This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Polynomial Complexity Algorithms for Full Utilization of Multi-hop Wireless Networks

Atilla Eryilmaz
Lab. for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA, 02139
Emails: eryilmaz@mit.edu

Asuman Ozdaglar
EECS Department
Massachusetts Institute of Technology
Cambridge, MA, 02139
Emails: asuman@mit.edu

Eytan Modiano
Aeronautics and Astronautics Department
Massachusetts Institute of Technology
Cambridge, MA, 02139
Email: modiano@mit.edu

***Abstract*— In this paper, we provide and study a general framework that allows the development of distributed mechanisms to achieve full utilization of multi-hop wireless networks. In particular, we describe a generic randomized routing, scheduling and flow control scheme that is applicable to a large class of interference models, and that allows for the development of distributed algorithms which maximize network throughput and utilization.**

In particular, we focus on a specific interference model, namely the secondary interference model, and develop distributed algorithms with polynomial communication and computation complexity in the network size. This is an important result given that earlier throughput-optimal algorithms developed for such a model relies on the solution to an NP-hard problem. This results in a polynomial complexity cross-layer algorithm that achieves throughput optimality and fair allocation of network resources amongst the users. We further show that our algorithmic approach enables us to efficiently approximate the capacity region of a multi-hop wireless network.

I. Introduction

There has been considerable recent interest in developing network protocols to achieve the multiple objectives of throughput maximization and fair allocation of resources among competing users. Much of the work in wireless communication networks has focused on centralized control and has developed *throughput-optimal* policies (e.g. [27], [19], [10]). However, these policies do not directly lend themselves to distributed implementation, which is essential in practice. In this paper, we provide a class of randomized routing, scheduling and flow control algorithms that achieve throughput-optimal and fair resource allocations that is amenable to distributed implementation with polynomial communication and computation complexity.

In their seminal work, Tassiulas and Ephremides developed a joint routing-scheduling algorithm that stabilizes the network whenever the arrival rates are within the stability (capacity) region. In [26], Tassiulas showed that randomized algorithms can be used to achieve maximum throughput in input queued switches with linear computational complexity. Other research, for example, [1], [23], [19], [10], have contributed to the analysis of *centralized* throughput optimal policies in wireless networks.

This paper contributes to the study of resource allocation in multi-hop wireless networks in several fundamental ways.

First, we propose a generic cross-layer mechanism with two components: a randomized scheduling-routing component (implemented by the network nodes) aimed at allocating resources to the flows efficiently; and a dual congestion control component (implemented at the sources) aimed at regulating the flow rates to achieve fairness. The scheduling-routing component, which is an extension of the idea introduced in [25] to the multi-hop setting, suggests a framework for distributed implementation of centralized throughput optimal policies. The congestion control component, which works in parallel with the scheduler-router, adapts the flow rates dynamically in a completely decentralized manner. With the inclusion of the congestion controller the overall cross-layer mechanism not only maximizes throughput, but also achieves fair division of the resources among flows, where *fairness* is defined using the utility-maximization framework of Kelly et al. [12], [13] and further improved in subsequent works [16], [30], [24].

Second, for the secondary interference model[1], we show that our cross-layer mechanism can be implemented via a distributed algorithmic approach. This approach involves the operation of two sequential algorithms. A novel feature of these algorithms is their operation on an appropriately constructed conflict graph. The use of the conflict graph leads to a partitioning of the network, whereby the decisions can be made independently in different partitions. Moreover, the operations on the conflict graph can be mapped into network level operations using the special structure of the problem. These distributed algorithms not only achieve throughput-optimal and fair allocations, but also have polynomial communication and computation complexity.

Finally, we demonstrate that our policy enables an algorithmic method of estimating the stability region of multi-hop wireless networks. This suggests a novel approach to

The first two authors were supported by the Control-Based Mobile Ad-Hoc Networking (CBMANET) Program under DARPA subcontract no. 060786, and the third author was supported by ONR grant N000140610064 and by NSF ITR grant CCR-0325401.

[1] In the secondary interference model, two links interfere if they share a node or if there is a link that connects any of the end nodes of the two links. This interference model prevents real world issues such as the hidden terminal problem (see [20]).

0743-166X/07/$25.00 ©2007 IEEE

stability region characterization, which is a very difficult task in general.

Our paper is related to recent work combining flow control with routing and scheduling, including [14], [25], [9], [18], [10]. While these papers also propose algorithms achieving fair and throughput-optimal allocations, the routing-scheduling component of these algorithms are based on the centralized control approach of [27], and cannot be implemented in a distributed manner in wireless networks. Moreover, for the secondary interference model, the centralized optimization involved in the operation of these algorithms is NP-hard, which further limits their practical implementation.

Other related works include [15], [28], which develop distributed algorithms that guarantee 50% utilization of the stability region for a primary interference model[2]. While distributed implementation of these algorithms is possible, this comes at the cost of sacrificing a significant portion of the capacity of the network (see, for example, [4], [3]). As more general interference models are considered, even more of the capacity of the network needs to be sacrificed for distributed implementation (e.g., [29], [5]). For example, in the case of a secondary interference model with the grid topology, distributed implementation can only guarantee 12.5% of the capacity of the network. In recent work [17], a similar approach to ours, also based on [26], has been used to develop distributed schedulers for the primary interference model by utilizing Gossip mechanisms. In this work, we provide a different scheduling-routing algorithm combined with a congestion controller in a slightly extended system model whereby multi-hop flows are considered. We develop a class of randomized cross-layer mechanisms that can be implemented in a distributed manner for the secondary interference model while guaranteeing throughput-optimality and fairness.

The paper is organized as follows. In Section II, we describe the system model and our goal. In Section III, we describe a generic randomized scheme for scheduling-routing-congestion control, and prove its throughput-optimality and fairness properties. In Section IV, we use the randomized scheme to design and analyze distributed algorithms for the secondary interference model. Finally, in Section V we provide simulation results.

II. System model and Goal

Consider a wireless network that is represented by an undirected graph, $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, which has a node set $\mathcal{N}$ (with cardinality N), a link set $\mathcal{L}$ (with cardinality L), and $|\mathcal{F}|$ source-destination node pairs $\{s_1, d_1\}, \ldots, \{s_{|\mathcal{F}|}, d_{|\mathcal{F}|}\}$. We refer to a source-destination pair $\{s_f, d_f\}$ as a flow, denoted by f, and denote the set of flows by $\mathcal{F}$. We let $\{x_f[t]\}$ denote the arrival process for flow f, i.e., $x_f[t]$ is the number of packets that arrive at node s_f in slot t. We use the notation $\lambda_f[t]$ to denote the *mean arrival rate of flow f in slot t*, i.e., $\lambda_f[t] = E[x_f[t]]$. Then, the *mean arrival rate of flow f* is defined as $\lambda_f = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \lambda_f[t]$ whenever it exists. We assume a time slotted system with synchronized nodes, where each slot is just long enough to accommodate a single packet transmission.

At each node, a buffer (queue) is maintained for each destination. We let $q_{n,d}[t]$ denote the length of the queue at node n destined for node d at the beginning of slot t.

*Definition 1 (***Stability):** A given queue is called *stable* if $\mathbb{E}[q_{n,d}[\infty]] < \infty$, where $q_{n,d}[\infty]$ denotes the random variable with distribution given by the steady-state distribution of $\{q_{n,d}[t]\}$. The network is *stable* if all queues are stable; and *unstable* otherwise.

We consider a general interference model formulation specified by a set of pairs of links that interfere with each other, i.e., we say that two links *interfere* if their concurrent transmissions collide. We assume that if two interfering links are activated in a slot, both transmissions fail. Note that this includes a large class of graph-theoretic interference models considered in the scheduling literature (e.g. the primary interference model [22], [15], [28], [4], or the secondary interference model [2], [29], [5]).

We use $\pi = \left\{\pi_{(n,m)}\right\}_{(n,m)\in\mathcal{L}}$ to denote a link *allocation vector* (or *schedule*), and Π to denote the *set of feasible allocations* where a feasible allocation is a set of links in which no two links interfere with each other.

*Definition 2 (***Capacity [Stability] Region Λ):** Let $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ be a given network and Π be the set of feasible allocations. The *capacity (or stability) region* Λ of the network is given by the set of vectors $\mathbf{r} = (r_f)_{f\in\mathcal{F}}$ for which there exists $z^{(f)}_{(n,m)} \geq 0$ for all $(n,m) \in \mathcal{L}$ and $f \in \mathcal{F}$, representing the rate of flow f traffic over link (n,m), such that both the flow conservation constraints at the nodes and the feasibility constraints are satisfied, i.e.,

(C1) For all $n \in \mathcal{N}$ and $f \in \mathcal{F}$, we have[3]

$$r_f \mathbf{1}_{s_f=n} + \sum_{k:(k,n)\in\mathcal{L}} z^{(f)}_{(k,n)} = \sum_{m:(n,m)\in\mathcal{L}} z^{(f)}_{(n,m)},$$

(C2) $$\left[\sum_{f\in\mathcal{F}} z^{(f)}_{(n,m)}\right]_{(n,m)\in\mathcal{L}} \in Conv(\Pi).$$ [4]

It is shown in [27], [19] that Λ is the set of mean arrival rates for which there exists a policy that stabilizes the network.

Given the general model described above, our goal is to design distributed algorithms that achieve *throughput-optimality* and *fair allocation* of the network resources amongst the flows. Following the extensive literature on the topic (e.g. [27], [23], [19], [9]) we call a policy throughput-optimal if it can support any mean arrival rate in the capacity region without violating the network stability.

To define fairness we use the "utility maximization" framework of economics: we associate a utility function for each

[2]In the primary interference model, each feasible allocation consists of links that do not share a node, i.e. each feasible allocation is a *matching*.

[3]We use $\mathbf{1}_A$ as the indicator function of event A.

[4]$Conv(A)$ denotes the convex hull of set A, which is the smallest convex set that includes A. The convex hull is included due to the possibility of timesharing between feasible allocations.

flow f, $U_f(\cdot)$, over the mean arrival rates whereby $U_f(\lambda_f)$ is a measure of the utility gained by flow f for the mean arrival rate λ_f. We assume, based on the law of diminishing returns, that the function U_f is concave and non-decreasing for all f. Then, a mean arrival rate vector λ^* is referred to as a *fair allocation* if it is an optimal solution of the convex optimization problem:

$$\lambda^* \in \arg\max_{\lambda \in \Lambda} \sum_{f \in \mathcal{F}} U_f(\lambda_f). \quad (1)$$

Hence, a fair allocation is a mean arrival rate vector that maximizes the aggregate utility over all flows in the network. It is known that by defining $U_f(\cdot)$ appropriately, different fairness criteria of interest, such as proportional or max-min fairness, can be achieved ([12], [13], [16], [24], [9], [18], [14]).

III. Generic Cross-layer Scheme

In this section, we provide the description of a generic congestion control-routing-scheduling scheme that achieves the throughput-optimality and fairness goals of Section II. The scheme combines ideas from recently studied congestion controllers designed for wireless networks (e.g. [9], [18], [15], [25], [6]), and the randomized scheduling strategy introduced by Tassiulas in his seminal work [26]. Our algorithm not only extends the use of randomized scheme of [26] to multi-hop networks with general interference models, but also utilizes the parallel use of a dual congestion controller to achieve fairness.

Here, we provide a concise description of our generic cross-layer algorithm, and refer the interested reader to an earlier paper [8] which includes discussions and detailed analysis of the scheme. The generic scheme is composed of two components: the scheduling-routing component that is implemented by the network, and the congestion control component that is implemented by the users (or the sources of the flows). The scheduling-routing component builds on two algorithms: one, called PICK, which randomly picks a feasible allocation satisfying a specific condition [cf. (4)]; and the other, called COMPARE, which contains a network-wide comparison operation [cf. (5)] that appears to necessitate centralized control. In Section IV we tackle these issues by developing specific distributed algorithms that implement these PICK and COMPARE operations for the secondary interference model.

The scheme operates in *stages*, each stage containing a finite number of time slots with the number of slots is a design choice. The scheduling-routing and congestion control decision is updated at the beginning of each stage, and is kept unmodified throughout the stage.

*Definition 3 (***Generic Cross-layer Scheme):**
SCHEDULING-ROUTING COMPONENT:
• At stage t, for each $(n,m) \in \mathcal{L}$, let its *weight* be

$$w_{(n,m)}[t] = w_{(m,n)}[t] \quad \triangleq \quad \max_d |q_{n,d}[t] - q_{m,d}[t]|, \quad (2)$$

which is also referred to as the *maximum differential backlog* of link (n,m)[5].

[5] $w_{(n,m)}[t]$ can be interpreted as a measure of the importance of the link.

• Also, define the *optimum feasible allocation* $\overset{\star}{\pi}_{\mathbf{w}}[t]$ for $\mathbf{w}$ as

$$\overset{\star}{\pi}_{\mathbf{w}}[t] \in \arg\max_{\pi \in \Pi} \sum_{l \in \mathcal{L}} w_l[t]\pi_l \quad \equiv \quad \arg\max_{\pi \in \Pi} (\mathbf{w}[t] \cdot \pi). \quad (3)$$

• PICK: Scheduler randomly picks *any* feasible allocation $\tilde{\pi}[t] \in \Pi$, that satisfies

$$P(\tilde{\pi}[t] = \overset{\star}{\pi}_{\mathbf{w}}[t]) \geq \delta, \qquad \text{for all } \mathbf{w}[t] \text{ and } t, \quad (4)$$

for some $\delta > 0$.
• COMPARE: The actual allocation for the next stage is updated as dictated by the following expression.

$$\pi[t+1] = \begin{cases} \pi[t] & \text{if } \mathbf{w}[t] \cdot \pi[t] \geq \mathbf{w}[t] \cdot \tilde{\pi}[t] \\ \tilde{\pi}[t] & \text{otherwise} \end{cases}. \quad (5)$$

DUAL CONGESTION CONTROL COMPONENT:
• Let $\mathbf{q}$ denote the queue-length vector at the beginning of stage t. Then, flow f generates $x_f[t]$ packets per slot throughout the stage according to

$$x_f[t] \quad = \quad \min\left\{ U_f'^{-1}\left(\frac{q_{s(f),d(f)}}{K}\right), M \right\}, \quad (6)$$

where M and K are positive scalars. ◇

The scheduling-routing component of the above algorithm is based on [26] whereby the existing feasible allocation is compared to a randomly picked feasible allocation, and the one with the larger total weight is implemented in the next stage. The same strategy is also used in a recent work [17] which develops another deterministic distributed algorithm for the primary interference model, and randomized algorithms based on gossiping techniques. Here, we make a few natural extensions to their model by considering multi-hop traffic and including congestion control to achieve fairness. In a more recent work [21], constant complexity implementations have been proposed at the expense of negligible efficiency loss for the primary interference model. But, the algorithm is not directly extendible to the secondary interference model scenario.

The dual congestion control component of our generic algorithm is easy to implement at each source because it only requires the queue-length of the buffer at the source. This is in contrast to several earlier mechanisms that require the price information of *all* the links on the route of that flow [13], [16], [24]. Also, since each source only needs to know its own utility function, the flow control mechanism can operate in a completely decentralized fashion.

The next theorem, proved in [8], establishes that the proposed cross-layer mechanism guarantees stability and achieves fair allocation with arbitrary degree of accuracy. It has been shown in earlier works [18], [9], [15], [25] that a similar result holds in the case of a centralized scheduler. Below, we state that it holds despite the imperfect nature of the randomized algorithm.

Theorem 1: For any generic cross-layer scheme, finite constants, C_1, C_2, can be found so that

$$\sum_{n\in\mathcal{N}}\sum_{d\in\mathcal{N}} \overline{q_{n,d}} \quad \leq \quad C_1 K, \tag{7}$$

$$\sum_{f\in\mathcal{F}} U_f(\bar{x}_f) \quad \geq \quad \sum_{f\in\mathcal{F}} U_f(x_f^\star) - \frac{C_2}{K}, \tag{8}$$

where $\bar{x}_f \triangleq \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[x_f[t]]$, and similarly for $\overline{q_{n,d}}$. Note that by choosing K sufficiently large, fair allocation can be achieved due to (8), while stability of the queues are guaranteed due to (7). Also note that using the above joint scheduling-flow control mechanism, it is possible to approximate the stability region of complicated traffic and network scenarios. In particular, by modifying the K parameter, one can find a highly accurate approximation to the capacity region, which is otherwise very difficult to characterize or compute. We will illustrate this approach through an example in Section V.

IV. Algorithm Design

In Section III, we established the throughput-optimality and fairness properties of a cross-layer mechanism that can be applied to a large class of interference models. In this section, we focus on the secondary interference model and outline a distributed low-complexity algorithmic approach for performing the tasks outlined in the generic randomized scheme as described in Definition 3.

This approach involves the sequential operation of two algorithms, which we refer to as PICK and COMPARE: The PICK algorithm is a randomized, distributed algorithm that yields a feasible schedule $\tilde{\pi}[t]$ satisfying (4) in finite time. The COMPARE algorithm compares the total weights of the old schedule $\pi[t]$ with the new schedule $\tilde{\pi}[t]$ according to (5) in a distributed manner. An important feature of the COMPARE algorithm is the use of the conflict graph of the two schedules. On the conflict graph, a spanning tree can be constructed in a distributed manner and used for comparison of the weights of the two schedules in polynomial time. The conflict graph enables a natural partitioning of the network, whereby decisions can be made independently in different partitions in a distributed manner. As we will show, the operations on the conflict graph can be mapped to the actual network operations owing to the special structure of the problem.

The schedule used for packet transmissions is updated at the beginning of each stage. Throughout a stage, packet transmissions are performed according to the schedule updated at the beginning of that stage. In parallel with the packet transmissions, PICK and COMPARE algorithms are implemented. Since the same medium is shared, the data packet transmissions can collide with the control messages generated by these algorithms. To prevent such collisions, time is divided into two intervals, namely the *control signalling interval* (CSI) during which control messages are locally communicated, and the *data transmission interval* (DTI) during which data packets are transferred (see Figure 1). Notice that both PICK and COMPARE algorithms operate during CSI, while queue-lengths are updated during DTI. It is assumed that all the nodes are synchronized to the same CSI/DTI division of time. This assumption can be relaxed by adding a buffer interval between CSI and DTI to accommodate propagation delays. Alternatively, the control signalling can be performed over an orthogonal channel through frequency division. Finally, we assume that each transceiver can perform carrier sensing during transmission without the need to decode its reception.

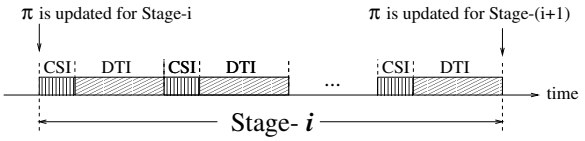

Fig. 1. Division of time into data transmission and control signalling intervals.

It is important to note that in our algorithm the overhead introduced by the control signalling can be made arbitrarily small by increasing the length of a stage to a high enough value. This fact follows from the *fixed* amount of control messages required by our algorithm *per stage*. Thus, the number of control messages versus the data messages in a stage can be made negligible by increasing the stage duration. This will naturally result in slower convergence, but the stability and fairness results of Theorem 1 will continue to hold.

We assume that each node has a unique ID number picked from a totally ordered set. Let $ID(n)$ denote the ID number of node n. Then, unique ID numbers can be assigned to links, denoted by $ID(n,m) = ID(m,n)$ for link (n,m). This assumption is essential for each node (and link) to identify its neighboring nodes (and links), and will be used in the distributed implementation of our algorithms.

A. PICK Algorithm

In this section, we present a distributed algorithm that randomly picks a feasible allocation $\tilde{\pi}$ with the property that any feasible allocation has a positive probability of being chosen as required by (4). In the description of the algorithm, when we say a node *withdraws*, we mean that the node stops its search for a feasible link during the current stage, but continues to listen to other transmissions. The algorithm makes sure that each node has a positive probability of attempting transmission at the beginning of the algorithm. The idea is to send Ready-to-Send (RTS) and Clear-to-Send (CTS) packets including the ID numbers of the nodes in order to create a feasible allocation. By appending ID numbers to the RTS/CTS packets, the algorithm enables each node to have a list of those links in its local neighborhood that are picked by the algorithm.

*Definition 4 (*PICK *Algorithm):* At every node $n \in \mathcal{N}$ perform the following steps:

(A1) In step 1, with probability $p_n \in [\alpha, 1)$ for some $\alpha \in (0,1)$, n transmits a (RTS) message.

(A1a) If n senses another transmission during its (RTS) transmission, it withdraws.

(A2a) If n does not sense another transmission, in step 2, it chooses one of its neighbors, say m, randomly with equal probabilities, and transmits (RTS, $ID(m)$).

(A2b) If m observes a collision, it withdraws.

(A3a) If m gets n's message, in step 3, it sends back a (CTS) message.

(A3b) If m senses another transmission during its (CTS) transmission, it withdraws.

(A3c) If n observes an idle, it withdraws.

(A4a) If m does not sense another transmission during its (CTS) transmission, in step 4, it transmits (CTS, $ID(n,m)$).

(A4b) If n does not receive m's response, it withdraws.

(A5) In step 5, n transmits (CTS, $ID(n,m)$), and the link between n and m is activated; link (n,m) is added to $\tilde{\pi}$. $\diamond$

The algorithm assures between steps (A1) and (A2a), that no two transmitters are neighboring each other; at (A2b), that no transmitter is a neighbor to a receiver; between (A3a) and (A4a), that no two receivers are neighbors. Finally, during (A4a) and (A5), the picked link is announced to the neighbors of the receiver and the transmitter, respectively.

Notice that the algorithm need not result in a *maximal* feasible allocation[6] at its termination. This does not influence the results of Theorem 1, but will have an effect on the rate of convergence of the algorithm. With a simple modification, the above algorithm can be extended to obtain a maximal feasible allocation and hence better convergence properties.

Proposition 1: The above PICK algorithm satisfies

(i) The resulting $\tilde{\pi}$ is a feasible allocation.

(ii) It takes at most 5 transmissions per node to terminate.

(iii) The probability of picking any feasible allocation is at least $(\min(\alpha, 1-\alpha)/D)^N > 0$, where D is the maximum degree[7] of $\mathcal{G}$. In particular, since $\overset{\star}{\pi}_{\mathbf{w}}[t]$ is a feasible schedule, we have $\mathbb{P}(\tilde{\pi}[t] = \overset{\star}{\pi}_{\mathbf{w}}[t]) \geq (\min(\alpha, 1-\alpha)/D)^N > 0$.

(iv) At the termination, for any link $(n,m) \in \mathcal{L}$, all the neighbors of n and m are aware of (n,m)'s state, i.e., know whether (n,m) is in $\tilde{\pi}$ or not.

Proof: Step (A1a) assures that if two neighboring nodes attempt to transmit, they sense each other and withdraw. In Step (A2b), the event that more than one neighbors of a node are attempting to transmit is detected, and in that event all of the transmitters withdraw from transmission in Step (A3c). Finally, step (A3b) guarantees that two neighbors do not become receiving ends of two different links. Thus, all the events that leads to interfering links are eliminated in these steps, and the resulting allocation must be feasible, which proves (i). Claim(ii) follows immediately from the construction of the algorithm.

To prove Claim(iii), note that if the initially picked set of links in steps (A1) and (A2a) happen to be feasible, they are not eliminated throughout the algorithm, because the algorithm is designed to eliminate only those links that interfere with each other. Thus, we are interested in finding a lower bound on the probability of picking a given feasible schedule, say $W \in \Pi$, at the start. Thus, we need to have exactly $|W|$ nodes, one from each link in W choose to transmit in step (A1) (which happens with probability $\alpha^{|W|}$), and all the remaining nodes must be silent (which happens with probability $\geq (1-\alpha)^{N-|W|}$). If each of those nodes which chose to transmit, picks its outgoing link that lies in W for transmission in step (A2a) (which happens with probability $\geq (\alpha/D)^{|W|}$), then the resulting schedule will be exactly W. Hence, the probability that PICK yields a given feasible schedule W is $\geq (\alpha/D)^{|W|}(1-\alpha)^{N-|W|} \geq (\min(\alpha, 1-\alpha)/D)^N$, where the last step follows from $D \geq 1$.

Claim(iv) follows from the fact that the links that are activated are announced to neighboring nodes via the message (CTS, $ID(n,m)$), and therefore all the neighbors know the IDs of the activated links in their two hop neighborhood. ■

We note that this algorithm does not depend on the queue-lengths, which greatly simplifies its implementation, because no queue-length information exchange is necessary between neighboring nodes. Further, due to part (iii), the best allocation must also have a positive probability. This fact together with parts (i)-(iii) prove that the algorithm is actually sufficient for Theorem 1 to hold. At the end of PICK, $\tilde{\pi}$ gives a feasible allocation, that is known only locally. In particular, due to part (iv) of Proposition 1, every node knows those links of its neighbors that are in $\tilde{\pi}$.

B. COMPARE Algorithm

In this section, we propose and analyze a distributed algorithm that compares the total weight associated with two feasible schedules, $\pi[t]$ and $\tilde{\pi}[t]$, with local control signal transmissions, and choose the one with the larger weight as the schedule to be used during the next stage. We note that this algorithm applies to interference models other than the secondary interference model. In the following, we will omit the time index for ease of presentation.

The algorithm relies on constructing the *conflict graph* associated with π and $\tilde{\pi}$ which contains information about interfering links in the two schedules. The conflict graph, $\mathcal{G}'(\pi, \tilde{\pi}) = (\mathcal{N}', \mathcal{L}')$, of π and $\tilde{\pi}$ can be generated as follows[8]: Each link l in $\pi \cup \tilde{\pi}$ corresponds to a node in the conflict graph, and if links $l_1 \in \pi$ and $l_2 \in \tilde{\pi}$ interfere with each other, an edge is drawn between the nodes corresponding to l_1 and l_2 in the conflict graph. Note that, since both π and $\tilde{\pi}$ are feasible schedules, no two links in the same schedule (π or $\tilde{\pi}$) can interfere with each other, i.e., there is no edge between two nodes of the same schedule in $\mathcal{G}'$. Every node in $\mathcal{G}'$ can compute its own weight [as defined in (2)], and has a list of its neighbors in $\mathcal{G}'$ by part (iv) of Proposition 1. We will develop the algorithms using the conflict graph $\mathcal{G}'$ and show at the end of this section that the special structure enables us to map the operations to the graph $\mathcal{G}$.

Our COMPARE Algorithm is composed of two procedures that are implemented consecutively: FIND SPANNING TREE

[6] A *maximal* feasible allocation is a set of links to which no new link that does not interfere with any of the existing links can be added.

[7] $deg(n) \triangleq |\{m \in \mathcal{N} : (n,m) \in \mathcal{L}, \text{ or } (m,n) \in \mathcal{L}\}|$.

[8] We use N', L' to denote the cardinalities of $\mathcal{N}', \mathcal{L}'$. Also, we will refer to $\mathcal{G}'(\pi, \tilde{\pi})$ simply as $\mathcal{G}'$ for convenience.

and COMMUNICATE & DECIDE. The FIND SPANNING TREE procedure finds a spanning tree for each connected component of $\mathcal{G}'$ in a distributed fashion. Then, the COMMUNICATE & DECIDE procedure exploits the constructed tree structure to communicate and compare the weights of the two schedules in a distributed manner.

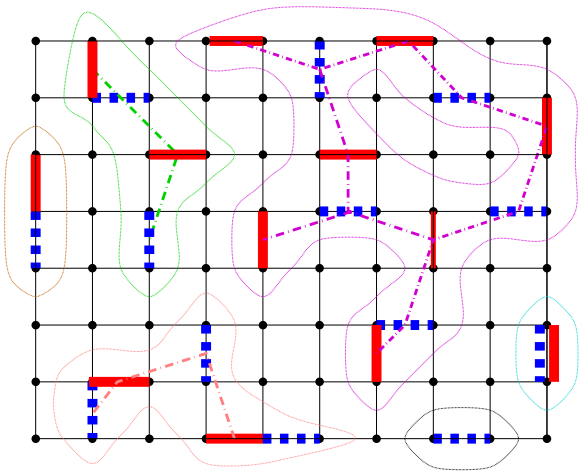

Fig. 2. 8x10 grid network example with two feasible schedules indicated by solid and dashed bold links. The conflict graph decomposes into 6 disconnected components.

To illustrate the definitions and operation of the algorithms we consider the grid network depicted in Figure 2. In this network, nodes are located on the corner points of a grid, and each interior node has four links incident to it. To demonstrate the construction of the conflict graph, suppose we are given two feasible schedules, π and $\tilde{\pi}$. In the figure, solid bold links belong to schedule π, while dashed bold links are in $\tilde{\pi}$. We use dash-dotted thin lines to connect the links of the two schedules that interfere with each other. In general, it is not necessary that the conflict graph be connected. For example, in Figure 2, we observe six disconnected components. The conflict graph corresponding to the largest connected component is given in Figure 3, where links in π are drawn as circular dots, while links in $\tilde{\pi}$ are drawn as square dots.

Remark 1: Disconnected components of the conflict graph can decide on which schedule to use, independent of each other. This is possible because by construction of the conflict graph the resulting schedule is guaranteed to be feasible even if the choices of two disconnected components are different. This decomposition contributes to the distributed nature of the algorithm. Namely, the size of the graph within which the comparison is to be performed is likely to be reduced. Notice that with this approach, the chosen schedule may be a combination of the two candidate schedules, π and $\tilde{\pi}$, because different connected components may prefer different schedules. This merging operation will result in a schedule that is better than both π and $\tilde{\pi}$.

Based on this remark, henceforth our algorithm will focus on the decision of a single connected component.

1) FIND SPANNING TREE *Procedure:* The object of the FIND SPANNING TREE procedure is to find, in a distributed fashion, a spanning tree for each of the connected components in the conflict graph. In our model, every node in the conflict graph $\mathcal{G}'$ corresponds to an undirected link in the original graph $\mathcal{G}$, and has a unique ID[9]. In order to compare two link IDs, we use lexicographical ordering[10].

Our distributed FIND SPANNING TREE procedure is based on token generation and forwarding operations. For the construction of a spanning tree, at least one token needs to be generated within each connected component. This can be guaranteed by requiring every node in the conflict graph that has the lowest ID number among its neighbors to generate a token. Each token, carrying the ID of its generator, performs a depth-first traversal (cf. [7]) within the connected component to construct a spanning tree. This token progressively adds nodes into its spanning tree while avoiding the construction of cycles. An example is depicted in Figure 3 for the largest connected component of Figure 2.

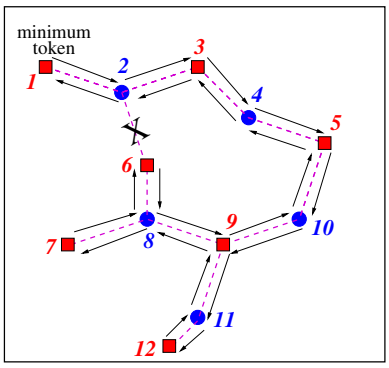

Fig. 3. A connected component of the conflict graph from which the link crossed is eliminated to obtain a spanning tree. The path of the minimum token is indicated with arrows. The nodes are labeled with numbers for future reference.

The above procedure focuses on the operation of a single token generated at one of the nodes within the connected component. In general, there may be multiple tokens generated within the same connected component. Each token attempts to form its spanning tree labeled with its ID number (i.e. the ID number of the token's generator). Since only one spanning tree is required at the end of the procedure, our algorithm is designed to keep the spanning tree with the smallest ID number, while eliminating the others. This elimination is performed when the token of a spanning tree enters a node that has been traversed by another token. If the incoming token has smaller ID, then the token ignores the previous token and continues the construction of its tree, and if its ID is larger, then it is immediately deleted. We have the following proposition for this algorithm.

[9]In [11], it was shown that unique IDs are required to be able to find a spanning tree in a distributed fashion.

[10]Without loss of generality, assume $ID(n) < ID(m)$ and $ID(i) < ID(j)$: If $ID(n) < ID(i)$, then $ID(n,m) < ID(i,j)$ for all m, j; and if $ID(n) = ID(i)$ and $ID(m) < ID(j)$, then $ID(n,m) < ID(i,j)$.

Proposition 2: Consider the conflict graph $\mathcal{G}' = (\mathcal{N}', \mathcal{L}')$, and let D' denote the maximum degree of $\mathcal{G}'$. The FIND SPANNING TREE Procedure finds a spanning tree of all components of the conflict graph in $O(D'L')$ time[11], and with $O(D'N')$ message exchanges for each $n' \in \mathcal{N}'$. In particular, at the termination of the procedure, every node $n' \in \mathcal{N}'$ has a list of its neighbors in the constructed spanning tree.

Proof: To prove that the constructed subgraph by the smallest token is in fact a spanning tree, we need to show that every node is in the formed subgraph, and that the subgraph contains no cycles. To argue that every node must be in the subgraph, let us assume, to arrive at a contradiction, there is a node, say $z' \in \mathcal{N}'$, that is not in the subgraph but is within the connected component. If any of z''s neighbors had held the token at any time, then it must have attempted to forward the token to z' before it sends the token back to its parent. But, if a token attempt is made to z', it will ACCEPT it because it is the first time it encounters such an attempt. This argument implies that none of the neighbors of z' can be in the subgraph. If the same arguments are made repeatedly, this implies that the whole subgraph must be empty. But, we know that the node that generates the token is in the subgraph by default. Hence, we get a contradiction, and the subgraph must contain every node within the connected component. The argument that the subgraph contains no cycles follows from the fact that every node ACCEPTs only those token transmissions that do not form a cycle. Thus, the resulting subgraph must be acyclic.

The procedure is constructed so that whenever a token with a larger ID crosses any node of the spanning tree being constructed by a token with a smaller ID, the token with the larger ID along with its spanning tree is eliminated. By definition, a spanning tree has to contain every node and thus all the tokens must meet with the spanning tree of the smallest token sometime. Therefore, by the end of the procedure, only the spanning tree of the smallest token survives.

To compute the complexity, note we take into account the complexity of resolving potential collisions of tokens. It is not difficult to see that such each collision can be resolved in $O(D')$ message exchanges. Since, an operation of $O(D')$ operations must be performed for $2L'$ times, we need $O(D'L')$ time for the operation to complete. However, each node will only transmit $O(D')$ messages in the process only when it is receiving and transmitting a token. Since, there are at most $O(N')$ tokens in the system, the number of messages transmitted by each node is $O(D'N')$. ∎

2) COMMUNICATE & DECIDE *Procedure:* We use the spanning tree formed on the conflict graph to compare weights. The idea is to convey the necessary information from the leaves up to the root of the tree (i.e. COMMUNICATE Procedure) so that the schedule with the higher weight is chosen (cf. (5)), and then send back the decision to the leaves (i.e. DECIDE Procedure). The COMMUNICATE & DECIDE procedure can be explained in two parts as follows:

[11] $f(n) = O(g(n))$ means that there exists a constant $c < \infty$ such that $f(n) \leq cg(n)$ for n large enough.

COMMUNICATE: The leaves communicate their weights to their parents. If the parent is in π it adds its weight to the sum of the weights announced by its children. If, on the other hand, it is in $\tilde{\pi}$ it subtracts its weight from the sum of its children's weights. The resulting value becomes the new weight of the parent. Then, the parent acts as a leaf with the updated weight in the next iteration. This recursive update is repeated until the root is reached.

DECIDE: At the end of COMMUNICATE, the weight of the root of the spanning tree will be $\sum_{l\in\pi} w_l - \sum_{l\in\tilde{\pi}} w_l$. Depending on whether the root's weight is positive or negative, the root decides π or $\tilde{\pi}$, respectively, as the better schedule, and broadcasts its decision down the tree.

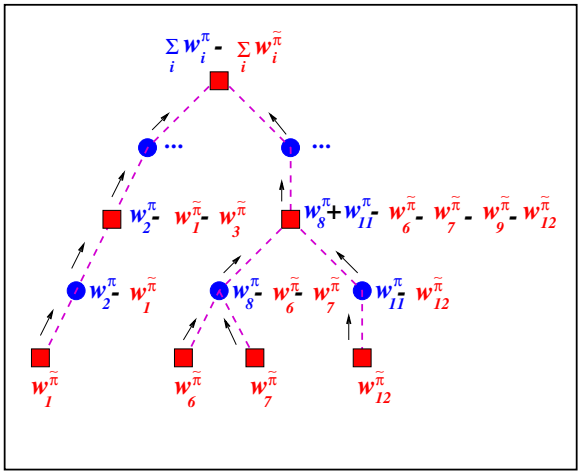

Fig. 4. The iterative communication of the weights of the two schedules from the leaves to the root for the spanning tree of Figure 3.

An example of this procedure is provided in Figure 4 for the spanning tree given in Figure 3. We have the following complexity result for this procedure.

Proposition 3: Consider the conflict graph $\mathcal{G}' = (\mathcal{N}', \mathcal{L}')$, and let D' denote the maximum degree of $\mathcal{G}'$. The COMMUNICATE & DECIDE procedure correctly finds the schedule with the larger weight in $O(D'L')$ time.

Proof: The algorithm is designed so that when node n' transmits its current sum to its parent, where the value of the sum is the difference of the weights of schedule π and $\tilde{\pi}$ only for the subtree rooted at n'. Thus, sum at the root of the spanning tree is the difference of two weights of schedule π and $\tilde{\pi}$. Decision is a simple comparison of the sign of this sum. This decision is broadcast to the children of the root, and hence all the nodes in the connected component knows about the better schedule and can switch to it by the end of the procedure. The depth of the spanning tree can be at most $O(L')$ and each collision resolution operation can take at most $O(D')$ time. Thus, the whole algorithm terminates in $O(L'D')$ time. ∎

Notice that the complexity results in the propositions are given in terms of $\mathcal{G}'$. We can translate them into bounds on $\mathcal{G}$ through the following inequalities: $L' < N^2$, $D' < N$. Propositions 1, 2 and 3, together with Theorem 1 yields the

following result.

Theorem 2: The distributed implementations of PICK and COMPARE Algorithms designed for the secondary interference model asymptotically achieve throughput-optimality and fairness with $O(N^3)$ time and $O(N^2)$ message exchanges per node, per stage. □

Before we complete the section, we make a few important remarks on the operation and extension of the algorithms.

Remark 2: The algorithms we develop in this section operate over the conflict graph $\mathcal{G}'$. These operations can be transformed into operations in the actual graph $\mathcal{G}$. Such a transformation would be difficult for a general conflict graph. However, in our scenario the graph has a special structure that enables the mapping. The critical observation is that transmissions within a feasible schedule has no interference. Thus, links that form π and $\tilde{\pi}$ can perform operations in $\mathcal{G}'$ by partitioning CSI (cf. Figure 1) into two disjoint time intervals. During the first interval, only links that make up π communicate, while in the second interval only nodes that make up $\tilde{\pi}$ communicate. The operation of each link can easily be mapped into operations at its two end nodes by assigning one node to each operation, who will then coordinate the operation. With such a separation of time, the operations described for the conflict graph can be translated into operations in the actual network.

Remark 3: Recall from Remark 1 that the conflict graph is likely to be composed of multiple disconnected components, which increases the distributed nature of the algorithms. Even though we did not pursue this direction here, this likelihood can be increased by dynamically modifying the activation probabilities, $\{p_n\}_n$, in the PICK Algorithm so that the picked schedule has more disconnected components. This way, the localized nature of the algorithm can be improved.

V. SIMULATIONS

In this section, we provide simulation results for the distributed algorithms developed in Section IV for the grid topology (see Figure 2). We use the notation $[i, j]$ to refer to the node at the i^{th} row and j^{th} column of the grid. Throughout, we simulate utility functions of the form $U_i(x) = \gamma_i \log(x)$, which corresponds to weighted proportionally fair allocation (see [13], [24]).

We first consider a network of size 6x6, with four flows: Flow-1 from $[1, 1]$ to $[6, 6]$, Flow-2 from $[5, 2]$ to $[6, 3]$, Flow-3 from $[5, 5]$ to $[5, 1]$, and Flow-4 from $[4, 1]$ to $[1, 4]$. Here, we are interested in the evolution of the throughputs of each flow for $K = 100$ and $\gamma_i = 0.5$ for each $i \in \{1, 2, 3, 4\}$. The simulation results are depicted in Figure 5. We observe that the throughputs of the flows converge to different values depending on their source-destination separation. For example, Flow-2 achieves the highest throughput since its source is only two hops from its destination. The fluctuations in the evolutions are due to the random nature of the algorithm, which tracks the queue-length evolutions.

Next, we simulate a 10x10 network with two flows: Flow-1 from $[1, 1]$ to $[8, 9]$, and Flow-2 from $[9, 2]$ to $[2, 10]$. Here, we

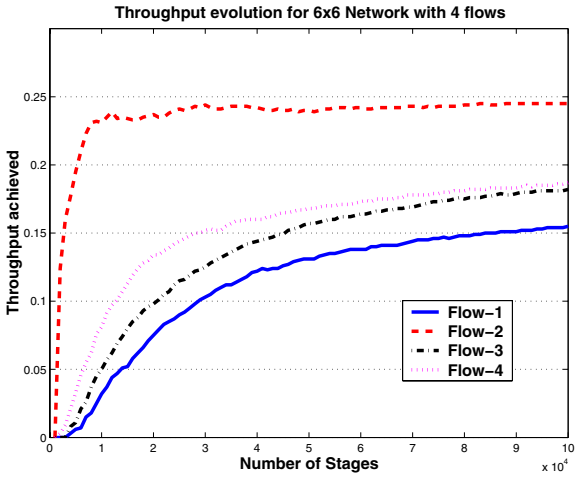

Fig. 5. The throughput evolution of the 6x6 network for $K = 100$, $\gamma_i = 0.5$.

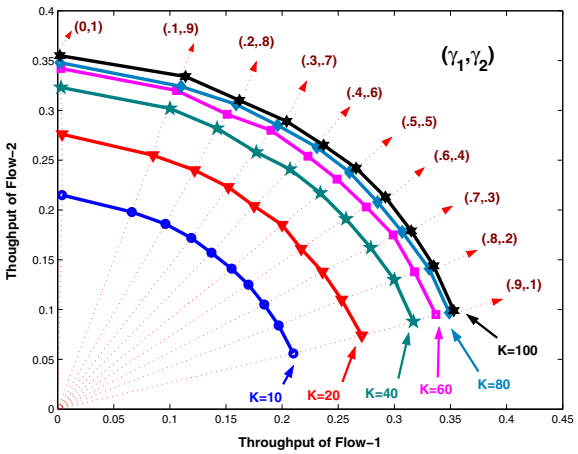

Fig. 6. Throughputs of flows with varying K and (γ_1, γ_2).

focus on the throughputs achieved for the flows as a function of K with varying γ_i for each flow. We aim to observe the average flow rates as functions of K and (γ_1, γ_2). Notice that each (γ_1, γ_2) combination corresponds to a different weighting for the weighted-proportionally fair allocation. Thus, for a fixed K, the throughputs corresponding to different (γ_1, γ_2) combinations actually outline the *rate region* that the algorithm achieves for that K. Then, as K grows Theorem 1 implies that this region grows at a decreasing rate, until it converges to the stability region Λ.

We performed simulations for K varying from 10 to 100, and (γ_1, γ_2) ranging from $(0, 1)$ to $(1, 0)$ with $\gamma_1 + \gamma_2 = 1$ at each intermediate point. The simulation results are provided in Figure 6. We observe that for a given K, the rate region is a convex region. Also, as K grows, the region expands at a decreasing rate agreeing with our expectations. We further note that with this algorithmic method, the stability region of a wireless network, that is otherwise difficult to find, can be determined with high accuracy.

Finally, for the previous simulation setting, we fix K to 100 and vary (γ_1, γ_2) as before, and compare the rate region

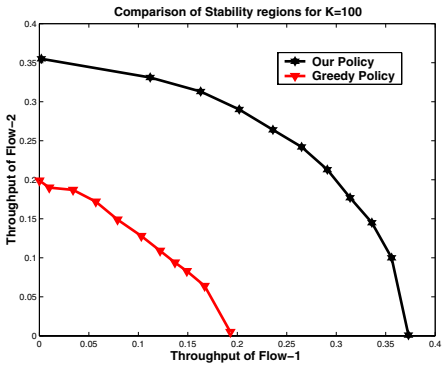

Fig. 7. Throughputs of flows with $K = 100$ and varying (γ_1, γ_2).

achieved with our policy to the greedy distributed maximal scheduling policy proposed in earlier works [15], [4]. The greedy policy always picks a random maximal schedule, but does not perform any comparison with an earlier schedule. The two regions are plotted in Figure 7. We observe that there is a considerable gain in throughput when our algorithm is implemented compared to the greedy algorithm.

VI. Conclusions

In this work, we provided a framework for the design of distributed cross-layer algorithms for full utilization of multi-hop wireless networks. To that end, we first described a generic scheduling-routing-congestion control mechanism that generalizes a randomized scheduling policy introduced in [26] to multi-hop networks, and incorporates a decentralized congestion control algorithm that works in parallel with the randomized algorithm. We showed that the resulting cross-layer algorithm achieves throughput-optimality and fair allocation of the resources despite the randomized and imperfect nature of the scheduling-routing algorithm. Then, we developed specific distributed algorithms for the secondary interference model. For this model, existing throughput-optimal strategies require that an NP-hard problem be solved by a centralized controller at every time instant. In this work, we showed that this is not necessary, and full utilization of the network can be achieved with distributed algorithms having only polynomial communication and computational complexity.

An important byproduct of our approach is the use of the developed cross-layer algorithms to find (with high accuracy) the stability region of ad-hoc wireless networks, that are otherwise difficult to characterize.

References

[1] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting. Scheduling in a queueing system with asynchronously varying service rates, 2000. Bell Laboratories Technical Report.

[2] E. Arikan. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory*, 30:681–685, 1984.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[4] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu. Joint asynchronous congestion control and distributed scheduling for wireless networks. Proceedings of IEEE Infocom 2006.

[5] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In *Proceedings of the Allerton Conference on Control, Communications and Computing*, 2005.

[6] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle. Jointly optimal congestion control, routing, and scheduling for wireless ad hoc networks. In *Proceedings of IEEE Infocom*, Barcelona, Spain, April 2006.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, McGraw-Hill Book Company, 2001.

[8] A. Eryilmaz, E. Modiano, and A. Ozdaglar. Randomized algorithms for throughput-optimality and fairness in wireless networks. In *Proc. of IEEE Conf. on Decision and Control*, San Diego, CA, December 2006.

[9] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proc. of IEEE Infocom*, pages 1794–1803, Miami, FL, March 2005.

[10] A. Eryilmaz and R. Srikant. Joint congestion control, routing and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications, special issue on Nonlinear Optimization of Communication Systems*, 14:1514–1524, August 2006.

[11] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5:66–77, 1983.

[12] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[13] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[14] X. Lin and N. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proceedings of IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

[15] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks. In *Proceedings of IEEE Infocom*, Miami, FL, March 2005.

[16] S. H. Low and D. E. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7:861–875, December 1999.

[17] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *ACM SIGMETRICS/IFIP Performance*, 2006.

[18] M.J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proceedings of IEEE Infocom*, pages 1723–1734, Miami, FL, March 2005.

[19] M.J. Neely, E. Modiano, and C.E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. In *Proceedings of IEEE Infocom*, pages 745–755, April 2003.

[20] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, Second edition, 2000.

[21] S. Sanghavi, L. Bui, and R. Srikant. Distributed link scheduling with constant overhead, 2006. Technical Report.

[22] G. Sasaki and B. Hajek. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 32:910–917, 1988.

[23] S. Shakkottai and A. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the AMS, Series 2,* A volume in memory of F. Karpelevich, 207:185–202, 2002.

[24] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhäuser, Boston, MA, 2004.

[25] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[26] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proceedings of IEEE Infocom*, pages 533–539, 1998.

[27] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 36:1936–1948, December 1992.

[28] X. Wu and R. Srikant. Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing. In *Proceedings of* IEEE Conference on Decision and Control., 2005.

[29] X. Wu and R. Srikant. Bounds on the capacity region of multi-hop wireless networks under distributedgreedy scheduling. In *Proceedings of IEEE Infocom*, 2006.

[30] H. Yaiche, R. R. Mazumdar, and C. Rosenberg. A game-theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking*, 8(5):667–678, October 2000.