistNeighborhoods** allows each node to update its local knowledge on the schedule.

Detailed operations are as follows. At each time slot t, MaxScheduleOneHop calculates new prices based on the schedule in the previous time slot. Then, it has two hierarchical loops to compute the schedule in a distributed fashion: $\lceil \mathbb{C}_P \log |V| \rceil$ phases, and $\mathbb{C}_I \delta \log |V|$ iterations for each phase, where $\mathbb{C}_P$ and $\mathbb{C}_I$ are constants whose value will be determined later (see Proposition 2). At each iteration i, each node u tries to transmit an Request-To-Send (RTS) message with probability based on $d(u)$ if the node is eligible (i.e., if $b(u) = 0$). If the RTS is responded with a Clear-To-Send (CTS) by the receiver v, then both u and v are set to be ineligible (i.e., $b(u) \leftarrow 1, b(v) \leftarrow 1$), and the link (u,v) is included in the schedule $\mathcal{S}_p^i$. When phase p ends, all nodes u update their local knowledge $N_p(u)$ based on the change of the schedule $\mathcal{S}_p \backslash \mathcal{S}_{p-1}$ through UpdateAndDistNeighborhoods. When all phases

end, if node u has a link $(u, v) \in \mathcal{S}(t)$, it transmits data packets to v during the data transmission period.

Algorithm 1 MaxScheduleOneHop$(G, \vec{q}(t-1), \mathcal{S}(t-1))$

1: $\vec{q}(t) \leftarrow$ **UpdatePrices**$(G, \vec{q}(t-1), \mathcal{S}(t-1))$
2: $\mathcal{S}_0 \leftarrow \phi$ and $b(u) \leftarrow 0$ for all $u \in V$
3: Compute $N_1(u)$ for all $u \in V$
4: **for** $p = 1$ to $\lceil \mathbb{C}_P \log |V| \rceil$ **do**
5: $\quad \mathcal{S}_p^1 \leftarrow \mathcal{S}_{p-1}$
6: $\quad$ **for** $i = 1$ to $\mathbb{C}_I \delta \log |V|$ **do**
7: $\quad\quad$ Each node u with $b(u) = 0$ chooses to transmit with probability $1/(d(u)+1)$. Upon deciding to transmit, it chooses a node v at random from $N_p(u)$ and sends an RTS message to node v.
8: $\quad\quad$ If node v successfully receives the RTS, it responds with a CTS message and sets $b(v) \leftarrow 1$
9: $\quad\quad$ Upon receiving the CTS, set $b(u) \leftarrow 1$
10: $\quad\quad \mathcal{S}_p^{i+1} \leftarrow \mathcal{S}_p^i \cup (u, v)$
11: $\quad$ **end for**
12: $\quad \mathcal{S}_p \leftarrow \mathcal{S}_p^{i+1}$
13: $\quad N_{p+1}(u) \leftarrow$ **UpdateAndDistNeighborhoods**$(G, \mathcal{S}_{p-1}, \mathcal{S}_p)$ for all $u \in V$
14: **end for**
15: $cs(t) \leftarrow \mathcal{S}_p$

Note that during each phase p, each node u has only local information of $\mathcal{S}_p$, i.e., whether $v \in N_p(u)$ or not. This also implies that, if node u transmits an RTS to node v at iteration i of phase p, it may fail due to i) simultaneous transmission of neighboring nodes (i.e., some node $w \in N(u) \cup N(v) \backslash \{u\}$ transmits an RTS simultaneously), or ii) inaccuracy of scheduling information (i.e., $v \notin \mathcal{S}_p$ but $v \in \mathcal{S}_p^i$). Later we consider both scenarios carefully in our analysis. We first describe the subroutines used by MaxScheduleOneHop, beginning with UpdatePrices.

Algorithm 2 UpdatePrices$(G, \vec{q}(t-1), \mathcal{S}(t-1))$

For each $u \in V$,
1: **if** $(u, v) \in \mathcal{S}(t-1)$ or $(v, u) \in \mathcal{S}(t-1)$ for some $v \in N(u)$ **then**
2: $\quad$ **ReliablyBroadcast**$(u,$ "matched to v")
3: **end if**
4: Compute the new congestion prices of outgoing links $q_{(u,w)}(t)$ for all $w \in N(u)$ based on the local knowledge of $\mathcal{S}(t-1)$.

UpdatePrices allows each node to compute the congestion prices of its outgoing links by extending the local knowledge of the schedule $\mathcal{S}(t-1)$. Specifically, each node u is included in the schedule $\mathcal{S}(t-1)$, during slot t, broadcasts this information to its neighbors using subroutine ReliablyBroadcast, which will be explained in a short while. Then, node u and all its neighbors can compute new congestion price for all their own outgoing links using this information and their local knowledge of user routes and data rates. For example, under the 1-hop interference model,

we can compute the prices as $q_{(u,v)}(t) = (q_{(u,v)}(t-1) + \alpha\Delta q_{(u,v)}(t-1))^+$, where $\Delta q_{(u,v)}(t)$ can be rewritten from (7) as

$$\Delta q_{(u,v)}(t) = \sum_{w\in N(u)} \Delta X_{(u,w)}(t) + \sum_{w\in N(v)\backslash\{u\}} \Delta X_{(v,w)}(t),$$

$$\text{and } \Delta X_{(u,w)}(t) := \sum_{k=1}^{\mathcal{K}} H_k^{(u,w)} \left(\int_t^{t+1} \frac{n_k(\tau)x_k(\tau)}{c_{(u,w)}} d\tau - \mathbb{1}_{\{(u,w)\in\mathcal{S}(t)\}} \right) + \sum_{k=1}^{\mathcal{K}} H_k^{(w,u)} \left(\int_t^{t+1} \frac{n_k(\tau)x_k(\tau)}{c_{(w,u)}} d\tau - \mathbb{1}_{\{(w,u)\in\mathcal{S}(t)\}} \right).$$

Note that for all $u, v \in V$, we have $\Delta X_{(u,v)}(t) = \Delta Q_{(u,v)}(t) + \Delta Q_{(v,u)}(t) = \Delta X_{(v,u)}(t)$, and hence, $\Delta q_{(u,v)}(t) = \Delta q_{(v,u)}(t)$. Node u can compute $\Delta q_{(u,v)}(t)$ for each of its outgoing links (u,v), provided that it has the local knowledge of user routes $[H_k^{(u,v)}]$, data rates $[x_k]$, and scheduling information during the previous time slot $\mathcal{S}(t)$. The information of user routes and data rates are maintained by the congestion control component, and the scheduling information can be provided by the following subroutine ReliablyBroadcast.

The subroutine ReliablyBroadcast requires $\lceil \mathbb{C}_B \delta \log |V| \rceil$ rounds, where $\mathbb{C}_B$ is a constant whose value will be determined later (see Lemma 1).

Algorithm 3 ReliablyBroadcast(v, data)

1: **for** $k = 1$ to $\lceil \mathbb{C}_B \delta \log |V| \rceil$ **do**
2: Node u broadcasts "data" with probability $1/(d(u)+1)$ to its neighbors.
3: **end for**

Finally, UpdateAndDistNeighborhoods allows each node u to update $N_p(u)$ when a phase of MaxScheduleOneHop ends (see Algorithm 4). If node u is matched during the current phase p (i.e., if $u \in \mathcal{S}_p \backslash \mathcal{S}_{p-1}$), it broadcasts this information to its neighbors using the subroutine ReliablyBroadcast. With this knowledge, each node w can update $N_p(w)$ by deleting those neighboring nodes included in the schedule during the current phase, i.e., $N_{p+1}(w) \leftarrow N_p(w) \backslash \{\mathcal{S}_p \backslash \mathcal{S}_{p-1}\}$, which is equal to $N_1(w) \backslash \mathcal{S}_p$.

Algorithm 4 UpdateAndDistNeighborhoods(G, S_{pr}, S_{cr})

For each $u \in V$,
1: **if** $(u,v) \in S_{cr} \backslash S_{pr}$ or $(v,u) \in S_{cr} \backslash S_{pr}$ for some $v \in N(u)$ **then**
2: **ReliablyBroadcast**(u, "matched to v")
3: **endif**
4: Update $N_p(u)$ considering only those nodes that are currently not scheduled.

We now show that if the constants $\mathbb{C}_P$, $\mathbb{C}_I$, and $\mathbb{C}_B$ in MaxScheduleOneHop and ReliablyBroadcast are appropriately chosen the scheduling algorithm returns a maximal schedule with high probability. We begin with the performance analysis of ReliablyBroadcast:

LEMMA 1. *Using ReliablyBroadcast with $\mathbb{C}_B \geq 8e$, all nodes can forward its "data" to all its own neighbors with probability no smaller than $1 - \frac{1}{|V|^2}$.*

The proof is given in the supplement for this article [Sharma et al. 2009].

Using Lemma 1, we obtain the performance of MaxScheduleOneHop as follows.

PROPOSITION 2. *If* $\mathbb{C}_P > 1$, $\mathbb{C}_I \geq 12e^2$, *and* $\mathbb{C}_B \geq 8e$, *then the schedule of MaxScheduleOneHop is maximal with probability no smaller than* $1 - \frac{2}{|V|}$.

The proof is given in the supplement for this article [Sharma et al. 2009].

Note that MaxScheduleOneHop has $\Theta(\log |V|)$ phases, and each phase has $\Theta(\delta \log |V|)$ iterations. UpdateAndDistNeighborhoods that runs at the end of each phase also involves $\Theta(\delta \log |V|)$ rounds of computation and local message exchange. Hence, MaxScheduleOneHop needs total $\Theta(\delta \log^2 |V|)$ rounds of computation and local message exchange.

Remark: If the maximum node degree in the network is significantly smaller than $|V| - 1$, then one can reduce the number of phases in MaxScheduleOneHop to $\lceil \mathbb{C}_P \log \delta \rceil$. MaxScheduleOneHop would then require $\Theta(\delta \log \delta \cdot \log |V|)$ rounds of computation and local message exchange. For example, if the maximum node degree in the network is $\Theta(\log |V|)$, as in the case of random geometric graphs, then by reducing the number of phases in MaxScheduleOneHop from $\Theta(\log |V|)$ to $\Theta(\log \log |V|)$, we can reduce its running time from $\Theta(\log^3 |V|)$ to $\Theta(\log^2 |V| \cdot \log \log |V|)$.

4.2 Maximal scheduling under 2-hop interference model

In this section, we propose a randomized distributed algorithm for implementing the maximal scheduling policy under the 2-hop interference model. The interference model now requires that no two links within a two-hop distance can transmit simultaneously. However, we assume the same distributed computing model that we adopt in Section 4.1 for the control messages. In particular, control messages can be broadcasted and can be heard by each (one-hop) neighboring node v, unless node v or some of v's neighbors transmits at the same time. The reason for having a different interference model for data and control packets is that in most networks (e.g., IEEE 802.11b-based networks) the control packets are usually much smaller in size than the data packets and are often transmitted at a much smaller rate than the data packets. Correspondingly, successful reception of a control packet requires much less SINR as compared to that of a data packet; thereby motivating the use of different interference models.

The algorithm we propose in this section, namely MaxScheduleTwoHop, is conceptually very similar to MaxScheduleOneHop. However, there are some additional difficulties that arise due to the 2-hop interference, and have to be dealt with. Indeed, a distinguishing feature of MaxScheduleTwoHop is the exchange of *collision* (COL) messages to ensure that no two links within two-hop distance decide to transmit or receive at the same time. To elaborate, if a sender detects an ongoing transmission while transmitting an RTS, it sends a subsequent COL immediately. Successful reception of an RTS by the receiver guarantees that no other transmitter can be within one-hop distance from the receiver. Further, no following COL message guarantees interference-free transmission at the sender side. If the receiver does not hear a COL or a collision due to multiple such messages, it sends a CTS message. If it detects an ongoing transmission while transmitting the CTS, it subsequently sends a COL. Then the sender can ensure that the transmission is interference-free

at the receiver side if no COL has been heard after the CTS. Let $N'(u)$ denote the set of nodes u within two-hop distance, i.e., $N'(u) := \cup_{v \in N(u) \cup \{u\}} N(v) \backslash \{u\}$. Also let $d'(u) := \max_{v \in N'(u)} |N'(v)|$. We provide the detailed description in Algorithm 5.

Algorithm 5 MaxScheduleTwoHop$(G, q(t))$

$\vec{q}(t) \leftarrow$ **UpdateAndDistPrices**$(G, \vec{q}(t-1), \mathcal{S}(t-1))$
$\mathcal{S}_0 \leftarrow \phi$ and $b(u) \leftarrow 0$ for all $u \in V$.
Compute $N_1(u)$ for all $u \in V$
for $p = 1$ to $\lceil \mathbb{C}_P \log |V| \rceil$ **do**
  $\mathcal{S}_p^1 \leftarrow \mathcal{S}_{p-1}$
  for $i = 1$ to $\mathbb{C}_I \delta^2 \log |V|$ **do**
    Each node u with $b(u) = 0$ chooses to transmit with probability $1/(d'(u)+1)$. Upon deciding to transmit, it chooses a node v at random from $N_p(u)$ and sends an RTS to node v.
    If node u detects any other transmission while transmitting, then it sends a COL immediately after the RTS.
    If node v successfully receives the RTS and does not subsequently hear a COL (or a collision due to multiple such messages), then it sends a CTS message.
    If node v detects any other transmission while transmitting the CTS, then it sends a COL immediately after the CTS. Otherwise, it sets $b(v) \leftarrow 1$.
    If node u hears the CTS from v but no subsequent COL, it sets $b(u) \leftarrow 1$.
    $\mathcal{S}_p^{i+1} \leftarrow \mathcal{S}_p^i \cup (u, v)$
  end for
  $\mathcal{S}_p \leftarrow \mathcal{S}_p^{i+1}$
  $N_{p+1}(u) \leftarrow$ **UpdateAndDistTwoHopNeighborhoods** $(G, \mathcal{S}_{p-1}, \mathcal{S}_p)$ for all $u \in V$.
end for
$\mathcal{S}(t) \leftarrow \mathcal{S}_p$

The overall operation of MaxScheduleTwoHop can be explained as follows. MaxScheduleTwoHop also has two subroutines: UpdateAndDistPrices and UpdateAndDistTwoHopNeighborhoods, which are very similar to their counterparts in case of MaxScheduleOneHop. The main difference, however, is that these subroutines must deliver information over two hops for each node u i) to compute the congestion price of its outgoing links (u, v), which now can be obtained by summing additional $\Delta X_{(x,y)}(t)$ for all $x \in N(u) \cup N(v)$ and $y \in \{N'(u) \cup N'(v)\} \backslash \{N(u) \cup N(v)\}$, and ii) to compute $N_{p+1}(u)$, which now requires the knowledge of $\mathcal{S}_p^i$ within two-hop distance. Broadcasting information over two-hop distance can be accomplished by using ReliablyBroadcast twice. Detailed descriptions of these algorithms with the same spirit can be found in [Sharma et al. 2006].

5. PERFORMANCE EVALUATION

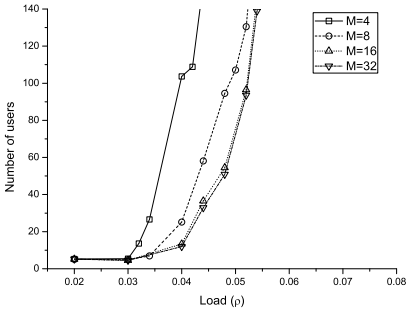
In this section, we evaluate the performance of the cross-layer solution of the rate controller (8) and randomized distributed scheduling algorithms provided in the previous section. We first simulate the cross-layer solution in a random topology under a restrictive scenario, i.e., with a time-slotted synchronous system under the 1-hop interference model. In this experiment, we show the impact of system parameters on the performance of the solution. Second, we suggest a heuristic scheduling scheme that further exploits the queue length information of neighboring links, and compare the performance of the original randomized algorithm, the heuristic, and the centralized Greedy Maximal Scheduling (GMS) policy. We introduce the heuristic scheduling since it inherits the distributed feature of the original randomized maximal scheduling and at the same time, it can be easily modified for asynchronous systems. Finally, we compare the modified heuristic cross-layer solution and the conventional layered solution (i.e., TCP over IEEE 802.11b DCF) under more realistic scenarios, i.e., with inaccurate synchronization under the SINR-based interference model.

We construct a network topology by placing 20 nodes randomly in 1x1 square area. Links are connected between two nodes if their distance is less than 0.3. Link capacity is set uniformly at random between $[5, 10]$ per time slot[5]. The interference constraints between links are modeled as the 1-hop interference model. We establish 5 sessions, whose source and destination nodes are chosen at random. For each session, route is pre-calculated using the shortest-path-first or Dijkstra's algorithm and fixed during the simulation. Users arrive at each source node according to a Poisson distribution, carry a file, the size of which is exponentially distributed with mean 10, and injects data at the rate governed by (8). Each user remains in the system until it finishes the file transfer. To examine the capacity of the solution, we fix a user arrival rate vector and scale it by multiplying a scalar $\rho > 0$. Increasing ρ, we can observe a steep increase of the number of users remained in the system when the traffic load approaches to the capacity bound. In simulations, we measure the total number of users in the system after 5000 time slots.

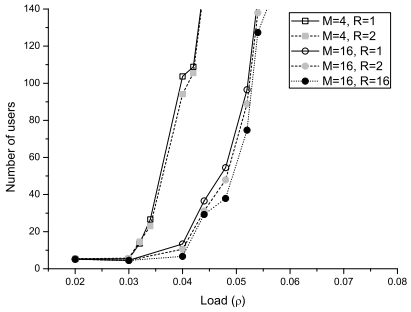
We first examine the performance of the cross-layer solution with our randomized distributed maximal scheduling (RMS). In our simulation settings, we assume that the complexity of the scheduling scheme is dominated by the iterations of MaxScheduleOneHop, and that the cost of computations and information exchanges between neighbors for UpdatePrices and UpdateAndDistNeighborhoods are negligible. In this case, RMS has $O(\delta \log^2 |V|)$ complexity. We evaluate the performance changing the total number of iterations of MaxScheduleOneHop. Implementing MaxScheduleOneHop, we divide the scheduling period into M mini-slots of unit length. During each mini-slot, nodes can finish an exchange of an RTS and a CTS. Then, we have $M = \lceil \mathbb{C}_P \log |V| \rceil \cdot \mathbb{C}_I \delta \log |V|$ from Algorithm 1, and further denote the number of phases for information updates by R, i.e., $R = \lceil \mathbb{C}_P \log |V| \rceil$. We also set $\alpha = 0.1$.

Figs. 1(a) and 1(b) show the simulation results with different M's and R's. Each result is an average of 10 simulation runs. Fig. 1(a) shows the performance of cross-

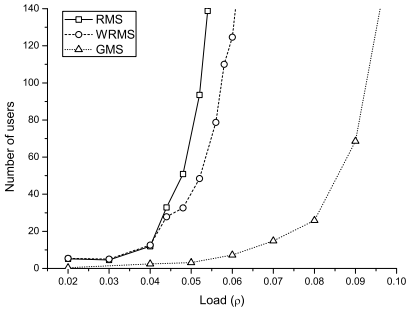
[5]We normalize the unit of transmission allowing packets to be as small as possible.



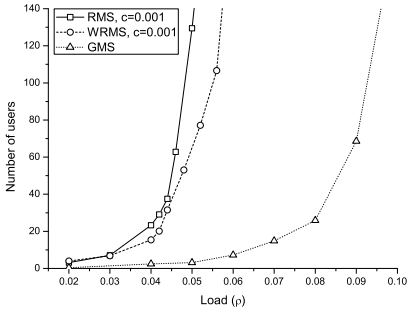
(a) RMS with different number of mini-slots; a single phase $R = 1$



(b) RMS with different number of mini-slots; multiple phases



(c) Performance comparison of different scheduling policies; $M = 32$, $R = 1$



(d) Performance considering the contention costs; $M = 32$, $R = 1$, $c = 0.001$

Fig. 1. Performance of cross-layer solutions with Randomized Maximal Scheduling (RMS) and other comparable scheduling policies.

layer solution with RMS for $R = 1$ (i.e., $\lceil \mathbb{C}_P \log |V| \rceil = 1$ and $\mathbb{C}_I \delta \log |V| = M$ in Algorithm 1). As expected, the performance improves with a larger M. However, it saturates with a small number of mini-slots. We also change R to see the effect of information updates with intermediate scheduling results. Fig. 1(b) illustrates the results with and without the updates, showing that there is no significant performance difference. The performance when $M = 16$ and $R = 1$ is almost the same as when $M = 16$ and $R = 16$.

Next, we simulate the solution with other comparable scheduling algorithms. The same rate controller (8) has been used for all scheduling schemes. We first suggest a heuristic modification of RMS, which is denoted by Weighted Random Maximal Scheduling (WRMS). WRMS operates as RMS except that each node attempts to

transmit an RTS message with a *weighted* probability. Specifically, for WRMS, we replace line 7 of Algorithm 1 with the following: "Each node u with $b(u) = 0$ chooses to transmit with probability $p(u)/\sqrt{M}$, where

$$p(u) = \frac{\sum_{v \in N_p(u)} q_{(u,v)}}{\sum_{v \in N_p(u) \cup \{u\}} \sum_{w \in N_p(v)} q_{(v,w)}}. \quad (14)$$

Upon deciding to transmit, it chooses a node v at random from $N_p(u)$ and sends an RTS message to node v." The factor $1/\sqrt{M}$ is motivated by [Joo and Shroff 2007], in which authors have shown that the probability of successful transmission can improve by decreasing the attempt probability at each mini-slot in the order of $1/\sqrt{M}$. We also compare Greedy Maximal Scheduling (GMS), under which a maximal schedule is chosen in decreasing order of queue length. Note that unlike RMS and WRMS, GMS is a centralized algorithm and known to empirically achieve the optimal performance in many network settings.

Fig. 1(c) illustrate the performance differences among RMS, WRMS, and GMS. We set the number of mini-slots M to 32 for RMS and WRMS, while the centralized GMS algorithm has $M = 0$. The results show that the scheduling schemes that further utilize queue length information achieve better performance. RMS requires the least information among three schemes, i.e., under RMS, each node has to know which of its neighbors are eligible (unscheduled and the link has a sufficiently high price (> 1)). On the other hand, WRMS needs more detailed information, i.e., the price values. GMS requires to know which link has the longest queue over the network. Hence, Fig. 1(c) implies that the performance of the cross-layer solution depends on accessibility to the queue (price) information in the network. So far, our results do not take into account the cost of mini-slots. In practice, however, the increase of contention mini-slots immediately implies the reduction of time for data transmission, which deteriorates the throughput performance. Let c denote the cost of a mini-slot, i.e., the time ratio of a mini-slot to a time slot. Fig. 1(d) shows that the performance of scheduling policies with $M = 32$ and $c = 0.001$[6]. We observe that the performance of RMS and WRMS retreats slightly. (The performance of GMS remains unchanged since it is a centralized algorithm with $M = 0$.) However, the differences are not significant and we can make the same conclusion as in the previous simulations with zero cost.

In the next experiment, we compare the performance of our cross-layer solution under more realistic network environments. We use NS-2 simulator, which considers transmitter power of each node and its attenuation, and a successful reception is determined by the SINR level at the receiver. Basically, it operates in an asynchronous manner. However, it is also possible to provide an imperfect synchronization using a guard time between intervals as IEEE 802.11b DCF networks. (See [Gast 2005] for the details.)

Under this asynchronous system with the SINR-based interference model, we compare our cross-layer solution with the existing TCP and IEEE 802.11b DCF networks. We use the default settings of NS-2 except the following. For IEEE

[6]Since IEEE 802.11b DCF has a mini-slot of 20 μs, we can make the ratio c less than 0.001 by transmitting a few packets during a time slot.

802.11b DCF, we enable RTS/CTS exchange for all data transmissions and set the maximum number of retransmissions[7] to 100 to remove the effect of packet drops. For the same reason, we set the queue size (IFQ) at each node to 100 packets. Physical link rate is set to 2Mbps and the packet size is fixed to 1460 bytes. We construct a 5x5 grid topology with 25 nodes. Distance between two neighboring nodes is 200m. All nodes use the same fixed transmission power achieving the transmission range of about 250m when there is no interference, which, however, does not imply that all transmissions within 250m distance are successful. A successful transmission/reception is determined by the received SINR level at the receiver. We establish four multi-hop UDP sessions with a fixed route. As before, for each session, users arrive at the source node according to a Poisson distribution, carry a file, the size of which is exponentially distributed, and injects packets at the rate governed by (8). The user arrival rate vector is fixed again and scaled by multiplying a scalar $\rho > 0$. We observe the number of users and packets remained in the system varying ρ.

Since we developed the distributed randomized scheduling algorithms for the time-slotted systems, we need the following modifications for asynchronous operations. We recompile the WRMS algorithm accordingly based on the IEEE 802.11b DCF protocol. Specifically, we have WRMS operate as 802.11b except:

(1) The maximum congestion window is fixed to 32 (i.e., $M = 32$ mini-slots) and there is no exponential backoff for WRMS. At each mini-slot, WRMS attempts to transmit an RTS in a probabilistic manner.

(2) At the beginning of a time slot, which can be defined as a series of mini-slots (≤ 32) followed by data transmission, each node is assumed to have the queue information of its neighbors.

(3) At each mini-slot, a node n attempts to transmit an RTS with probability $\bar{p}(n)/\sqrt{M}$, where

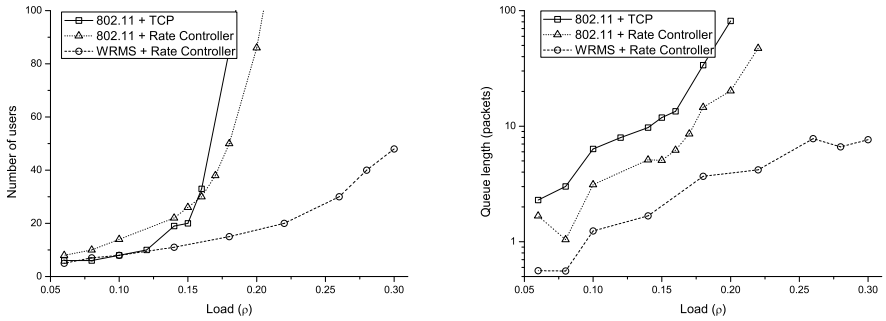
$$\bar{p}(n) = \frac{\bar{q}_n}{\sum_{m \in N(n) \cup \{u\}} \bar{q}_m},$$

where $\bar{q}_n = \sum_{k \in N(n) \cup \{n\}} \sum_{m \in N(k)} Q_{(k,m)}$, and $\sum_{m \in N(k)} Q_{(k,m)}$ corresponds the total queue length in node k. (Hence, the modified WRMS works with per-node queue instead of per-link queue.) If node n have not attempted for successive 32 mini-slots, the node updates the queue information of its neighbors and starts a new time slot, i.e., another scheduling period of 32 mini-slots. Hence, each node starts its own time slot asynchronously.

Note that in this asynchronous version of WRMS, if a node overhears neighbor's transmissions, it freezes its contention until the transmission ends.

For GMS, unfortunately, we cannot apply the same approach, and we are unable to convert it to fit in asynchronous systems. Indeed, the concept of greedy maximal schedule becomes unclear when nodes operates asynchronously. Hence, in asynchronous systems, we only use the modified WRMS and the standard IEEE 802.11b DCF as a scheduling policy.

[7]We also run simulations with the default value of 7 and obtain similar results.



(a) Number of users (b) Queue length (log scale)

Fig. 2. Performance evaluation under asynchronous system and the SINR-based interference model. The cross-layer solution of rate controller over Weighted RMS (WRMS) is compared with the standard layered solution of TCP over IEEE 802.11b DCF.

We run simulations for 2000 seconds, and measure the maximum total number of users and the total number of packets in the system during the simulation time. We compare the performance of TCP over 802.11b, rate controller (8) over 802.11b, and rate controller (8) over modified WRMS. Fig. 2(a) shows the results on the number users, in which it is clear that rate controller over WRMS significantly outperforms others. Although TCP over 802.11b works well under light loads, the number of users in the system rapidly increases as the load increases beyond a certain threshold. Under light loads, rate controller over 802.11b has a larger queue length than TCP over 802.11b, but in terms of capacity region, it performs slightly better than that of TCP over 802.11b, though the difference is marginal. In contrast, rate controller over WRMS succeeds in maintaining a small number of users in the system with heavier loads ($\rho > 2.0$), and thus achieves the largest capacity region among three.

Fig. 2(b) illustrates the total number of packets in the network when the simulations end. Roughly, a large queue length can be interpreted as a high delay. The results show that our cross-layer solution significantly outperforms the layered solutions over 802.11b, even under light loads.

6. CONCLUDING REMARKS

In this paper, we considered the problem of throughput-optimal cross-layer design of wireless networks. We propose a joint congestion control and scheduling algorithm that works under a general contention matrix based interference model and stabilizes the network for all user arrival rate vectors within a fraction $1/d_I(G)$ of the capacity region, where $d_I(G)$ is the interference degree of the underlying connectivity graph G (see Definition 2). Upper bounds on the interference degree can be obtained under a large set of graphs and interference models that are of practical interest. For example, with geometric unit-disk graphs and K-hop interference

models, it is known that $d_I(G) \leq 49$.

For practical implementation of the cross-layer solution, developing a simple distributed scheduling algorithm is essential. In this paper, we develop two randomized distributed algorithms for the maximal scheduling policy under 1-hop and 2-hop interference models. Although the performance of the maximal scheduling policy has been studied under various network settings, to the best of our knowledge, it is for the first time in the literature that the difficulties involved in its distributed implementation in the presence of interference have been discussed in detail and fully accounted for in the analysis.

In view of our results, it appears that a distributed implementation of maximal scheduling would incur significant overhead in networks where the maximum node degree is large. For example, our algorithms require $\Theta(\delta \log^2 |V|)$ rounds of computation and local message exchange under the 1-hop interference model, and $\Theta(\delta^2 \log^2 |V|)$ under the 2-hop interference model. Thus, the actual throughput obtained with the maximal scheduling policy might be considerably smaller than the theoretically guaranteed throughput. It remains to be determined if there exist other distributed scheduling policies that can provide better throughput guarantees and have lower complexity than the maximal scheduling policy under various interference models. Recently, Joo et al. [2008] have shown that under the K-hop interference models, GMS significantly outperforms the maximal scheduling policy and achieves the optimal throughput in many network settings. However, its distributed version developed by Hoepman [2004] still requires a higher complexity $O(|V|)$.

We evaluate the performance of our cross-layer solution under more realistic network settings, i.e., under the SINR-based interference model and with imperfect synchronization. The numerical results show that the cross-layer solution significantly outperforms the layered solution like TCP over IEEE 802.11 DCF network. In this direction, it also remains as an interesting open problem whether one can develop a distributed cross-layer solution with a provable performance guarantee under the SINR-based interference model and/or time-varying wireless channels.

ACKNOWLEDGEMENT

This work has been supported in part by NSF Awards CNS-0626703 and CNS-0721236, ARO MURI Award W911NF-08-1-0238, and AFOSR grant FA 9550-07-1-0456, USA.

REFERENCES

Bui, L., Eryilmaz, A., Srikant, R., and Wu, X. 2006. Joint Asynchronous Congestion Control and Distributed Scheduling for Multi-Hop Wireless Networks. In *IEEE INFOCOM*.

Chaporkar, P., Kar, K., and Sarkar, S. 2005. Throughput Guarantees Through Maximal Scheduling in Wireless Networks. 43rd Annual Allerton Conf. on Communications, Control, and Computing.

Eryilmaz, A., Ozdaglar, A., Shah, D., and Modiano, E. 2009. Distributed Cross-Layer Algorithms for the Optimal Control of Multi-hop Wireless Networks. submitted.

Gast, M. S. 2005. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc.

Georgiadis, L., Neely, M. J., and Tassiulas, L. 2006. Resource Allocation and Cross-Layer Control in Wireless Networks. *Found. Trends Netw. 1,* 1, 1–144.

H. Balakrishnan et al. 2004. The Distance-2 Matching Problem and its Relationship to the MAC-layer Capacity of Ad Hoc Wireless Networks. *IEEE JSAC 22,* 6 (Aug), 1069–1079.

Hajek, B. and Sasaki, G. 1988. Link Scheduling in Polynomial Time. *IEEE Trans. Inf. Theory 34,* 5 (Sep), 910–917.

Hoepman, J.-H. 2004. Simple Distributed Weighted Matchings. eprint.

Joo, C., Lin, X., and Shroff, N. B. 2008. Understanding the Capacity Region of the Greedy Maximal Scheduling Algorithm in Multi-hop Wireless Networks. In *IEEE INFOCOM*.

Joo, C. and Shroff, N. B. 2007. Performance of Random Access Scheduling Schemes in Multihop Wireless Networks. In *IEEE INFOCOM*.

Kelly, F., Maulloo, A., and Tan, D. 1998. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. In *Journal of the Oper. Res. Society*. Vol. 49. 237–252.

Lin, X. and Rasool, S. 2006. Constant-Time Distributed Scheduling Policies for Ad Hoc Wireless Networks. In *IEEE CDC*.

Lin, X. and Shroff, N. B. 2005. The Impact of Imperfect Scheduling on Cross-Layer Rate Control in Multihop Wireless Networks. In *IEEE INFOCOM*.

Lin, X., Shroff, N. B., and Srikant, R. 2008. On the Connection-Level Stability of Congestion-Controlled Communication Networks. *IEEE Trans. Info. Theory 54,* 5 (May), 2317–2338.

Low, S. H. and Lapsley, D. E. 1999. Optimization Flow Control — I: Basic Algorithm and Convergence. *IEEE/ACM Trans. Netw. 7,* 6, 861–874.

McKeown, N. 1995. Scheduling Algorithms for Input-Queued Cell Switches. Ph.D. thesis, University of California at Berkeley.

Modiano, E., Shah, D., and Zussman, G. 2006. Maximizing Throughput in Wireless Networks via Gossiping. *Sigmetrics Performance Evaluation Review 34,* 1, 27–38.

Neely, M. J., Modiano, E., and Li, C. 2008. Fairness and Optimal Stochastic Control for Heterogeneous Networks. *IEEE/ACM Trans. Netw. 16,* 2 (April), 396–409.

Neely, M. J., Modiano, E., and Rohrs, C. E. 2003. Power Allocation and Routing in Multibeam Satellites with Time-varying Channels. *IEEE/ACM Trans. Netw. 11,* 1, 138–152.

Peleg, D. 2000. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Sarkar, S. and Tassiulas, L. 2005. End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Ad-hoc Networks. *IEEE Trans. Autom. Control 50,* 9 (Sep), 1246–1259.

Sharma, G., Joo, C., Mazumdar, R. R., and Shroff, N. B. 2009. Supplement for Joint Congestion Control and Distributed Scheduling for Throughput Guarantees in Wireless Networks. Online supplement.

Sharma, G., Joo, C., and Shroff, N. B. 2006. Distributed Scheduling Schemes for Throughput Guarantees in Wireless Networks. In *the 44th Annual Allerton Conference on Communications, Control, and Computing*.

Sharma, G., Mazumdar, R. R., and Shroff, N. B. 2006a. Maximum Weighted Matching with Interference Constraints. In *IEEE FAWN*.

Sharma, G., Mazumdar, R. R., and Shroff, N. B. 2006b. On the Complexity of Scheduling in Wireless Networks. In *ACM MOBICOM*.

Stolyar, A. L. 2005. Maximizing Queueing Network Utility subject to Stability: Greedy Primal-Dual Algorithm. *Queueing Systems 50,* 4, 401–457.

Tassiulas, L. 1998. Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches. In *IEEE INFOCOM*.

Tassiulas, L. and Ephremides, A. 1992a. Jointly Optimal Routing and Scheduling in Packet Radio Networks. *IEEE Trans. Info. Theory 38,* 1 (Jan), 165–168.

Tassiulas, L. and Ephremides, A. 1992b. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Trans. Autom. Control 37,* 12 (December), 1936–1948.

Wu, X., Srikant, R., and Perkins, J. R. 2007. Scheduling Efficiency of Distributed Greedy Scheduling Algorithms in Wireless Networks. *IEEE Trans. Mobile Computing 6,* 6, 595–605.

Xiao, L., Johansson, M., and Boyd, S. 2004. Simultaneous routing and resource allocation via dual decomposition. *IEEE Trans. Comm. 52,* 7 (July), 1136–1144.

Yaiche, H., Mazumdar, R., and Rosenberg, C. 2000. A Game-theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks. *IEEE/ACM Trans. Netw. 8,* 5 (Oct), 667–678.

Received June 2009; September 2009; accepted October 2009