Distributed Cross-Layer Algorithms for the Optimal Control of Multi-hop Wireless Networks

Atilla Eryilmaz, Asuman Ozdaglar, Devavrat Shah, and Eytan Modiano

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 allows for a set of imperfections in the operation of the randomized scheduler to account for potential errors in its operation. These imperfections enable the design of a large class of low-complexity and distributed implementations for different interference models. We study the effect of such imperfections on the stability and fairness characteristics of the system, and explicitly characterize the degree of fairness achieved as a function of the level of imperfections. Our results also reveal the relative importance of different types of errors on the performance of the system, and provide valuable insight to the design of distributed controllers with favorable fairness characteristics.**

In the second part of the paper, 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. [41], [30], [14]). 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 rate of the exogenous flows are within the stability (capacity) region. In [40], Tassiulas showed that randomized algorithms can be used to achieve maximum throughput in input queued switches with linear computational complexity. To improve the exponentially high delay performance of [40], [17] introduced randomized algorithms for switches. Other research, for example, [1], [22], [34], [37], [30], [2], [14], [35], [36], 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 three components: a randomized scheduling component and a 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 component extends the idea introduced in [40], which is proposed as a low-complexity implementation of the centralized scheduler of [41]. However, in its original form the algorithm is not suitable for distributed implementation since it requires global operations. To facilitate distributed implementation, several types of imperfections are added to the scheduler in [27]. In this work, we further add a routing component to the framework to optimally steer multi-hop traffic, and a congestion control component that regulates the flow rates to achieve fair division of the resources among flows, where *fairness* is defined using the utility-maximization framework of Kelly et al. [20], [21] and further improved in subsequent works [25], [44], [38].

Second, under our proposed generic cross-layer scheme, we study the proximity of the achieved rate allocation to the fair allocation, and explicitly characterize the degree of performance loss as a function of the imperfections of the underlying scheduler. Moreover, by revealing the relative importance of different types of errors on the performance, our analysis also yields principles for efficient design of distributed network controllers.

Third, 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

Atilla Eryilmaz is with the Ohio State University {eryilmaz@ece.osu.edu}. Asuman Ozdaglar, Devavrat Shah, and Eytan Modiano are with the Massachusetts Institute of Technology {{asuman, devavrat, modiano}@mit.edu}.

[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 [31]).

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, our policy suggests an algorithmic method of estimating the stability region of multi-hop wireless networks, which is otherwise a very difficult task to characterize.

Our paper is related to recent work combining flow control with routing and scheduling, including [23], [39], [13], [29], [14]. 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 [41], 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 [11], [26], [24], [42], 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, [7], [6]). As more general interference models are considered, even more of the capacity of the network needs to be sacrificed for distributed implementation (e.g., [43], [8]). 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. [27] also used [40] to develop distributed schedulers by utilizing Gossip mechanisms.

More recently, the throughput performance of greedy maximal matching schedulers are investigated for general interference models and geometric graphs ([18]), which proves that $1/6$ of the stability region is guaranteed to be achievable by such schedulers. In other recent works ([19], [5]), distributed schedulers are proposed with attractive low delay characteristics.

In this work, we provide a scheduling-routing algorithm combined with a congestion controller for a general system model whereby multi-hop flows are considered. Following the approach of [27], we allow various types of errors to occur during the scheduling operation, which facilitates the design of distributed implementations. We analyze the impact of such imperfections on the fairness characteristics of the overall joint mechanism, and explicitly characterize effect of different types of errors on the performance.

The rest of 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 V, we use the randomized scheme to design and analyze distributed algorithms for the secondary interference model. Finally, in Section VI we provide simulation results.

Throughout the paper, we denote the dot product of two vectors, say $\mathbf{X}$ and $\mathbf{Y}$, as $\langle \mathbf{X}, \mathbf{Y} \rangle$.

[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*.

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). We assume a time slotted system with synchronized nodes, where each slot is long enough to accommodate a single packet transmission over each link in $\mathcal{L}$ unless there is interference. We refer to the flow that enters the network at node n and leaves it at node d as Flow-(n,d). We let $\mathbf{X}[t] = \left(X_n^{(d)}[t]\right)_{n\in\mathcal{N}}^{d\in\mathcal{N}}$ denote the vector of arrivals to the network in slot t with $X_n^{(d)}[t]$ corresponding to the arrivals for Flow-(n,d). We use the notation $x_n^{(d)}[t]$ to denote the *mean flow rate of Flow-(n,d) in slot t*, i.e., $x_n^{(d)}[t] = E[X_n^{(d)}[t]]$. Then, the *mean flow rate of Flow-(n,d)* is defined as $\bar{x}_n^{(d)} = \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} x_n^{(d)}[t]$ whenever it exists.

We consider a general interference model 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 [33], [24], [42], [7], or the secondary interference model [3], [43], [8]).

We use $\mathbf{S}[t] = \left(S_{(n,m)}[t]\right)_{(n,m)\in\mathcal{L}}$ to denote a link *allocation vector* (or *schedule*) at time t, and $\mathcal{S}$ 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. We introduce the notation $S_{(n,m)}^{(d)}$ to distinguish packets destined for different nodes: at any given slot t, $S_{(n,m)}^{(d)}[t] \in \{0,1\}$ is 1 if link (n,m) serves a packet destined for node d in that slot, and 0 otherwise. This implies that $\sum_{d\in\mathcal{N}} S_{(n,m)}^{(d)}[t] = S_{(n,m)}[t]$, for all $(n,m) \in \mathcal{N}$.

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. Evolution of $Q_n^{(d)}[t]$ when $n \neq d$ satisfies

$$Q_n^{(d)}[t+1] \leq \left(Q_n^{(d)}[t] - S_{out(n)}^{(d)}[t]\right)^+ + X_n^{(d)}[t] + S_{into(n)}^{(d)}[t], \tag{1}$$

where $(y)^+ = \max(0, y)$. Also,

$$S_{into(n)}^{(d)}[t] \triangleq \sum_{\{k:(k,n)\in\mathcal{L}\}} S_{(k,n)}^{(d)}[t]$$

is a shorthand for the maximum number of packets that can be internally routed to node n that are destined for node d. Similarly, $S_{out(n)}^{(d)}[t] \triangleq \sum_{\{m:(n,m)\in\mathcal{L}\}} S_{(n,m)}^{(d)}[t]$ are the maximum number of packets that can leave node n and are destined for node d. When, $n = d$, we set $Q_d^{(d)}[t] = 0$, for all t, because in that case the packets have already reached their destination.

Next, we introduce the concepts of network stability and capacity region.

*Definition 1 (***Stability):** A given queue, say $Q_n^{(d)}$, is *stable* if $\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[Q_n^{(d)}[t]] < \infty$. The network is *stable* if all queues are stable; and *unstable* otherwise.

*Definition 2 (***Capacity [Stability] Region $\mathcal{C}$):** The capacity (stability) region $\mathcal{C}$ is the set of $(\bar{x}_n^{(d)})_{n,d\in\mathcal{N}} \geq \mathbf{0}$ for which there exists an algorithm that can stabilize the network[3].

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. [41], [37], [30], [13]) we call a policy *throughput-optimal* if it can support any mean flow rate in the capacity region without violating the network stability.

To define fairness we use the "utility maximization" framework of economics: with each flow, say Flow-(n,d), we associate a utility function $\mathcal{U}_{n,d}(\cdot)$, of the mean flow rates whereby $\mathcal{U}_{n,d}(\bar{x}_n^{(d)})$ is a measure of the utility gained by Flow-(n,d) for the mean flow rate $\bar{x}_n^{(d)}$. We assume, based on the law of diminishing returns, that the function $\mathcal{U}_{n,d}(\cdot)$ is concave and non-decreasing for all flows. Then, a mean flow rate vector $\overset{\star}{\mathbf{x}}$ is referred to as a *fair allocation* if it is an optimal solution of the convex optimization problem:

$$\overset{\star}{\mathbf{x}} \in \arg\max_{\overline{\mathbf{x}}\in\mathcal{C}} \sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\bar{x}_n^{(d)}). \tag{2}$$

Hence, a fair allocation is a mean flow rate vector that maximizes the aggregate utility over all flows in the network. It is known that by defining $\mathcal{U}_{n,d}(\cdot)$ appropriately, different types od fairness, such as proportional or max-min fairness, can be achieved ([20], [21], [25], [38], [13], [29], [23]).

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. [13], [29], [24], [39], [9]), and the randomized scheduling strategy introduced by Tassiulas in his seminal work [40]. Our algorithm not only extends the use of randomized scheme of [40] to multi-hop networks with general interference models, but also utilizes the parallel use of a dual congestion controller to achieve fairness.

The generic scheme is composed of three components: the scheduling and routing components that are implemented by the network, and the congestion control component that is implemented by the users (or the sources of the flows). The scheduling component builds on two algorithms: one, called PICK, which randomly picks a feasible allocation satisfying a specific condition [see Eq. (6) below]; and the other, called UPDATE, which contains a network-wide comparison operation [see Eq. (7) below]. In the operation of PICK and UPDATE algorithms, we allow for various types of imperfections and relaxations to accommodate errors and to facilitate distributed implementations. We will comment on the nature of these imperfections and relaxations after the description of the cross-layer scheme. The routing component determines which packets to be served over which links so as to optimize their routes. Finally, the congestion controller component adjusts the rate of injected traffic into the network to full utilize the resources. In particular, the goal of the congestion controller is to solve (2).

The scheme operates in *stages*, each stage containing a finite number of time slots where 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):** The cross-layer algorithm is composed of three components: a randomized scheduler with imperfections characterized by the parameters (δ,γ,ψ); a routing component that steers packets towards optimal paths; and a congestion controller component that regulates the amount of injected traffic into the network to maximize the network utilization. Next, we describe each of these components.

SCHEDULING(δ,γ,ψ) COMPONENT: The scheduling component determines the service rates of all the links in the network, namely $\mathbf{S}[t]$, by performing the following operations

• At stage t, for each link $(n,m)\in\mathcal{L}$, we define its *weight* as

$$W_{(n,m)}[t] = W_{(m,n)}[t] \quad \triangleq \quad \max_d \left| Q_n^{(d)}[t] - Q_m^{(d)}[t] \right|, \tag{3}$$

which is also referred to as the *maximum differential backlog* ([30], [14]) of link (n,m).

• We define the *optimum feasible allocation* $\overset{\star}{\mathbf{S}}_{\mathbf{W}}[t]$ for $\mathbf{W}$ as

$$\begin{aligned} \overset{\star}{\mathbf{S}}_{\mathbf{W}}[t] \quad &\in \quad \arg\max_{\mathbf{S}\in\mathcal{S}} \sum_{l\in\mathcal{L}} w_l[t]S_l \;\equiv\; \arg\max_{\mathbf{S}\in\mathcal{S}} \langle\, \mathbf{W}[t], \mathbf{S}\rangle \qquad (4)\\ &= \quad \arg\max_{\mathbf{S}\in\mathcal{S}} \langle\, \mathbf{Q}[t], \mathbf{S}_{out} - \mathbf{S}_{in}\rangle. \qquad (5)\end{aligned}$$

• PICK: Scheduler randomly picks *any* feasible allocation $\mathbf{R}[t]\in\mathcal{S}$, that satisfies

$$\mathbb{P}(\mathbf{R}[t] = \overset{\star}{\mathbf{S}}_{\mathbf{W}}[t]) \geq \delta, \qquad \text{for all } \mathbf{W}[t] \text{ and } t, \tag{6}$$

for some $\delta > 0$.

• UPDATE: The schedule for time t, $\mathbf{S}[t]$ is updated such that it satisfies

$$\begin{aligned} &\mathbb{P}\left(\langle \mathbf{Q}[t], \mathbf{S}_{out}[t] - \mathbf{S}_{in}[t]\rangle \geq \right.\\ &\quad \max\{\langle \mathbf{Q}[t], \mathbf{S}_{out}[t-1] - \mathbf{S}_{in}[t]\rangle,\\ &\qquad (1-\gamma)\langle \mathbf{Q}[t], \mathbf{R}_{out}[t] - \mathbf{R}_{in}[t]\rangle\}) \geq 1-\psi, \end{aligned} \tag{7}$$

for some $\gamma,\psi\in[0,1)$.

The parameters (δ,γ,ψ) in the above scheduler capture different type of imperfections and relaxations: δ relaxes the constraint of picking the optimum feasible allocation in each iteration, hence significantly reduces the complexity of this operation; γ captures the potential errors in the computation of the total weight of the randomly selected schedule; and ψ captures the potential errors in the comparison of the weights of the previous and the random scheduler.

[3]Note that, under this definition, the capacity region is monotone, i.e., if $\overline{\mathbf{x}}\in\mathcal{C}$, then $\overline{\mathbf{y}}\leq\overline{\mathbf{x}}$ (component-wise) must also be in $\mathcal{C}$.

ROUTING COMPONENT: Once the link rate vector $\mathbf{S}[t]$ is determined by the scheduler, the router determines which packets to transmit over them.

• Let $d^\star_{nm} = d^\star_{mn} \triangleq \arg\max_d \left|Q_n^{(d)}[t] - Q_m^{(d)}[t]\right|$. Then,

– if $Q_n^{(d^\star_{nm})}[t] \geq Q_m^{(d^\star_{nm})}[t]$: Serve $S_{(n,m)}[t]$ packets from $Q_n^{(d^\star_{nm})}$ to $Q_m^{(d^\star_{nm})}$, and

– if $Q_n^{(d^\star_{nm})}[t] < Q_m^{(d^\star_{nm})}[t]$: Serve $S_{(n,m)}[t]$ packets from $Q_m^{(d^\star_{nm})}$ to $Q_n^{(d^\star_{nm})}$.

DUAL CONGESTION CONTROL COMPONENT:

• Let $\mathbf{Q}$ denote the queue-length vector at the beginning of stage t. Then, each node, say n, generates $X_n^{(d)}[t]$ packets to be transmitted to node d, for each d, such that

$$X_n^{(d)}[t] \quad = \quad \left\{ \mathcal{U}'^{-1}_{n,d}\left(\frac{Q_n^{(d)}[t]}{K}\right)\right\}_0^M, \tag{8}$$

where M and K are positive scalars, and $\{z\}_a^b := \min(\max(z,a),b)$.

Notice that (8) is equivalent to solving

$$X_n^{(d)}[t] = \arg\max_{y\in[0,M]} \left(K\mathcal{U}_{n,d}(y) - y\, Q_n^{(d)}[t]\right), \tag{9}$$

and also, note that if there exists no flow from n to d, we can define $\mathcal{U}_{n,d}(\cdot) \equiv 0$ to get $X_n^{(d)} \equiv 0$.

The imperfections included in the scheduling component are likely to occur when randomized or distributed methods are employed to perform these operations. Our goal in this work is to understand the effect of these parameters on the optimality (or fairness) characteristics of the cross-layer mechanism. Our framework covers various schedulers that are introduced in the literature (e.g. [40], [27], [32]). In particular, [32] yields a scheduler for the first-order interference model with $\gamma = 1/m$, and $\psi = 0$, where m is a design parameter. Also, [27] contain algorithms with $\gamma = \psi = 0$, as well as gossip-based algorithms with arbitrarily small γ, ψ parameters. None of these works, however, contain a study of the cross-layer scheme with congestion control. Hence, our analysis of the generic cross-layer scheme will be directly applicable to all these cases. In Section V, we will introduce a new algorithm that is applicable to higher order interference models, and use our results to show that it achieves optimality using operations that grow polynomially with the number of nodes.

IV. ANALYSIS

In this section, we study the throughput-optimality and fairness properties of the proposed generic cross-layer scheme. Our analysis relies on the notions of ε-relaxed stability region and ε-fair allocation, which we define next.

Definition 4 (ε-relaxed stability region, $\mathcal{C}(\varepsilon)$):

$$\mathcal{C}(\varepsilon) \quad \triangleq \quad \{\overline{\mathbf{x}} \geq \mathbf{0} : (\bar{x}_n^{(d)} + \varepsilon)_{n,d\in\mathcal{N}} \in \mathcal{C}\}.$$

Definition 5 (ε-fair allocation, $\overset{\star}{\mathbf{x}}(\varepsilon)$):

$$\overset{\star}{\mathbf{x}}(\varepsilon) = \arg\max_{\overline{\mathbf{x}}\in\mathcal{C}(\varepsilon)} \sum_{n,d} \mathcal{U}_{n,d}(\bar{x}_n^{(d)})$$

Note that as $\varepsilon \downarrow 0$, we have $\overset{\star}{\mathbf{x}}(\varepsilon) \rightarrow \overset{\star}{\mathbf{x}}$. Let us define a new state, $\mathbf{Y} := (\mathbf{Q}, \mathbf{S})$, which forms a Markov Chain under our generic cross-layer scheme.

In the following theorem, we focus on the scheduling-routing component of the algorithm by assuming that the mean flow rates lie inside the ε-relaxed stability region. Here, we assume that the arrivals are inelastic, i.e., their statistics are not modified throughout the operation of the scheduling-routing component. Later, we will add the congestion control component into the framework.

Theorem 1: Assume that $X_n^{(d)}[t]$ is independent and identically distributed (i.i.d.) [4] for all t and flows with $\mathbb{E}\left[\left(X_n^{(d)}\right)^2[1]\right] \leq A < \infty$. Assume that $\varepsilon > 0$ is chosen such that $\frac{\varepsilon}{d_{max}} > \gamma + \sqrt{\frac{\psi}{\delta}}$, where d_{max} is the maximum degree of the network, and δ, ψ, and γ are the parameters of the generic cross-layer scheme.

Let the Lyapunov function $V(\mathbf{Y})$ be defined as $V(\mathbf{Y}) = \sum_{n,d}\left(Q_n^{(d)}\right)^2 = \|\mathbf{Q}\|_2^2$. Then, for any mean arrival vector $\lambda := \left(\mathbb{E}[X_n^{(d)}[1]]\right)_{n,d\in\mathcal{N}} \in \mathcal{C}(\varepsilon)$, the scheduling-routing (δ,γ,ψ) policy guarantees, for some finite T,

$$\begin{aligned}\Delta V_t^{(T)}(\mathbf{Y}) \quad &\triangleq \quad \mathbb{E}\left[V(\mathbf{Y}[t+T]) - V(\mathbf{Y}[t]) \mid \mathbf{Y}[t] = \mathbf{Y}\right] \\ &\leq \quad -cT\left(\frac{\varepsilon}{d_{max}} - \gamma - \sqrt{\frac{\psi}{\delta}}\right)\sum_{n,d} Q_n^{(d)} + B_1,\end{aligned}$$

where B_1 is a bounded positive number, and c is a positive constant.

Proof: The proof is moved to the Appendix. ■

An immediate consequence of Theorem 1 is provided next.

Corollary 1: The generic cross-layer algorithm stabilizes any traffic with a mean flow rate vector lying inside $\mathcal{C}(d_{max}(\gamma + \sqrt{\psi/\delta}))$.

Proof: Pick ε to be larger but arbitrarily close to $d_{max}(\gamma + \sqrt{\psi/\delta})$. Then, from Theorem 1, we have

$$\Delta V_t^{(T)}(\mathbf{Y}) \quad \leq \quad -\varepsilon' \sum_{n,d} Q_n^{(d)} + B_1,$$

for some $\varepsilon' > 0$. This shows that the Foster-Lyapunov criterion is satisfied (see e.g. [4]), and therefore we must have $\limsup_{T\rightarrow\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n,d}\mathbb{E}[Q_n^{(d)}[t]] < \infty$. ■

The scheduling-routing component of the above algorithm is based on [40] 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 [27] which develops another deterministic distributed algorithm for the primary interference model, and randomized algorithms based on gossiping techniques that are applicable to more general interference models. In other works (e.g. [24], [8], [7], [43], [32], [18]), low complexity implementations have been proposed at the expense of different levels of efficiency loss for the primary interference model. But, many of these results are not applicable in the higher interference models scenarios.

[4] The assumption of i.i.d. arrivals is not critical to the analysis. The same results continue to hold for processes with mild ergodicity properties ([15]).

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 [21], [25], [38]. 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 studies the impact of the parameters (δ, γ, ψ) in the proposed cross-layer mechanism on its stability and fairness characteristics. Earlier works in this context (e.g. [29], [13], [24], [39]) are applicable only to the case of a centralized scheduler. Below, we extend these results in the presence of a randomized scheduling-routing component with imperfections.

Theorem 2: For any generic cross-layer (δ, γ, ψ) scheme, there exists finite constants, C_1, C_2, such that: for any $\varepsilon > 0$ for which $\frac{\varepsilon}{d_{max}} > \gamma + \sqrt{\frac{\psi}{\delta}}$, we have

$$\sum_{(n,d)\in\mathcal{N}^2} \bar{q}_n^{(d)} \leq C_1 K, \tag{10}$$

$$\sum_{(n,d)\in\mathcal{N}^2} \mathcal{U}_{n,d}(\bar{x}_n^{(d)}) \geq \sum_{(n,d)\in\mathcal{N}^2} \mathcal{U}_{n,d}\left(\overset{\star}{x}{}_n^{(d)}(\varepsilon)\right) - \frac{C_2}{K}, \tag{11}$$

where $\bar{q}_n^{(d)} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[Q_n^{(d)}[t]]$,
and $\bar{x}_n^{(d)} \triangleq \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[X_n^{(d)}[t]]$.

Proof: Recall that $\mathbf{Y} = (\mathbf{Q}, \mathbf{S})$, and that the definition of the Lyapunov function introduced in Theorem 1 is: $V(\mathbf{Y}) = \sum_{n,d}\left(Q_n^{(d)}\right)^2$. Then, by using the same arguments as in the derivation of (17) and (18), it can be shown that
$\Delta V_t^{(T)}(\mathbf{Y})$

$$\begin{aligned}
&\triangleq \mathbb{E}\left[V(\mathbf{Y}[t+T]) - V(\mathbf{Y}[t]) \mid \mathbf{Y}[t] = \mathbf{Y}\right] \\
&\leq \sum_{\tau=0}^{T-1} \mathbb{E}\left[V(\mathbf{Y}[t+\tau+1]) - V(\mathbf{Y}[t+\tau] \mid \mathbf{Y}[t] = \mathbf{Y}\right] \\
&\leq 2\sum_{\tau=0}^{T-1} \mathbb{E}\left[\langle\, \mathbf{Q}[t+\tau], \mathbf{S}_{in}[t+\tau] + \mathbf{X}[t+\tau]\right. \\
&\qquad \left. -\mathbf{S}_{out}[t+\tau]\rangle \mid \mathbf{Y}[t] = \mathbf{Y}\right] + T(b_1+b_2),
\end{aligned}$$

for some finite constants b_1, b_2 that were introduced in the proof of Theorem 1. Next, we add and subtract $2K\sum_{\tau=0}^{T-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) \mid \mathbf{Y}[t] = \mathbf{Y}\right]$, and re-arrange the terms to get

$\Delta V_t^{(T)}(\mathbf{Y})$

$$\begin{aligned}
&\leq 2K\sum_{\tau=0}^{T-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) \mid \mathbf{Y}[t]\right] \\
&\quad +2\sum_{\tau=0}^{T-1}\mathbb{E}[\mathbb{E}[K\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) - \\
&\qquad\qquad \langle\, \mathbf{Q}[t+\tau], \mathbf{X}[t+\tau]\rangle \mid \mathbf{Y}[t+\tau]] \mid \mathbf{Y}[t]] \\
&\quad -2\sum_{\tau=0}^{T-1}\mathbb{E}\left[\langle\, \mathbf{Q}[t+\tau], \mathbf{S}_{out}[t+\tau] - \mathbf{S}_{in}[t+\tau] \mid \mathbf{Y}[t]\right] \\
&\quad +T(b_1+b_2)
\end{aligned}$$

Lemma 1:

$$\begin{aligned}
&\mathbb{E}[K\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) \\
&\qquad -\langle\, \mathbf{Q}[t+\tau], \mathbf{X}[t+\tau]\rangle \mid \mathbf{Y}[t+\tau]] \\
&\geq K\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) - \langle\, \mathbf{Q}[t+\tau], \overset{\star}{\mathbf{x}}(\varepsilon)\rangle,
\end{aligned}$$

where $\overset{\star}{\mathbf{x}}(\varepsilon)$ is defined in Definition 5.

Proof: [Lemma 1] Note that the congestion control mechanism picks $X_n^{(d)}[t]$ to solve (9). Since $\overset{\star}{x}{}_n^{(d)} \in [0, M]$, we have that $X_n^{(d)}[t]$ satisfies

$$\begin{aligned}
&K\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) - \langle\, \mathbf{Q}[t+\tau], \mathbf{X}[t+\tau]\rangle \\
&\quad \geq K\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) - \langle\, \mathbf{Q}[t+\tau], \overset{\star}{\mathbf{x}}(\varepsilon)\rangle,
\end{aligned}$$

for all τ. ■

We use the result of Lemma 1 in the previous expression to get
$\Delta V_t^{(T)}(\mathbf{Y})$

$$\begin{aligned}
&\leq 2K\sum_{\tau=0}^{T-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) \mid \mathbf{Y}[t]\right] \\
&\quad -2TK\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) \\
&\quad +2\sum_{\tau=0}^{T-1}\mathbb{E}\left[\langle\, \mathbf{Q}[t+\tau], \overset{\star}{\mathbf{x}}(\varepsilon)\right. \\
&\qquad\qquad \left. +\mathbf{S}_{in}[t+\tau] - \mathbf{S}_{out}[t+\tau] \mid \mathbf{Y}[t]\right] \qquad (12) \\
&\quad +T(b_1+b_2)
\end{aligned}$$

$$\begin{aligned}
&\leq 2K\sum_{\tau=0}^{T-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[t+\tau]) \mid \mathbf{Y}[t]\right] \\
&\quad -2TK\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) \\
&\quad -2cT\left(\frac{\varepsilon}{d_{max}} - \gamma - \sqrt{\frac{\psi}{\delta}}\right)\sum_{n,d} Q_n^{(d)}[t] \\
&\quad +2B_1 + T(b_1+b_2),
\end{aligned}$$

where the last inequality follows from the application of Theorem 1 to (12).

Next, we take the expectation of both sides of the inequality to eliminate the conditioning, and then take the telescoping sum of P such T-step drifts to obtain
$\mathbb{E}\left[V(\mathbf{Y}[PT]) - V(\mathbf{Y}[0])\right]$

$$\leq 2K \sum_{k=0}^{PT-1} \mathbb{E}\left[\sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(X_n^{(d)}[k])\right] \quad (13)$$

$$-2KPT \sum_{n,d\in\mathcal{N}} \mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) \quad (14)$$

$$-2cT\rho \sum_{p=0}^{P-1}\sum_{n,d} \mathbb{E}[Q_n^{(d)}[pT]] + 2B_1P \quad (15)$$

$$+PT(b_1+b_2), \quad (16)$$

where we define $\rho := \left(\frac{\varepsilon}{d_{max}} - \gamma - \sqrt{\frac{\psi}{\delta}}\right)$. Noting that $V(\mathbf{Y}) \geq 0$ for all feasible $\mathbf{Y}$, and re-arranging the terms in this expression, we can obtain

$$\frac{1}{P}\sum_{p=0}^{P-1}\sum_{n,d}\mathbb{E}[Q_n^{(d)}[pT]] \leq \frac{K\sum_{n,d}\mathcal{U}_{n,d}(M) + \frac{B_1}{T} + \frac{(b_1+b_2)}{2} + \frac{\mathbb{E}[V(\mathbf{Y}[0])]}{2PT}}{\rho c}$$

Also noting that $Q_n^{(d)}[t+k] \leq Q_n^{(d)}[t] + kM$ for each n, d, we have

$$\frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{n,d\in\mathcal{N}}\mathbb{E}[Q_n^{(d)}[pT+\tau]] \leq MTN^2,$$

which, when combined with the previous inequality, yields

$$\begin{aligned}
&\limsup_{P\to\infty}\frac{1}{PT}\sum_{k=0}^{PT-1}\sum_{n,d\in\mathcal{N}}\mathbb{E}[Q_n^{(d)}[k]] \\
&\leq \limsup_{P\to\infty}\sum_{p=0}^{P-1}\sum_{n,d}\mathbb{E}[Q_n^{(d)}[pT]] + MTN^2 \\
&\leq \frac{K\sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(M) + \frac{B_1}{T} + \frac{(b_1+b_2)}{2}}{\rho c} + MTN^2 \\
&\leq C_1K,
\end{aligned}$$

for some C_1 when K is large enough.

Next, we re-organize the terms in (13)-(16) in a different way to obtain

$$\frac{1}{PT}\sum_{k=0}^{PT-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(X_n^{(d)}[k])\right] \geq \sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) - \frac{\frac{B_1}{T} + \frac{(b_1+b_2)}{2} - \frac{\mathbb{E}[V(\mathbf{Y}[0])]}{2PT}}{K}.$$

Also revoking Jensen's inequality, we have

$$\frac{1}{PT}\sum_{k=0}^{PT-1}\mathbb{E}\left[\sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(X_n^{(d)}[k])\right] \leq \sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}\left(\frac{1}{PT}\sum_{k=0}^{PT-1}\mathbb{E}[X_n^{(d)}[k]]\right).$$

Combining these two results as letting $P\to\infty$, we have

$$\sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(\bar{x}_n^{(d)}) \geq \sum_{n,d\in\mathcal{N}}\mathcal{U}_{n,d}(\overset{\star}{x}{}_n^{(d)}(\varepsilon)) - \frac{C_2}{K},$$

where $\bar{x}_n^{(d)}$ is as defined in the statement of the theorem, and C_2 is a bounded number. This completes the proof. ■

Theorem 2 reveals the effect of the errors and relaxation in the operation of the scheduler. In particular, we see that, when $\gamma = \psi = 0$, and $\delta > 0$, the cross-layer scheme achieves optimal performance. Also, we observe that the effect of ψ can be detrimental unless it is significantly smaller than δ. In comparison, the effect of γ appears to be milder if it can be made small. Ideally, we would like to design schedulers with $\gamma = \psi = 0$ in which case optimal performance can be guaranteed. In the next section, we propose one such scheduler that is applicable to second order interference model, but can also be extended to higher order interference models.

V. Algorithm Design

In Section III, we studied the throughput and fairness properties of a cross-layer mechanism that can be applied to a large class of interference models. We observed that a scheduler with $\delta > 0$, and $\gamma = \psi = 0$ achieves optimal performance. In this section, we focus on the secondary interference model and outline a distributed low-complexity algorithm with these parameters. We will also note a modification to our algorithm that yields $\gamma > 0$.

Our 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 $\mathbf{R}[t]$ satisfying (6) in finite time. The COMPARE algorithm compares the total weights of the old schedule $\mathbf{S}[t]$ with the new schedule $\mathbf{R}[t]$ according to (7) 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.

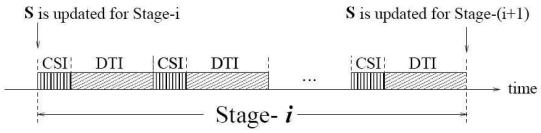


Fig. 1. Division of time into data transmission and control signalling intervals.

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 2 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 $\mathbf{R}$ with the property that any feasible allocation has a positive probability of being chosen as required by (6). 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 6 (*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 $\mathbf{R}$. $\diamond$

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[5] at its termination. This does not influence the results of Theorem 2, 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 $\mathbf{R}$ 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 $(\min(\alpha, 1-\alpha)/d_{max})^N > 0$, where d_{max} is the maximum degree[6] of $\mathcal{G}$. In particular, since $\overset{\star}{\mathbf{S}}_{\mathbf{W}}[t]$ is a feasible schedule, we have $\mathbb{P}(\mathbf{R}[t] = \overset{\star}{\mathbf{S}}_{\mathbf{W}}[t]) \geq (\min(\alpha, 1-\alpha)/d_{max})^N > 0$.

(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 $\mathbf{R}$ 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

[5] 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.

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

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 $W \in \mathcal{S}$, 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 $\geq (\alpha/d_{max})^{|W|}$), then the resulting schedule will be exactly W. Hence, the probability that PICK yields a given feasible schedule W is $\geq (\alpha/d_{max})^{|W|}(1-\alpha)^{N-|W|} \geq (\min(\alpha, 1-\alpha)/d_{max})^{N}$, where the last step follows from $d_{max} \geq 1$.

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 2 to hold. At the end of PICK, $\mathbf{R}$ 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 $\mathbf{R}$.

B. COMPARE Algorithm

In this section, we propose and analyze a distributed algorithm that compares the total weight associated with two feasible schedules, $\mathbf{S}[t]$ and $\mathbf{R}[t]$, 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 $\mathbf{S}$ and $\mathbf{R}$ which contains information about interfering links in the two schedules. The conflict graph, $\mathcal{G}'(\mathbf{S},\mathbf{R}) = (\mathcal{N}', \mathcal{L}')$, of $\mathbf{S}$ and $\mathbf{R}$ can be generated as follows[7]: Each link l in $\mathbf{S} \cup \mathbf{R}$ corresponds to a node in the conflict graph, and if links $l_1 \in \mathbf{S}$ and $l_2 \in \mathbf{R}$ 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 $\mathbf{S}$ and $\mathbf{R}$ are feasible schedules, no two links in the same schedule ($\mathbf{S}$ or $\mathbf{R}$) 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 (3)], 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 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.

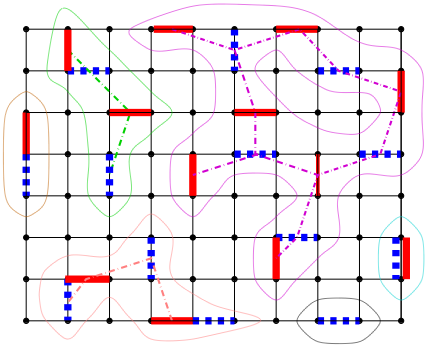


Fig. 2. 8x10 grid network example with two feasible schedules indicated by solid and dashed bold links. The conflict graph decomposes into 6 disconnected components.

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, $\mathbf{S}$ and $\mathbf{R}$. In the figure, solid bold links belong to schedule $\mathbf{S}$, while dashed bold links are in $\mathbf{R}$. 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 $\mathbf{S}$ are drawn as circular dots, while links in $\mathbf{R}$ 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, $\mathbf{S}$ and $\mathbf{R}$, because different connected components may prefer different schedules. This merging operation will result in a schedule that is better than both $\mathbf{S}$ and $\mathbf{R}$.

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

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

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[8]. In order to compare two link IDs, we use lexicographical ordering[9].

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. [10]) 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.

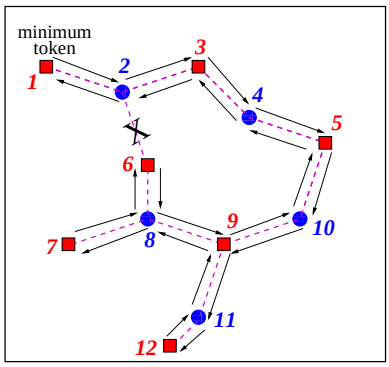


Fig. 3. A connected component of the conflict graph from which the link crossed is eliminated to obtain a spanning tree. The path of the minimum token is indicated with arrows. The nodes are labeled with numbers for future reference.

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 already 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.

Proposition 2: Consider the conflict graph $\mathcal{G}' = (\mathcal{N}', \mathcal{L}')$, and let d'_{max} denote the maximum degree of $\mathcal{G}'$. The FIND SPANNING TREE Procedure finds a spanning tree of all components of the conflict graph in $O(d'_{max}L')$ time[10], and with $O(d'_{max}N')$ 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 $O(d'_{max})$ message exchanges. Since, an operation of $O(d'_{max})$ operations must be performed for $2L'$ times, we need $O(d'_{max}L')$ time for the operation to complete. However, each node will only transmit $O(d'_{max})$ 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 $O(d'_{max}N')$. ■

We note that the FIND SPANNING TREE procedure that we described here is deterministic and achieves $\gamma = 0$ in the context of the generic cross-layer scheme of Definition 3. Alternatively, a randomized gossip style mechanism can be used that yields $\gamma > 0$ (see [28]). Theorem 2 can be used to understand the fairness characteristics of both the these approaches. While our approach lead to optimal performance, the latter approach may be easier to implement with acceptable degree of suboptimality.

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

[8]In [16], it was shown that unique IDs are required to be able to find a spanning tree in a distributed fashion.

[9]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 $ID(n,m) < ID(i,j)$.

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

leaves up to the root of the tree (i.e. COMMUNICATE Procedure) so that the schedule with the higher weight is chosen (cf. (7)), 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:

COMMUNICATE: The leaves communicate their weights to their parents. If the parent is in $\mathbf{S}$ it adds its weight to the sum of the weights announced by its children. If, on the other hand, it is in $\mathbf{R}$ 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 $\sum_{l\in\mathbf{S}} w_l - \sum_{l\in\mathbf{R}} w_l$. Depending on whether the root's weight is positive or negative, the root decides $\mathbf{S}$ or $\mathbf{R}$, respectively, as the better schedule, and broadcasts its decision down the tree.

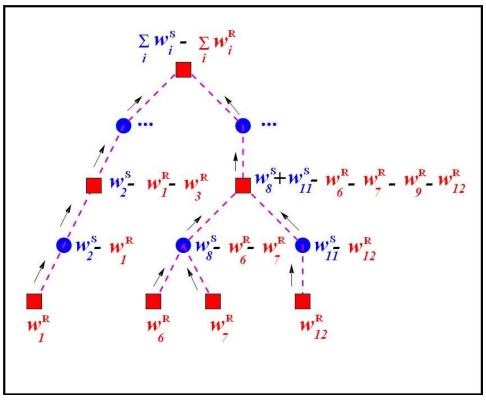


Fig. 4. The iterative communication of the weights of the two schedules from the leaves to the root for the spanning tree of Figure 3.

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'_{max} denote the maximum degree of $\mathcal{G}'$. The COMMUNICATE & DECIDE procedure correctly finds the schedule with the larger weight in $O(d'_{max}L')$ 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 $\mathbf{S}$ and $\mathbf{R}$ only for the subtree rooted at n'. Thus, sum at the root of the spanning tree is the difference of two weights of schedule $\mathbf{S}$ and $\mathbf{R}$. 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 $O(d'_{max})$ time. Thus, the whole algorithm terminates in $O(L'd'_{max})$ 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: $L' < N^2$, $d'_{max} < N$. Propositions 1, 2 and 3, together with Theorem 2 yields the following result.

Theorem 3: 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 $\mathbf{S}$ and $\mathbf{R}$ 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 $\mathbf{S}$ communicate, while in the second interval only nodes that make up $\mathbf{R}$ 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.

VI. SIMULATIONS

In this section, we provide simulation results for the distributed algorithms developed in Section V 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 [21], [38]).

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

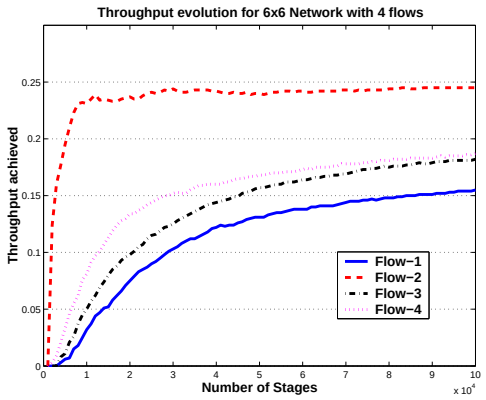


Fig. 5. The throughput evolution of the 6x6 network for $K = 100$, $\gamma_i = 0.5$.

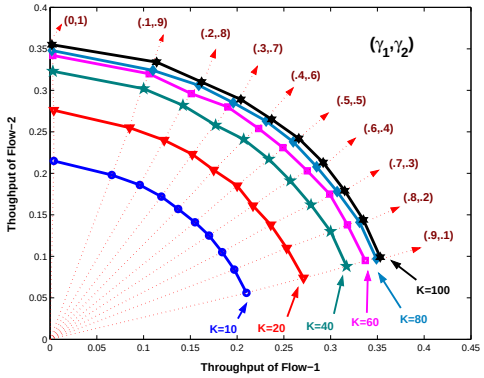


Fig. 6. Throughputs of flows with varying K and (γ_1, γ_2).

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 mean 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 2 implies that this region grows at a decreasing rate, until it converges to the stability region $\mathcal{C}$.

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.

VII. 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 allows for various imperfections and relaxation in its operation which facilitates its distributed operation in wireless networks. We studied the stability and fairness characteristics of the generic cross-layer algorithm, and explicitly characterized the effect of different type of imperfections on its performance. We saw that certain types of imperfections are more detrimental than other, which revealed the critical components in the design of algorithms.

Based on this foundation, 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.

Appendix

We defined the notion of capacity (stability) region in Definition 2. A characterization of this region in terms of flow conservation and feasibility constraints is provided by Tassiulas and Ephremides in their seminal work [41], which is reproduced in the following proposition to be used in the proof of Theorem 1.

Proposition 4: Let $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ be a given network and $\mathcal{S}$ be the set of feasible allocations. The capacity (or stability) region $\mathcal{C}$ of the network is given by the set of vectors $\mathbf{r} = (r_n^{(d)})_{n,d\in\mathcal{N}}$ for which there exists $z_{(n,m)}^{(d)} \geq 0$ for all $(n, m) \in \mathcal{L}$ and $d \in \mathcal{N}$, 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 $d \in \mathcal{N}\backslash\{n\}$, we have

$$r_n^{(d)} + \sum_{k:(k,n)\in\mathcal{L}} z_{(k,n)}^{(d)} = \sum_{m:(n,m)\in\mathcal{L}} z_{(n,m)}^{(d)},$$

(C2) $$\left[\sum_{d\in\mathcal{N}} \left(z_{(n,m)}^{(d)} + z_{(m,n)}^{(d)}\right)\right]_{(n,m)\in\mathcal{L}} \in Conv(\mathcal{S}).$$ [11]

Proof of Theorem 1:

Before the start the proof, we note that it closely follows the technique of [27], except that it is extended to multi-hop flows and more general arrival processes. The multi-hop extension adds a routing component to the mechanism and add some technical complications to the proof. More importantly, in this work we further include congestion control into the framework of [27] to investigate the fairness characteristics of the joint congestion control, scheduling and routing mechanism.

We start by deriving an upper bound on the single-step mean drift of the Lyapunov function, $\Delta V_t^{(1)}(\mathbf{Y})$, for a given $\mathbf{Y} =$

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

$(\mathbf{Q},\mathbf{S})$.

$$\begin{aligned}
&\Delta V_t^{(1)}(\mathbf{Y}) \\
&= \mathbb{E}\left[\|\mathbf{Q}[t+1]\|_2^2 - \|\mathbf{Q}[t]\|_2^2 \mid \mathbf{Y}[t]\right] \\
&\leq \sum_{n,d}\mathbb{E}\Bigg[\Bigg(\left(Q_n^{(d)}[t] - S_{out(n)}^{(d)}[t]\right)^+ + X_n^{(d)}[t] \\
&\qquad +S_{into(n)}^{(d)}[t]\Bigg)^2 - (Q_n^{(d)}[t])^2 \mid \mathbf{Y}[t]\Bigg] \\
&= \sum_{n,d}\mathbb{E}\Bigg[\left(Q_n^{(d)}[t] - S_{out(n)}^{(d)}[t] + U_{out(n)}^{(d)}[t] + X_n^{(d)}[t]\right. \\
&\qquad \left.+S_{into(n)}^{(d)}[t]\right)^2 - (Q_n^{(d)}[t])^2 \mid \mathbf{Y}[t]\Bigg] \\
&= \sum_{n,d}\mathbb{E}\Bigg[\left(Q_n^{(d)}[t] - S_{out(n)}^{(d)}[t] + X_n^{(d)}[t]\right. \\
&\qquad \left.+S_{into(n)}^{(d)}[t]\right)^2 - (Q_n^{(d)}[t])^2 \mid \mathbf{Y}[t]\Bigg] \qquad (17) \\
&\quad +\sum_{n,d} 2\mathbb{E}\Bigg[U_{out(n)}^{(d)}[t]\left(Q_n^{(d)}[t] - S_{out(n)}^{(d)}[t] + X_n^{(d)}[t]\right. \\
&\qquad \left.+S_{into(n)}^{(d)}[t]\right) + \left(U_{out(n)}^{(d)}[t]\right)^2 \mid \mathbf{Y}[t]\Bigg] \qquad (18)
\end{aligned}$$

where $U_{out(n)}^{(d)}[t]$ denotes the amount of unused service by node n to transmit packets of type d in slot t. Note that $U_{out(n)}^{(d)}[t]$ can be non-zero only when $Q_n^{(d)}[t]$ is low. Also, since the service rate over each link is upper-bounded by one, $U_{out(n)}^{(d)}[t]$ must also be upper-bounded by the maximum degree d_{max} of the network. First, we show that (18) is upper-bounded.

$$\begin{aligned}
(18) &\leq \sum_{n,d}\Big(2\mathbb{E}\left[U_{out(n)}^{(d)}[t]Q_n^{(d)}[t] \mid \mathbf{Y}[t]\right] \\
&\qquad +4d_{max} + 2\lambda_n^{(d)} + d_{max}^2\Big) \\
&\leq N^2(3d_{max}^2 + 6d_{max}) =: b_1,
\end{aligned}$$

where, in the last step, we used the fact that $U_{out(n)}^{(d)}[t] \leq d_{max}$ and that $\lambda_n^{(d)} \leq d_{max}$.

Next, we study (17). We can re-write it in inner-product form after cancelations

$$\begin{aligned}
(17) &= 2\mathbb{E}\left[\langle\, \mathbf{Q}[t], \mathbf{S}_{in}[t] + \mathbf{X}[t] - \mathbf{S}_{out}[t]\rangle\right. \\
&\qquad \left.+\|\mathbf{S}_{in}[t] + \mathbf{X}[t] - \mathbf{S}_{out}[t]\|_2^2 \mid \mathbf{Y}[t]\right] \\
&\leq 2\mathbb{E}\left[\langle\, \mathbf{Q}[t], \mathbf{S}_{in}[t] + \mathbf{X}[t] - \mathbf{S}_{out}[t]\rangle \mid \mathbf{Y}[t]\right] + b_2, \qquad (19)
\end{aligned}$$

where b_2 is a finite constant since: the service rate into or out of any node is bounded by d_{max}; and the second moment of the arrival process is assumed to be bounded.

Next, we study the expectation in (19) in further detail. We omit the the time index $[t]$ in the following derivation for notational convenience.

$$\begin{aligned}
\mathbb{E}\left[\langle\, \mathbf{Q}, \mathbf{S}_{in} + \mathbf{X} - \mathbf{S}_{out}\rangle \mid \mathbf{Y}\right] &= \langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{in} + \lambda - \overset{\star}{\mathbf{S}}_{out}\rangle \\
&\quad +\langle\, \mathbf{Q}, \mathbf{S}_{in} - \overset{\star}{\mathbf{S}}_{in} + \mathbf{S}_{out} - \overset{\star}{\mathbf{S}}_{out}\rangle \qquad (20)
\end{aligned}$$

where $\overset{\star}{\mathbf{S}}_{\mathbf{W}}$ is chosen according to (4). Since $\lambda \in \mathcal{C}(\varepsilon)$, Proposition 4 implies that there exists a non-negative vector $\hat{\mathbf{S}} = \left(\hat{S}_{(n,m)}^{(d)}\right)_{(n,m)\in\mathcal{L}}^{d\in\mathcal{N}}$ such that: $\left[\sum_{d\in\mathcal{N}} \hat{S}_{(n,m)}^{(d)}\right]_{(n,m)\in\mathcal{L}} \in Conv(\mathcal{S})$; and

$$\lambda_n^{(d)} + \hat{S}_{into(n)}^{(d)} = \hat{S}_{out(n)}^{(d)} - \varepsilon, \qquad \forall n, d \neq n,$$

which can be written compactly as $\lambda = \hat{\mathbf{S}}_{out} - \hat{\mathbf{S}}_{in} - \varepsilon\mathbf{1}$ in vector form, where $\mathbf{1}$ is a vector of all ones. Substituting this into the first inner product in (20) yields.

$$\begin{aligned}
\langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{in} + \lambda - \overset{\star}{\mathbf{S}}_{out}\rangle &= \langle\, \mathbf{Q}, \hat{\mathbf{S}}_{out} - \hat{\mathbf{S}}_{in}\rangle \\
&\quad -\langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{out} - \overset{\star}{\mathbf{S}}_{in}\rangle - \varepsilon\langle\, \mathbf{Q}, \mathbf{1}\rangle
\end{aligned}$$

Note that

$$\begin{aligned}
\langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{out} - \overset{\star}{\mathbf{S}}_{in}\rangle &= \sum_{n,d} Q_n^{(d)}\left(\overset{\star}{\mathbf{S}}_{out(n)}^{(d)} - \overset{\star}{\mathbf{S}}_{into(n)}^{(d)}\right) \\
&= \sum_{(n,m)\in\mathcal{L}}\sum_d \overset{\star}{S}_{(n,m)}^{(d)}\left(Q_n^{(d)} - Q_m^{(d)}\right) \\
&= \sum_{(n,m)\in\mathcal{L}} \overset{\star}{S}_{(n,m)} \max_d\left|Q_n^{(d)} - Q_m^{(d)}\right| \\
&\overset{(a)}{\geq} \sum_{(n,m)\in\mathcal{L}} \hat{S}_{(n,m)} \max_d\left|Q_n^{(d)} - Q_m^{(d)}\right| \\
&\geq \sum_{(n,m)\in\mathcal{L}}\sum_d \hat{S}_{(n,m)}^{(d)}\left(Q_n^{(d)} - Q_m^{(d)}\right) \\
&= \langle\, \mathbf{Q}, \hat{\mathbf{S}}_{out} - \hat{\mathbf{S}}_{in}\rangle,
\end{aligned}$$

where the inequality (a) follows from (4). Substituting this in the previous expression yields

$$\begin{aligned}
\langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{in} + \lambda - \overset{\star}{\mathbf{S}}_{out}\rangle &\leq -\varepsilon\langle\, \mathbf{Q}, \mathbf{1}\rangle \\
&\leq -\frac{\varepsilon}{d_{max}}\langle\, \mathbf{Q}, \overset{\star}{\mathbf{S}}_{out} - \overset{\star}{\mathbf{S}}_{in}\rangle,
\end{aligned}$$

where the last inequality follows from the fact that $\overset{\star}{S}_{out(n)} - \overset{\star}{S}_{in(n)} \leq d_{max}$ for all n. We substitute this upper bound in (20) with the new notation: $\Psi[t] := \langle\, \mathbf{Q}[t], \mathbf{S}_{out}[t] - \mathbf{S}_{in}[t]\rangle$, and $\overset{\star}{\Psi}[t] := \langle\, \mathbf{Q}[t], \overset{\star}{\mathbf{S}}_{out}[t] - \overset{\star}{\mathbf{S}}_{in}[t]\rangle$, and $\tilde{\Psi}[t] := \langle\, \mathbf{Q}[t], \mathbf{R}_{out}[t] - \mathbf{R}_{in}[t]\rangle$.

$$\begin{aligned}
&\mathbb{E}\left[\langle\, \mathbf{Q}[t], \mathbf{S}_{in}[t] + \mathbf{X}[t] - \mathbf{S}_{out}[t]\rangle \mid \mathbf{Y}[t] = \mathbf{Y}\right] \\
&\qquad \leq -\frac{\varepsilon}{d_{max}}\overset{\star}{\Psi}[t] + \overset{\star}{\Psi}[t] - \Psi[t]
\end{aligned}$$

We use this bound in (19) and bound the T-step mean drift as $\Delta V_t^{(T)}(\mathbf{Y})$

$$\begin{aligned}
&\leq -2\frac{\varepsilon}{d_{max}}\sum_{\tau=0}^{T-1}\mathbb{E}[\overset{\star}{\Psi}[t+\tau] \mid \mathbf{Y}[t] = \mathbf{Y}] + Tb_2 \qquad (21) \\
&\quad +\sum_{\tau=0}^{T-1}\mathbb{E}[\overset{\star}{\Psi}[t+\tau] - \Psi[t+\tau] \mid \mathbf{Y}[t] = \mathbf{Y}] \qquad (22)
\end{aligned}$$

To bound (21) note that

$$\begin{aligned}
\sum_{\tau=0}^{T-1}\mathbb{E}[\overset{\star}{\Psi}[t+\tau] \mid \mathbf{Y}[t] = \mathbf{Y}] &\geq \sum_{\tau=0}^{T-1}\left(\overset{\star}{\Psi}[t] - \tau b_3\right) \\
&\geq T\overset{\star}{\Psi}[t] - \frac{T^2}{2}b_3, \qquad (23)
\end{aligned}$$

where the first inequality follows from the fact that in a single time slot, each queue can change by at most a bounded value, and therefore there exists a constant b_3, such that $|\overset{\star}{\Psi}[\tau+1]-\overset{\star}{\Psi}[\tau]| \leq b_3$ for any τ.

Next, we are interested in upper-bounding (22). For notational convenience, let us define $\nabla[t] :=\overset{\star}{\Psi}[t]-\Psi[t]$. Hence, we are interested in upper-bounding $\sum_{\tau=0}^{T-1}\mathbb{E}[\nabla[t+\tau]|\mathbf{Y}[t]=\mathbf{Y}]$. To that end, let us define

$$\begin{aligned} T_0 &= \inf\{\tau \geq 0 : \mathbf{R}[t+\tau] =\overset{\star}{\mathbf{S}}[t+\tau]\} \\ T_1 &= \inf\{\tau \geq T_0 : \langle \mathbf{Q}[t+\tau], \mathbf{S}_{out}[t+\tau]-\mathbf{S}_{in}[t+\tau]\rangle \\ &\quad < \max\left(\langle \mathbf{Q}[t+\tau], \mathbf{S}_{in}[t+\tau-1]-\mathbf{S}_{out}[t+\tau-1]\rangle,\right. \\ &\quad \left.(1-\gamma)\langle \mathbf{Q}[t+\tau], \mathbf{R}_{out}[t+\tau]-\mathbf{R}_{in}[t+\tau]\rangle\right)\}. \end{aligned}$$

Thus, T_0 is the first slot after t when the random schedule $\mathbf{R}$ picked according to (6) is equal to the optimum schedule; and T_1 is the first slot after T_0 when the condition in (7) is violated. Note that in the interval between T_0 and T_1, the system is well-behaved, and no undesired event such as that in (7) occurs. Finally, let us define $T_2 := T-\min(T,T_1)$ as the remaining time after T_1 until the end of T slots, if any. The idea is to show that if T is sufficiently large, the duration between T_0 and T_1 will dominate the interval of duration T. Next, we make this argument rigorous.

First note that for any $\tau \geq 0$, we have

$$\nabla[t+\tau] \leq 2\overset{\star}{\Psi}[t+\tau] \leq 2\overset{\star}{\Psi}[t]+2\tau b_3.$$

Next, note that at $\tau = T_0$, we have

$$\begin{aligned} &\langle \mathbf{Q}[t+T_0], \mathbf{S}_{out}[t+T_0]-\mathbf{S}_{in}[t+T_0]\rangle \\ &\quad \geq (1-\gamma)\langle \mathbf{Q}[t+T_0], \overset{\star}{\mathbf{S}}_{out}[t+T_0]-\overset{\star}{\mathbf{S}}_{in}[t+T_0]\rangle \end{aligned}$$

due to (7). This is the same as

$$\begin{aligned} \nabla[t+T_0] &\leq \gamma\overset{\star}{\Psi}[t+T_0] \\ &\leq \gamma\overset{\star}{\Psi}[t+T_0]+\gamma T_0 b_3 \end{aligned}$$

Then, in the interval when $\tau \in [T_0, T_1]$, we have that

$$\nabla[t+\tau] \leq \gamma\overset{\star}{\Psi}[t]+2\gamma\tau b_3$$

Hence, we can write

$$\begin{aligned} \sum_{\tau=0}^{T-1}\nabla[t+\tau] &\leq 2(\overset{\star}{\Psi}[t]+\gamma\tau b_3)(\min(T_0,T)+T_2) \\ &\quad +\gamma\overset{\star}{\Psi}[t]T+2\gamma T^2 b_3. \end{aligned}$$

Note that we have $\mathbb{E}[\min(T_0,T)] \leq 1/\delta$. Also, $\mathbb{E}[\min(T,T_1)] \geq (1-\psi T)T$. Therefore, $\mathbb{E}[T_2] \leq \psi T^2$. Thus, we can write

$$\begin{aligned} \mathbb{E}\left[\sum_{\tau=0}^{T-1}\nabla[t+\tau] \mid \mathbf{Y}[t]=\mathbf{Y}\right] &\leq T\overset{\star}{\Psi}[t]\left(\psi T+\frac{1}{\delta T}+2\gamma\right) \\ &\quad +2Tb_3\left(\frac{1}{\delta}+\psi T^2\right) \\ &\quad +2\gamma T^2 b_3 \end{aligned}$$

Substituting (23) in (21), and the previous upper bound in (22) yields

$$\begin{aligned} \Delta V_t^{(T)}(\mathbf{Y}) &\leq -T\overset{\star}{\Psi}[t]\left(2\frac{\varepsilon}{d_{max}}-\psi T-\frac{1}{\delta T}-2\gamma\right) \\ &\quad +2\left(\frac{\varepsilon}{d_{max}}+\gamma\right)T^2 b_3 \\ &\quad +2\left(\frac{1}{\delta}+\psi T^2\right)Tb_3 \end{aligned}$$

Letting $T = \frac{1}{\sqrt{\delta\psi}}$, we have

$$\begin{aligned} \Delta V_t^{(T)}(\mathbf{Y}) &\leq -2T\overset{\star}{\Psi}[t]\left(\frac{\varepsilon}{d_{max}}-\sqrt{\frac{\psi}{\delta}}-\gamma\right)+B_1 \\ &\leq -cT\left(\frac{\varepsilon}{d_{max}}-\gamma-\sqrt{\frac{\psi}{\delta}}\right)\sum_{n,d}Q_n^{(d)}+B_1, \end{aligned}$$

where c and B_1 are bounded positive valued numbers. This completes our proof.

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] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting. Scheduling in a queueing system with asynchronously varying service rates. *Probability in the Engineering and Informational Sciences*, 18:191–217, 2004.

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

[4] S. Asmussen. *Applied Probability and Queues*. Springer-Verlag, New York, NY, 2003.

[5] M. Bayati, B. Prabhakar, D. Shah, and M. Sharma. Iterative scheduling algorithms. In *Proceedings of IEEE INFOCOM*, 2007.

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

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

[8] 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.

[9] 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.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. M.I.T. Press, McGraw-Hill Book Company, London, England, 2001.

[11] J. Dai and B. Prabhakar. The throughput of switches with and without speed-up. In *Proceedings of INFOCOM*, 2000.

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

[13] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proceedings of IEEE Infocom*, volume 3, pages 1794–1803, Miami, FL, March 2005.

[14] 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.

[15] A. Eryilmaz, R. Srikant, and J. R. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Transactions on Networking*, 13:411–425, April 2005.

[16] 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.

[17] P. Giaccone, B. Prabhakar, and D. Shah. Randomized scheduling algorithms for high-aggregate bandwidhth switches. *IEEE Journal on Selected Areas in Communications*, 21(4):546–559, 2003.

[18] C. Joo, X. Lin, and N. Shroff. Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks. In *Proceedings of IEEE INFOCOM*, 2008.

[19] K. Jung and D. Shah. Low delay scheduling in wireless networks. In *Proceedings of ISIT*, 2007.

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

[21] 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.

[22] I. Keslassy and N. McKeown. Analysis of scheduling algorithms that provide 100% throughput in input-queued switches. In *Proceedings of the Allerton Conference on Control, Communications and Computing*, 2001.

[23] 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.

[24] 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.

[25] 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.

[26] M. Marsan, E. Leonardi, M. Mellia, and F. Neri. On the stability of input-buffer cell switches with speed-up. In *Proceedings of INFOCOM*, 2000.

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

[28] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings IEEE PODC*, Denver, 2006.

[29] 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.

[30] 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.

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

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

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

[34] D. Shah. Stable algorithms for input queued switches. In *Proceedings of the Allerton Conference on Control, Communications and Computing*, 2001.

[35] D. Shah and D. J. Wischik. Optimal scheduling algorithms for input-queued switches. In *Proceedings of IEEE INFOCOM*, 2006.

[36] D. Shah and D. J. Wischik. Heavy traffic analysis of optimal scheduling algorithms for networks, 2007. submitted for publication.

[37] 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.

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

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

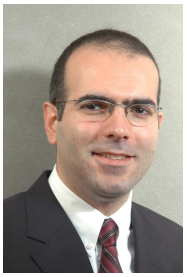
[40] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proceedings of IEEE Infocom*, pages 533–539, 1998.

[41] 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.

[42] 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.

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

[44] 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.



Atilla Eryilmaz (S '00-M '06) received his B.S. degree in Electrical and Electronics Engineering from Boğaziçi University, Istanbul, in 1999, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2001 and 2005, respectively. Between 2005 and 2007, he worked as a Postdoctoral Associate at the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. He is currently an Assistant Professor of Electrical and Computer Engineering at the Ohio State University. His research interests include communication networks, optimal control of stochastic networks, optimization theory, distributed algorithms, stochastic processes and network coding.

Asu Ozdaglar received the B.S. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, in 1996, and the S.M. and the Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1998 and 2003, respectively. Since 2003, she has been a member of the faculty of the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, where she is currently the Class of 1943 Career Development Associate Professor. She is also a member of the Laboratory for Information and Decision Systems and the Operations Research Center. She is the recipient of the MIT Graduate Student Council Teaching award, the NSF Career award, and the 2008 Donald P. Eckman award of the American Automatic Control Council. Her research interests include optimization theory, with emphasis on nonlinear programming and convex analysis, game theory, distributed optimization methods, and network optimization and control.

Devavrat Shah is currently an assistant professor with EECS, MIT since Fall 2005. He is a member of the Laboratory of Information and Decision Systems (LIDS). He received his BTech degree in Computer Science & Eng. from IIT-Bombay in 1999 with the honor of the President of India Gold Medal. He received his Ph.D. from the Computer Science department, Stanford University in October 2004. He was a post-doc in the Statistics department at Stanford in 2004-05. He was co-awarded the IEEE INFOCOM best paper award in 2004 and ACM SIGMETRIC/Performance best paper awarded in 2006. He received 2005 George B. Dantzig best disseration award from the INFORMS. He received NSF CAREER award in 2006. His research interests include network algorithms, stochastic networks, network information theory and statistical inference.

Eytan Modiano received his B.S. degree in Electrical Engineering and Computer Science from the University of Connecticut at Storrs in 1986 and his M.S. and PhD degrees, both in Electrical Engineering, from the University of Maryland, College Park, MD, in 1989 and 1992 respectively. He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow during 1992-1993. Between 1993 and 1999 he was with MIT Lincoln Laboratory where he was the project leader for MIT Lincoln Laboratory's Next Generation Internet (NGI) project. Since 1999 he has been on the faculty at MIT; where he is presently an Associate Professor. His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks.