Distributed Link Scheduling With Constant Overhead

Loc X. Bui, Sujay Sanghavi, and R. Srikant, *Fellow, IEEE*

***Abstract*—This paper proposes a new class of simple, distributed algorithms for scheduling in multihop wireless networks under the primary interference model. The class is parameterized by integers $k \geq 1$. We show that algorithm k of our class achieves $k/(k+2)$ of the capacity region, for every $k \geq 1$. The algorithms have small and constant worst-case overheads. In particular, algorithm k generates a new schedule using a) time less than $4k+2$ round-trip times between neighboring nodes in the network and b) at most three control transmissions by any given node for any k. The control signals are explicitly specified and face the same interference effects as normal data transmissions. Our class of distributed wireless scheduling algorithms are the first ones guaranteed to achieve any fixed fraction of the capacity region while using small and constant overheads that do not scale with network size. The parameter k explicitly captures the tradeoff between control overhead and throughput performance and provides a tuning-knob protocol designers can use to harness this tradeoff in practice.**

***Index Terms*—Congestion control, distributed algorithms, matchings, multihop, primary interference, wireless networks scheduling.**

I. Introduction

THIS PAPER presents novel distributed algorithms for scheduling of transmissions in multihop wireless networks. The algorithms represent the first instance in which any arbitrary fraction of the capacity region can be achieved with constant overhead, independent of the size of the network but possibly dependent on the chosen fraction. In addition, our algorithms are very simple. We now motivate our work and summarize our contributions.

The task of wireless scheduling is challenging due to the simultaneous presence of two characteristics: interference between transmissions and the need for practical distributed implementation. Interference effects result in a fundamental upper limit on the data rates that any scheduling algorithm—distributed or otherwise—can hope to achieve. This fundamental limit, or capacity region, serves as a benchmark against which the performance of various distributed scheduling algorithms can be compared.

In practice, the need for distributed implementation invariably leads to an overhead, as the same resources that could have been used for data transmission have to, instead, be wasted on control signals in an effort to combat interference. Most of the currently proposed scheduling algorithms, which we survey and compare in Section II, do not explicitly take overheads into account. As a result, it may be the case (especially for large networks) that after using a large—and unaccounted for—portion of resources for control signaling, the algorithms perform well with regards to the benchmark in the *remaining* portion used for data transmission.

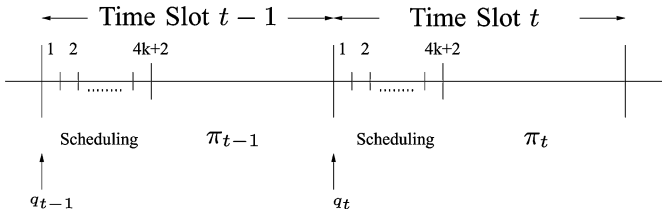

Fig. 1. Each scheduling cycle is divided into a control part and a data part. The control part consists of $4k+2$ phases—each phase being the length of a round-trip between neighbors—after which a new set π of active links will be decided. This new set is active for the data part of the cycle. The whole process is repeated in the next cycle with updated queues. t counts the cycle number.

In our paper, we carefully take overheads into account *a priori* in the performance evaluation. For our algorithms it is clear: 1) how efficiently the overall wireless resources are utilized; 2) what is the tradeoff between scheduling performance and control overhead; and 3) how a system designer can choose his or her operating point on this tradeoff. These three aspects are elaborated below after a brief description of our results.

We are not the first to recognize the need for protocols with constant overheads. Some recent pieces of work [1]–[3] also propose constant-overhead algorithms. Their results and approaches are summarized and compared to ours in Section II. These existing constant-overhead algorithms can guarantee at most half of the capacity region (in the portion of resources dedicated to data transmission) and essentially involve using enhanced contention resolution as a way to approximate maximal matching in constant time. As opposed to these protocols, our algorithms can capture any desired fraction of the capacity region (in the data transmission part) and do not attempt enhanced contention resolution.

In this paper, we assume the "node exclusive" or "primary" interference model. In this model any node in the network can communicate with at most one other node at any time. This is an important model with a rich history of dedicated work, which we survey in Section II.

In our algorithms, bandwidth is assumed to be fixed and time is divided into scheduling cycles, with a new schedule generated by the algorithm in every cycle. As in other papers in this area, the length of a cycle is left to the protocol designer. Our algorithms partition each cycle into two parts: a scheduling (control signaling) part and a service (data-transmission) part. Fig. 1 depicts this partition. The partitioning of the scheduling cycle thus

Manuscript received January 13, 2008; revised September 06, 2008 and December 15, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Borst. First published May 26, 2009; current version published October 14, 2009. This work was supported by a Vodafone fellowship, NSF Grant CCF 06-34891 and Grant CNS 07-21286, and the DARPA CBMANET Project. An earlier version of this paper appeared in the *Proceedings of ACM SIGMETRICS*, June 2007.

L. X. Bui is with Airvana Inc., Chelmsford, MA 01824 USA (e-mail: locbui@ifp.uiuc.edu).

R. Srikantis is with the Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: rsrikant@illinois.edu).

S. Sanghavi is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: sanghavi@purdue.edu).

Digital Object Identifier 10.1109/TNET.2009.2013621

explicitly captures the wastage in control signaling: The fraction of resources wasted is the ratio of the length of the control part to the length of the overall cycle.

At the heart of our algorithm is a scheduler that generates a new schedule based on the queue lengths. For a parameter value k, the scheduler requires that the length of the scheduling part be $4k+2$ round-trip times, where one *round-trip time* is the amount of time required for a node to make a very basic two-way handshake with a neighboring node. Also, in any scheduling cycle and for any k, the algorithm requires that each node transmit at most three control signals, and the size of each control signal is independent of network size.

Congestion control and routing algorithms are built on top of this scheduler and control the number and identity of the packets injected into the network. The congestion controller for a multihop flow operates only based on the queue length at the source of the flow.

This algorithm with parameter k is guaranteed to achieve a fraction $(\frac{k}{k+2})$ of the capacity region *during the data part* of the cycle for any network and its associated flows. A larger value of k requires a longer absolute length of the control part and, in return, guarantees better performance in the data part, as explicitly detailed above. Thus, k captures the overhead-performance tradeoff and is a tunable knob that the protocol designer can use to optimize performance, with respect to other system considerations.

One such consideration that has a direct bearing on the appropriate choice of k is the length of the scheduling cycle. This is usually determined by the physical conditions (e.g., mobility, data rates, etc.) that the network is expected to operate under. If long cycles are determined to be feasible—where "long" is as compared to the round-trip time—it may make sense to choose a protocol with larger k. Conversely, short scheduling cycles may favor a small-k implementation. The choice of the parameter k may depend on network characteristics like mobility and arrival statistics; however, *it does not depend on network size*.

II. Background and Existing Work

Scheduling in the presence of interference constraints is a central problem in communication networks. In this summary, we will mainly concentrate on the work involving primary interference constraints, also known as the "node-exclusive" model in wireless networks. Primary interference constraints arise both in wireless networks and input-queued crossbar switches in Internet routers, and the results of the papers listed below are often of interest in both applications. In the following, "complexity" refers to the number of operations/amount of time that has to be spent every time a new schedule has to be found.

Hajek and Sasaki [4] introduced the primary interference model, which they studied in the wireless context and for fixed given arrivals. Tassiulas and Ephremides [5] were the first to consider stochastic arrivals in general interference models, of which primary interference is a special case. They characterized the maximum attainable capacity region and also presented a centralized algorithm guaranteed to achieve it. In the case of primary interference, this algorithm boils down to finding maximum weight matchings (with queue lengths being weights). This algorithm thus has $\Omega(n^2)$ complexity. McKeown *et al.* [6] also showed the same result for switches.

The need for speedy implementation and low overhead spurred the development of algorithms with lower complexity (but possibly higher delays). Tassiulas [7] studied randomized centralized algorithms that achieve the capacity region with $O(n)$ complexity. This algorithm samples a new candidate matching uniformly from the set of all matchings and switches schedules to this new sample if and only if it represents a larger weight. For the case of switches, this algorithm was derandomized by Giaccone *et al.* [8].

Weller and Hajek [9] showed that any algorithm that uses a maximal matching in every time slot can achieve half the capacity region. They showed this result for deterministically upper-constrained traffic. Dai and Prabhakar [10] showed the same performance holds for stochastic packet arrivals as well. Lin and Shroff [11] extended this result to the case of flow arrivals and departures. For tree topologies, Sarkar and Kar [12] presented a sequential maximal scheduling algorithm that achieves 2/3 of the capacity region.

Recently, distributed algorithms achieving the entire capacity region have been proposed (see, e.g., [13]–[16]). This guarantee, of course, refers to the scheduling efficiency with regards to data transmission since these papers do not account for resources used in overheads. Also, these protocols have overheads that grow with network size.

All of the above algorithms involve complexities that grow with network size. In some more recent work, scheduling algorithms with constant overheads have been proposed. Lin and Rasool [1] showed that close to 1/3 of the capacity region can be achieved with $O(1)$, i.e., constant, overhead. Gupta *et al.* [2] and Joo and Shroff [3] build on this result to achieve close to 1/2 of the capacity region with constant overheads. These algorithms attempt to generate (approximately) maximal matchings in every time slot using local contention algorithms that terminate in $O(1)$ time. Our approach in this paper is thus different from these papers, as we do not attempt to resolve contention. Furthermore, some recent developments have pushed in the directions of multihop traffic, e.g., [17], or more general interference constraints, e.g., [18]–[20].

III. System Model

We formally describe our system model, which is now standard in the literature. A wireless network is represented by a graph, $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of links. If a link (n, m) is in $\mathcal{L}$, then it is possible to transmit packets from node n to node m subject to the interference constraints that will be described shortly. We assume that time is slotted, denoted by t. Let c_{nm} be the fixed capacity of link $(n, m) \in \mathcal{L}$, which is the number of packets per time slot that can be transferred over that link. Also, let $l_{\max}$ denote the maximum number of links in any path or cycle in the network. Since $|\mathcal{N}|$ and $|\mathcal{L}|$ are finite, $l_{\max}$ is a finite integer.

We let $\mathcal{F}$ be the set of flows that share the network resources. For each flow $f \in \mathcal{F}$, let $b(f)$ and $e(f)$ denote the source and destination of f, respectively. Let us define D as the set of destination nodes in the network, i.e., $D = \{d \in \mathcal{N} : d = e(f)$ for some $f \in \mathcal{F}\}$. For each pair (n, d) where $d \in D$, $n \in \mathcal{N}$ and $n \neq d$, let $F(n, d)$ be the set of flows with source n and destination d, i.e., $F(n, d) = \{f \in \mathcal{F} : b(f) = n, e(f) = d\}$.

At each node, a queue is maintained for each destination.[1] We let $q_n^d[t]$ denote the length of the queue maintained at node n and storing packets that are destined to node d at the beginning of time slot t.

We assume that each flow has a utility function associated with it. The utility function of flow f, denoted by $U_f(\cdot)$, is defined as a function of the data rate sent by flow f and assumed to be strictly concave, nondecreasing, and twice differentiable.

We now describe the *node-exclusive spectrum sharing model*, also referred to as the *primary interference model*, assumed in this paper. Under this interference model, a node only communicates with at most one other node in any time slot. This means that the set of simultaneously active links is constrained to be a *matching* in $\mathcal{G}$. We let $\pi[t]$ be the binary vector of length $|\mathcal{L}|$ that denotes the set of active links at time t, with the convention that $\pi_{nm}[t] = 1$ if and only if link (n,m) is active for transmission at time t. We introduce the notation $\pi_{nm}^d[t]$ to distinguish packets destined to different destinations: At time slot t, $\pi_{nm}^d[t] = 1$ if link (n,m) serves packets destined to node d, and $\pi_{nm}^d[t] = 0$, otherwise. Hence

$$\pi_{nm}[t] = \sum_{d \in D} \pi_{nm}^d[t].$$

Let $\mathcal{M} = \{\pi^1, \pi^2, \ldots, \pi^{|\mathcal{M}|}\}$ be the set of all possible matching vectors π in $\mathcal{G}$. The *capacity (or stability) region* Λ of the network is defined as the set of flow rates that are supportable by the network. It is known (in [5] and [21]) that this region Λ is given by the set of vectors $\mathbf{x} = \{x_f\}_{f \in \mathcal{F}}$ for which there exists $\mu_{nm}^d \geq 0$, for all $(n,m) \in \mathcal{L}$ and $d \in D$, such that we have the following.

C1) For all $d \in D$, $n \in \mathcal{N}$, and $n \neq d$,

$$\sum_{f \in F(n,d)} x_f + \sum_{k:(k,n) \in \mathcal{L}} c_{kn} \mu_{kn}^d = \sum_{m:(n,m) \in \mathcal{L}} c_{nm} \mu_{nm}^d.$$

C2) $\left\{\sum_{d \in D} \mu_{nm}^d\right\}_{(n,m) \in \mathcal{L}} \in \mathrm{co}(\mathcal{M})$

where $\mathrm{co}(\mathcal{M})$ denotes the convex hull of the set $\mathcal{M}$. Notice that condition C1) is the flow conservation constraint at each node, and condition C2) captures the interference constraints.

IV. Algorithms and Results

Given the above model, a natural goal is to find the optimal network resource allocation, i.e., to find the mean flow rate vector $\mathbf{x}^\star$ satisfying

$$\mathbf{x}^\star \in \arg\max \sum_{f \in \mathcal{F}} U_f(x_f) \quad \text{s.t.} \quad \mathbf{x} \in \Lambda. \tag{1}$$

The strict concavity assumption of the utility functions implies that $\mathbf{x}^\star$ is unique. Now, let us describe an appropriate suboptimal solution to the optimization problem (1). Specifically, for any $\beta \in (0,1]$, we define the set $\beta\Lambda$ as follows:

$$\beta\Lambda \triangleq \left\{\mathbf{x} : \frac{1}{\beta}\mathbf{x} \in \Lambda\right\}.$$

[1]Our results can be easily adapted to the case when each flow has a fixed route in the network. In that case, each node maintains a queue for each flow going through it.

The set $\beta\Lambda$ can be viewed as the resulting set of flow rates when one takes only a "fraction" β of the capacity region. Then, the *β-suboptimal* solution $\mathbf{x}^\star(\beta)$ is defined as the optimal solution to the following optimization problem:

$$\mathbf{x}^\star(\beta) \in \arg\max \sum_{f \in \mathcal{F}} U_f(x_f) \quad \text{s.t.} \quad \mathbf{x} \in \beta\Lambda. \tag{2}$$

In this paper, we propose a joint congestion control, routing, and scheduling mechanism that achieves the β-suboptimal solution $\mathbf{x}^\star(\beta)$ for any given $0 < \beta < 1$ (see Theorem 2). Crucially, it does so while ensuring that *the time/overhead taken to generate a new schedule for each slot is constant*, independent of the size of the network but dependent on β. Specifically, the overhead of the algorithm is $O(\frac{1}{1-\beta})$, independent of the size $|\mathcal{N}|$ of the network.

We now present our main algorithms and results.

A. Routing and Scheduling

At time slot t, we define the α-differential backlog for d at link (n,m) as

$$w_{nm}^d[t] = c_{nm}\left(\left(q_n^d[t]\right)^\alpha - \left(q_m^d[t]\right)^\alpha\right)$$

where α is a system parameter which is positive. The weight of link (n,m) at time slot t is defined as

$$w_{nm}[t] = \max_{d: q_n^d[t] \geq c_{nm}} w_{nm}^d[t]. \tag{3}$$

Note that by letting $\alpha = 1$, we get back to the differential backlog weight that is usually used in the literature. Hence, the weight form that we consider here is more general. It is motivated by the recent work of Shah and Wischik [22] in the context of switches showing that if we use the *maximum-weight matching* (MWM) algorithm with weights being in a similar "α-form," then the delay performance will be improved as α decreases to zero.

Let $\mathbf{w}[t] = \{w_{nm}[t]\}_{(n,m) \in \mathcal{L}}$ be the vector of link weights and $\pi[t-1]$ be the old matching used in the previous time slot. Also, let $\pi^\star(\mathbf{w}[t])$ denote the optimal matching corresponding to the weight vector $\mathbf{w}[t]$, i.e.,

$$\mathbf{w}[t] \cdot \pi^\star(\mathbf{w}[t]) = \max_{\pi \in \mathcal{M}} \mathbf{w}[t] \cdot \pi.$$

It is well known that if the matching $\pi^\star(\mathbf{w}[t])$ is used at each step, then the resulting network throughput is optimal. However, computing $\pi^\star(\mathbf{w}[t])$ is computationally prohibitive. (The worst-case complexity is $O(|\mathcal{N}|^3)$ [23], and it requires a centralized authority.) Thus, in Section V, we present an algorithm that approximates the maximum weight matching $\pi^\star$ with a certain probability. This algorithm has low overhead and complexity and can be implemented in distributed manner. It is parameterized by a parameter k that provides a tradeoff between performance and overhead.

Let $\pi[t]$ be the matching determined by our algorithm. When the matching $\pi[t]$ is determined, if link (n,m) is scheduled (i.e., $\pi_{nm}[t] = 1$), then in the next time slot, c_{nm} packets of the commodity d that attains the maximum in (3) will be transmitted over link (n,m). The main result of the paper is the following theorem that establishes the properties of our scheduling algorithm.

Theorem 1: In any time slot t and for any values of $(\mathbf{w}[t], \pi[t-1])$, there exists a $\delta > 0$ such that the matching augmentation scheduling algorithm, operating using a fixed $k \geq 1$, generates a matching $\pi[t]$ with the following properties.

- The weight of new matching $\pi[t]$ is larger than or equal to the weight of old matching $\pi[t-1]$, i.e.,

$$\mathbf{w}[t] \cdot \pi[t] \geq \mathbf{w}[t] \cdot \pi[t-1]. \tag{4}$$

- If $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then with probability at least δ, $\pi[t]$ is an optimal weight matching corresponding to the weight vector $\mathbf{w}[t]$, i.e.

$$\mathbf{w}[t] \cdot \pi[t] = \mathbf{w}[t] \cdot \pi^{\star}(\mathbf{w}[t]).$$

- If $k < \lceil \frac{l_{\max}-1}{2} \rceil$, then with probability at least δ, $\pi[t]$ has weight at least $\frac{k}{k+2}$ fraction of the optimal weight

$$\mathbf{w}[t] \cdot \pi[t] \geq \frac{k}{k+2} \mathbf{w}[t] \cdot \pi^{\star}(\mathbf{w}[t]).$$

- The overhead in deriving a new schedule is $4k+2$ round-trip times.[2]

We will prove this theorem in Section VI.

B. Congestion Control

At the beginning of time slot t, each flow, say f, has access to the queue length of their first nodes, i.e., $q_{b(f)}^{e(f)}[t]$. Then, the instantaneous mean data rates $x_f[t]$ of flow f is set as follows:

$$x_f[t] = \min \left\{ U_f'^{-1} \left(\frac{\left(q_{b(f)}^{e(f)}[t] \right)^{\alpha}}{L} \right), M \right\}$$

where M, which is chosen to be large enough, represents the largest possible value of the flow rates. The positive constant L will be used to guarantee convergence of the achieved rates to the fair allocation. In particular, we are interested in the performance of the system for large L.

Each source has to convert the instantaneous mean data rate determined by the congestion controller into a packet injection rate. While the instantaneous rate could be a nonnegative real number, the number of packets injected into the network has to be a nonnegative integer that could be determined by some complicated mechanism that converts rates to packets. Instead of modeling this conversion, at each time slot t, the number of arrivals for flow f is assumed to be a Poisson random variable with mean x_f. The Poisson assumption is not important; any distribution with finite variance would work as well. This assumption can be easily relaxed in a number of ways without affecting our main conclusions.

C. Fair Resource Allocation

Theorem 1 naturally allows us to establish the following property of the joint congestion control, routing, and scheduling algorithm.

Theorem 2: For any given $0 < \beta < 1$, if the joint congestion control, routing, and scheduling algorithm described above is used with parameter $k \geq \frac{2\beta}{1-\beta}$, then the queues in the network are stable (i.e., the Markov chain of queue occupancies is positive recurrent), and the source rates satisfy

$$\sum_{f \in \mathcal{F}} U_f(\bar{x}_f) \geq \sum_{f \in \mathcal{F}} U_f(x_f^{\star}(\beta)) - \frac{B_1}{L}$$

where $\bar{x}_f = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[x_f[t]]$ and B_1, L are finite constants. Also, if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then the above expression also holds for $\beta = 1$.

Proof: Theorem 2 is a consequence of Theorem 1 and can be established along the lines of the proofs in [15] and [24]–[27]. The reader is also referred to the related work in [28]. For details, see Appendix A. ■

V. The Matching Augmentation Scheduling Algorithm

In this section, we present our algorithm for determining the new schedule $\pi[t]$ from $(q_t, \pi[t-1])$. To do so, we will need a few simple definitions. In the following, we will abuse notation by letting π denote the matching as well as the associated binary vector of length $|\mathcal{L}|$: $\pi_{nm} = 1$ if and only if link (n,m) is in the matching π.

Recall that, in π, no two adjacent links can be active. An *augmentation* A of a matching π is a path or cycle in which every alternate link is in π, with the property that if all links in $A \cap \pi$ are removed from π and all links in $A \setminus \pi$ are added[3] to π, then the resulting set of links will remain a matching in G. This process of changing π using A is called *augmenting* π with A, and the resulting augmented matching is denoted by $\pi \oplus A$. The process is illustrated in Fig. 2.

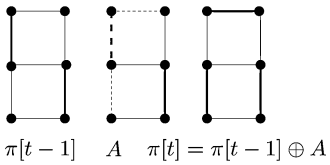

Fig. 2. An illustration of the augmenting process.

The bold lines in the left-most figure indicate links in $\pi[t-1]$, the existing matching. The dotted lines in the central figure indicate the links in augmentation A. The bold-dotted lines are in $A \cap \pi[t-1]$, and the thin-dotted lines are in $A \setminus \pi[t-1]$. The bold lines in the last figure are the links in $\pi[t] = \pi[t-1] \oplus A$, the new matching obtained by augmenting $\pi[t-1]$ with A.

The *size* $|A|$ of augmentation A is the number of links in A. Two augmentations, A_1 and A_2, are *disjoint* if they have no common links. This implies that π can be simultaneously augmented by A_1 and A_2 and still be a valid matching. $\mathcal{A}$ is a *set of disjoint augmentations* if every pair in $\mathcal{A}$ is disjoint. Clearly, if $\mathcal{A}$ is a set of disjoint augmentations, then $\pi \oplus \mathcal{A}$ will be a matching in G.[4] Finally, for any time t, the *gain* of an augmentation A to $\pi[t-1]$ is defined as

$$\text{gain}_t(A) := \sum_{(n,m) \in A \setminus \pi[t-1]} w_{nm}[t] - \sum_{(n,m) \in A \cap \pi[t-1]} w_{nm}[t]. \tag{5}$$

Hence, $\text{gain}_t(A)$ represents the change in the weight of $\pi[t-1]$ if it is augmented by A. Similarly, the gain of a set $\mathcal{A}$ of disjoint

[2]One *round-trip time* is the amount of time it takes for a pair of neighbors in the network to execute a basic two-way handshake.

[3]Here, $A \setminus \pi$ denotes the set of links in A but not in π.

[4]$\pi \oplus \mathcal{A}$ is π augmented by every $A \in \mathcal{A}$. The fact that the augmentations are disjoint means that this can be done in any order.

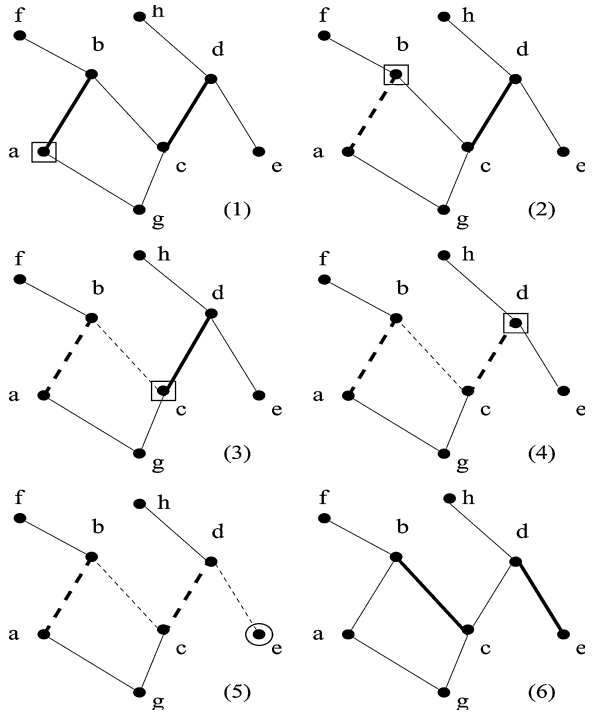

Fig. 3. Augmentation building in our algorithm (Example 1). The □ indicates active nodes. Dashed lines are links are in A. Bold lines in the first 5 graphs are links in $\pi[t-1]$, and in the last graph are links in $\pi[t-1] \oplus A$. ◯ indicates terminus.

augmentations is the sum of the gains of each augmentation in that set. Here, the weights are as defined in (3).

Our algorithm, with parameter value k, can be summarized as follows: *Build a random set of disjoint augmentations to* $\pi[t-1]$*, each of size at most* $2k+1$*, and switch the ones with positive gain*.[5]

In our algorithm, augmentations are built in a random distributed fashion. We now present a simplified example to aid in the visualization of how our algorithm makes one augmentation.

Example 1: Augmentation Building in the Absence of Contention: Consider Fig. 3, which depicts our algorithm operating in phases on a simple graph. The bold links in the top left graph of Fig. 3 are the links of $\pi[t-1]$.

In our algorithm, an augmentation begins at a *seed* and ends at a *terminus*. This seed is *active* in phase 1. When a node is active, it tries to elongate its augmentation A by adding links as follows.

1) If the node has a link in $\pi[t-1]$ that is not already in A, it adds that link to A.
2) Else, it adds a random new link to A.

Every time a link is added, for the next phase the currently active node becomes inactive, and the other endpoint of the new link becomes active instead.

As seen in the first figure, node a is the only seed and is active in phase 1. It adds link (a,b) to its (currently empty) augmentation in phase 1, since $(a,b) \in \pi[t-1]$, and it is not currently in a's augmentation. For phase 2, a is inactive and b is active.

[5]The idea of using fixed-length augmentations to obtain approximations to maximum-weight matching has been used previously (see, e.g., Pettie and Sanders [29]) in a different, pure graph-theoretic context to find an approximation to maximum-weight matching with linear complexity.

Node b can choose to add any one of the links (b,c) or (b,f). Say it chooses (b,c). Therefore, for phase 3, node c is active and b is inactive.

Node c adds the link (c,d) because it is a new link in $\pi[t-1]$. Then, in phase 4, d becomes active. d picks randomly from links (d,e) and (d,h). Say it picks (d,e).

Node e would have become active as a result, but it has no further links to add to the augmentation. It instead becomes the *terminus* of the augmentation.

The net weight is communicated forward in each hop. Terminus e then evaluates the gain

$$\text{gain}_t(A) = w_{bc}[t] + w_{de}[t] - w_{ab}[t] - w_{cd}[t]$$

using the link weight information that has been passed on along a,b,c,d,e during the building of the augmentation. In our example, it finds that $\text{gain}_t(A) > 0$ and decides to switch. This decision is then passed on back along the links (e,d), (d,c), (c,b), and (b,a) over the next 4 phases. Then, the links in A are switched to obtain the final graph in Fig. 3, where bold links are the ones in $\pi[t-1] \oplus A$. ■

The above example illustrates the simple basic ideas underlying our algorithm, namely 1) randomly seed and grow disjoint augmentations, and 2) switch all the augmentations that have positive gain.

The above example illustrated the augmentation building procedure in an idealized network where it was the only augmentation. In an actual wireless network, however, augmentations are seeded at random. This means that there will be multiple augmentations, which will have to contend for access to links while ensuring that they remain consistent (i.e., valid augmentations) and disjoint.

We now succinctly describe the algorithm that constructs $\pi[t]$ from $(\mathbf{w}[t], \pi[t-1])$. A more realistic, and more detailed, example and a brief discussion follow the description.

In our algorithm, each augmentation builds up in phases starting from a seed and ending in a terminus. For any phase and node v, let $\text{aug}(v)$ denote the augmentation (if any) that v is a part of in that phase. Also, the term "new link" for a node v refers to any link (u,v) that is not already in $\text{aug}(v)$. For any active v except the seed, one of its links will be in $\text{aug}(v)$, and all the others will be new. Similarly, a "new neighbor" for v is any node that shares with v a link that is new for v.

A. Algorithm Description

1) **Initialization:** Before phase 1,
   a) each node randomly decides to be a *seed* with probability p;
   b) each seed chooses an *intended size* for its augmentation, uniformly from the set $\{1,\ldots,k\}$.

   The seeds are the *active* nodes in phase 1.
2) **Iteration:** Every seed s that has some link $(s,r) \in \pi[t-1]$ adds (s,r) to $\text{aug}(s)$ and sends a REQ to the corresponding r in phase 1. Every other seed sends a REQ along a random link. In each phase from 2 to $2k+1$, each active node v tries to extend its $\text{aug}(v)$ as follows.
   a) If $\text{aug}(v)$ needs a new link $\pi[t-1]$, and if one such link $(v,u) \in \pi[t-1]$ exists, then (v,u) is added to $\text{aug}(v)$. Also, v sends a REQ to u along (v,u). If no such link exists, v becomes *terminus*.

b) If $\mathrm{aug}(v)$ needs a new link outside $\pi[t-1]$, but $\mathrm{size}(\mathrm{aug}(v))$ is already as intended, v sends no REQ and becomes terminus. Otherwise, v sends REQ to a random, uniformly chosen, new neighbor. If no new neighbor exists, v becomes terminus.

Contention: The REQs above may face contentions. In any phase, if active v sends REQ to w,

a) another active u also sends REQ to w in that phase, and a *collision* occurs. w does not ACK, and v becomes terminus.
b) If w is a *used* node, i.e., it is already part of an augmentation, then w does not ACK, and v becomes terminus.
c) If w is unused and there is no collision, then w sends ACK to v. The link (v,w) is added to $\mathrm{aug}(v)$. w will become active in the next phase, and v will become inactive.

Every node that becomes terminus is inactive in subsequent phases.

3) **Termination:** After $2k+1$ phases, every terminus w checks the following three conditions:
   a) if it is adjacent to its seed v,
   b) if $\mathrm{aug}(w)$ began and ended with links in $\pi[t-1]$, and
   c) if $\mathrm{aug}(w)$ has not reached its intended size.

   If all of above are true, link (w,v) is added to $\mathrm{aug}(w)$. Also, in either case, every terminus w evaluates $\mathrm{gain}_t(\mathrm{aug}(w))$ and makes the decision of switching if and only if $\mathrm{gain}_t(\mathrm{aug}(w)) > 0$.
4) **Back-propagation and Switching:** Switching decision is relayed back in phases $2k+2$ to $4k+2$ from terminus to seed, along the path of each augmentation. These communications will be noninterfering. After phase $4k+2$, all nodes in augmentation implement decision.

B. Discussion

Note that, in our algorithm, there is a small but crucial difference between links in $\pi[t-1]$ and links outside $\pi[t-1]$ with regards to the timing of link addition to an augmentation. Specifically, a link in $\pi[t-1]$ is added *before* a REQ is sent (and irrespective of whether an ACK is received), while a link outside $\pi[t-1]$ is added only after a REQ is sent *and* an ACK is received. This difference in timing ensures that augmentations are consistent, i.e., whenever a link $e \notin \pi[t-1]$ is added to A, all the links in $\pi[t-1]$ adjacent to e are *also* added to A.

It is easy to see that our algorithm will ensure disjointness of augmentations. Consider the addition of a link $(v,u) \notin \pi[t-1]$ to $\mathrm{aug}(v)$ by an active node v in some phase. This will only happen if v sends a REQ to u in that phase and u responds with an ACK. Now, if u is already a member of an augmentation by that phase (including the case of it already being a member of $\mathrm{aug}(v)$ itself), then u will not respond with an ACK. Also, if any other augmentation tries to add some other link $(w,u) \notin \pi[t-1]$ at the same phase, then neither augmentation will be successful. Thus, no two adjacent links outside $\pi[t-1]$ will be part of augmentations, and all augmentations are consistent and disjoint. Note, of course, that any link (z,u) in $\pi[t-1]$ *can* be added to another augmentation. This, however, does not hurt the disjointedness.

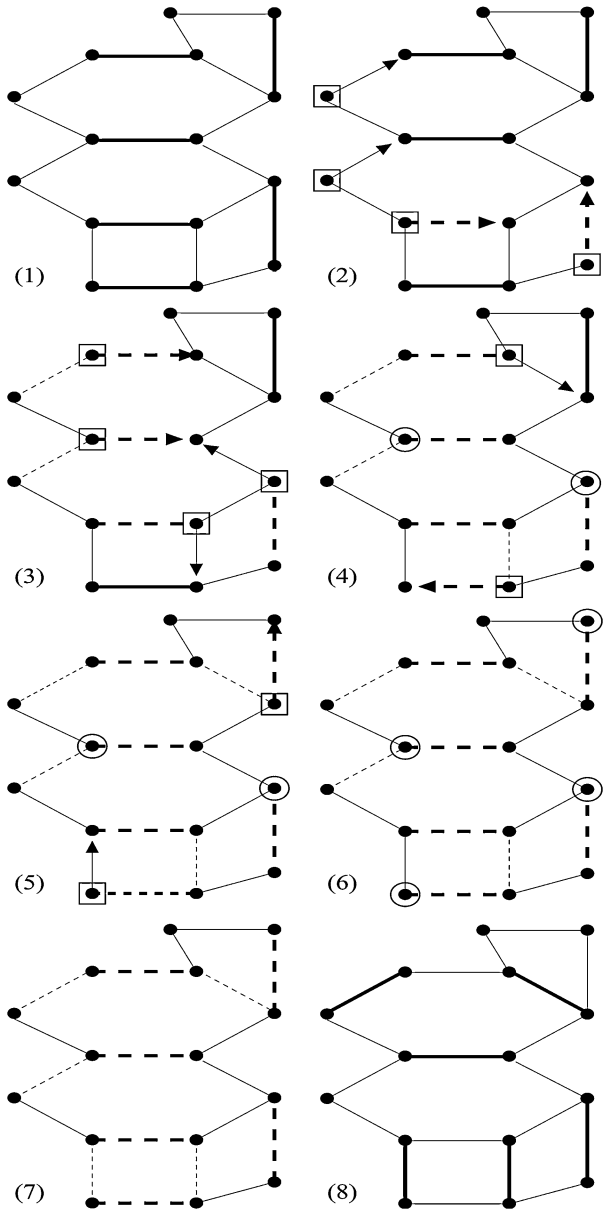

Fig. 4. The working of our algorithm in phases (Example 2). Bold lines in the first 7 graphs are links in $\pi[t-1]$, and in the last graph are links in $\pi[t]$. Dashed lines are links in augmentations. □ denotes active nodes, and ◯ denotes terminus nodes. Arrows depict the REQ signals in each phase.

We now illustrate how our algorithm works by means of a slightly detailed example.

Example 2: Illustration of the Algorithm: Consider the network shown in Fig. 4. The first graph shows the schedule $\pi[t-1]$ of the previous time slot. The second graph shows the nodes active in phase 1, which are the seeds of the network. Assume $k=2$ and that the intended size chosen by every seed is also 2. Note that the active nodes that have a link in $\pi[t-1]$ incident on them immediately include it in their augmentations, and also send the REQ along that link. Other active nodes do not include any links in their augmentations, and send REQs at random.

Since all REQs went to different targets, there were no collisions, and so there are four new nodes active in phase 2. Again, the newly active nodes that have a new link in $\pi[t-1]$ incident on

them immediately add it to their augmentation and send REQs along them. Other newly active nodes send REQs at random.

We see that two of the REQs sent in phase 2 will collide. This means that the corresponding transmitters of the collided REQs will become terminus of their augmentations, which then stop growing. The other two augmentations continue growing.

In phase 4, the active node at the bottom sends REQ to a used node (its own seed in this case). Hence, it does not receive an ACK and becomes the terminus. The other active node in phase 4 adds its link without problems. However, the addition of another link to this augmentation would make it exceed its intended size. Thus, after the addition, the new node becomes terminus and does not further add links.

After the last phase 5, note that one of the augmentations (the bottom one) satisfies the conditions in step 4 of the algorithm: Namely, it began and ended with links in $\pi[t-1]$, its seed and terminus are joined by a link not in $\pi[t-1]$, and the addition of this link does not violate the intended size. Thus, this link is added to get the final set of disjoint augmentations after phase $5(=2k+1)$.

These augmentations are switched depending on their gains. In our example, two of the four augmentations are switched and the other two are not. The switching decisions are relayed back in phases 6 to 10, after which the switching happens. The final graph depicts the resulting $\pi[t]$. ■

The example illustrates the fact that without the last link addition in the "termination" part of the above algorithm, we would not be able to build augmentations that are (small) cycles.

Note that the information relayed on at each phase does not grow with k or the network size. Indeed, all that is needed is the net gain of the augmentation up until the current phase and the identity of the first node (so that the termination part of the algorithm can be implemented).

The probability p in the above algorithm is a parameter that can be chosen by the system designer. If p is too high, there will be too many seeds in the network, which will result in too much contention and not enough augmentations. If p is too small, there will not be enough seeds to ensure a good enough set of augmentations.

VI. Properties of the Scheduling Algorithm

In this section, we present the Proof of Theorem 1. Recall that $\mathbf{w}[t]$ is the vector of link weights at time t, and $\pi^*(\mathbf{w}[t])$ is the optimal matching for $\mathbf{w}[t]$. For Lemmas 1–3, we drop the index t with the understanding that $\mathbf{w}=\mathbf{w}[t]$ and $\pi=\pi[t-1]$.

Lemma 1: Given any vector $\mathbf{w}$ of queue lengths and existing matching π, there exists a set $\mathcal{A}^*$ of disjoint augmentations of π such that:

- if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then
$$(\pi \oplus \mathcal{A}^*)\cdot \mathbf{w} = \pi^\star(\mathbf{w})\cdot \mathbf{w};$$
- if $k < \lceil \frac{l_{\max}-1}{2} \rceil$, then
$$(\pi \oplus \mathcal{A}^*)\cdot \mathbf{w} \geq \left(\frac{k}{k+2}\right)\pi^\star(\mathbf{w})\cdot \mathbf{w}$$

and every augmentation in $A \in \mathcal{A}^*$ has $\text{size}(A) \leq k$.

Lemma 1 says there exists a "good set" $\mathcal{A}^*$ of augmentations, where each augmentation is not too large, and that augmenting π using the set represents a certain amount of gain. We build up toward the proof of Lemma 1 by designing a candidate set $\mathcal{A}^*$ having augmentations of size at most k. We will then prove the gain it represents is as claimed by Lemma 1.

The *symmetric difference* $S := \pi \triangle \pi^\star(\mathbf{w})$ of matchings π and $\pi^\star(\mathbf{w})$ is the set of links that are in exactly one of π or $\pi^\star(\mathbf{w})$. Links that are in both or in neither are excluded from S. Consider the graph $G' := (V, S)$ containing only the links in S. Since each π is a matching, a vertex in G' has degree at most 2 and each connected component of S is either an alternating path or even-length cycle. Also, each component is an augmentation of π, so we can define the size of a component in the same way as we defined the size of an augmentation of π.

For any component C of S, let $C_1 := C \cap \pi^\star(\mathbf{w})$ denote the links of C in the optimal matching and $C_2 := C \cap \pi$ be the links in the current matching. Note that C_1 and C_2 are also matchings in G, and the terms $C_1 \cdot \mathbf{w}$ and $C_2 \cdot \mathbf{w}$ are as defined before for matchings. Also, note that $\text{size}(C) = |C_1|$.

We build the set $\mathcal{A}^*$ from S by finding a suitable set $\mathcal{A}_C$ in each component C of S. The following two lemmas ensure this can be done.

Lemma 2: If any component C of S is a path (i.e., not a cycle), then there exists a set $\mathcal{A}_C$ of disjoint augmentations such that:

1) every $A \in \mathcal{A}_C$ is contained in C, and has $\text{size}(A) \leq k$;
2) if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then
$$\text{gain}(\mathcal{A}_C) = C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w};$$
3) if $k < \lceil \frac{l_{\max}-1}{2} \rceil$, then
$$\text{gain}(\mathcal{A}_C) \geq \frac{k}{k+1} C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w}.$$

Proof: If C is small, with $\text{size}(C) \leq 2k+1$, then let $\mathcal{A}_C = \{C\}$ be the set containing only C. This obviously satisfies all of the above conditions since $\text{gain}(C) = C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w}$. Specially, if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then $\text{size}(C) \leq k$ for every path C of S.

Now, consider a path C with $\text{size}(C) \geq 2k+2$. Starting from any endpoint of the path, number all links in C_1. Let $e_1, e_2, \ldots$ be the links in C_1. For the first few e_i, $1 \leq i \leq k+1$, build a set $\mathcal{A}_i$ of disjoint augmentations by deleting every $(k+1)$th link in C_1, starting with e_i; i.e., link e_m is deleted if and only if $m-i = 0$ or $m-i$ is divisible by $k+1$. Fig. 5 shows this process for a simple example. After the deletions, each remaining fragment of C will be an augmentation of π and will have size at most $2k+1$. These fragments together make the set $\mathcal{A}_i$ of disjoint augmentations.

Consider now the sets $\mathcal{A}_1, \ldots, \mathcal{A}_{k+1}$, made from the links $e_1, \ldots, e_{k+1}$, respectively. It is clear that each link in C_2 will be a member of all $k+1$ of these sets, and each link in C_1 will be a member of k of the sets. Recall from (5) that the gain of any set $\mathcal{A}$ is the weight of all its links outside π minus the weight of all its links in π. Thus

$$\sum_{i=1}^{k+1} \text{gain}(\mathcal{A}_i) = kC_1 \cdot \mathbf{w} - (k+1)C_2 \cdot \mathbf{w}$$

which means that there exists at least one j such that

$$\text{gain}(\mathcal{A}_j) \geq \frac{k}{k+1} C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w}.$$

Setting $\mathcal{A}_C = \mathcal{A}_j$ proves the lemma. ■

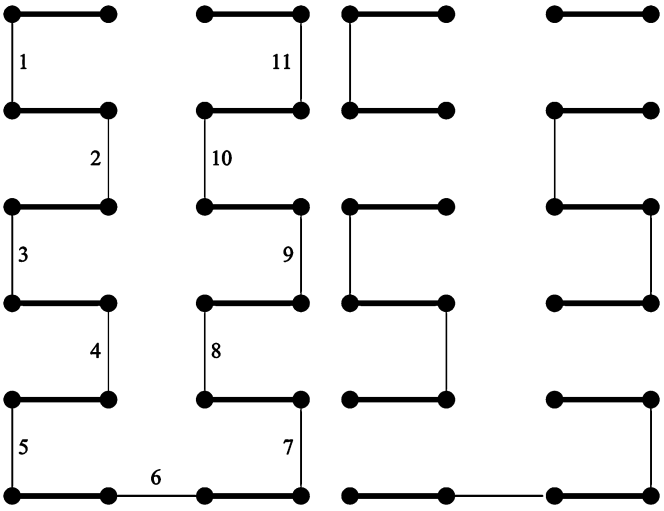

Fig. 5. The path on the left is a path component C of S, with nonbold links denoting $C_1 = C \cap \pi^\star(\mathbf{w})$ and bold ones for $C_2 = C \cap \pi$. Links in C_1 are numbered, and $k = 2$. C is cut up starting at link number 2, going counterclockwise, to obtain the set $\mathcal{A}_2$ on the right. Note that $\mathcal{A}_2$ contains disjoint augmentations of size at most $k = 2$.

Lemma 2 shows the existence of a good set of disjoint augmentations in path components of S. We now use this to prove a slightly weaker result for cycle components of S.

Lemma 3: If component C of S is a cycle, then there exists a set $\mathcal{A}_C$ of disjoint augmentations such that:

1) every $A \in \mathcal{A}_C$ is contained in C and has $\text{size}(A) \leq 2k+1$;
2) if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then

$$\text{gain}(\mathcal{A}_C) = C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w};$$

3) if $k < \lceil \frac{l_{\max}-1}{2} \rceil$, then

$$\text{gain}(\mathcal{A}_C) \geq \frac{k}{k+2} C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w}.$$

Note that the ratio is $\frac{k}{k+2}$ instead of $\frac{k}{k+1}$ as it was for paths.

Proof: If $\text{size}(C) \leq 2k+1$, set $\mathcal{A}_C = \{C\}$, and we are done. Specially, if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then $\text{size}(C) \leq k$ for every path C of S.

Now, consider cycles C in S with $\text{size}(C) \geq 2k+2$. Let $e \in C_1$ be the link with the smallest weight: $w_e[t] \leq w_{e'}[t]$ for all $e' \in C_1$. Consider path $\widehat{C} = C - e$, and define $\widehat{C}_1 := \widehat{C} \cap \pi^\star(\mathbf{w})$ and $\widehat{C}_2 := \widehat{C} \cap \pi[t]$ as before. Obviously, $C_2 = \widehat{C}_2$. Also, since e was chosen to be the link in C_1 with smallest weight and $|C_1| = \frac{\text{size}(C)}{2} \geq k+1$

$$\widehat{C}_1 \cdot \mathbf{w} \geq \frac{|C_1|}{|C_1|+1} C_1 \cdot \mathbf{w} \geq \frac{k+1}{k+2} C_1 \cdot \mathbf{w}.$$

Also, by Lemma 2, there exists a set $\mathcal{A}_{\widehat{C}}$ of disjoint augmentations of size at most $2k+1$ in $\widehat{C}$ such that

$$\begin{aligned} \text{gain}(\mathcal{A}_{\widehat{C}}) &\geq \frac{k}{k+1} \widehat{C}_1 \cdot \mathbf{w} - \widehat{C}_2 \cdot \mathbf{w} \\ &\geq \frac{k}{k+1} \left(\frac{k+1}{k+2} C_1 \cdot \mathbf{w} \right) - C_2 \cdot \mathbf{w} \\ &= \frac{k}{k+2} C_1 \cdot \mathbf{w} - C_2 \cdot \mathbf{w}. \end{aligned}$$

Setting $\mathcal{A}_C = \mathcal{A}_{\widehat{C}}$ proves the lemma. ∎

We are now ready to build the set $\mathcal{A}^*$: For each component C of S, add to $\mathcal{A}^*$ the augmentations in the corresponding $\mathcal{A}_C$—where $\mathcal{A}_C$ is as specified by Lemmas 2 and 3 for paths and cycles, respectively. We are now ready to prove Lemma 1.

Proof of Lemma 1: Let $\mathcal{A}^*$ be as constructed above. Its gain will be the sum of the gains of each of the $\mathcal{A}_C$'s of which it is composed. Therefore, if $k < \lceil \frac{l_{\max}-1}{2} \rceil$, then we have

$$\begin{aligned} \text{gain}(\mathcal{A}^*) &= \sum_C \text{gain}(\mathcal{A}_C) \\ &\geq \frac{k}{k+2} \left(\sum_C C_1 \cdot \mathbf{w} \right) - \left(\sum_C C_2 \cdot \mathbf{w} \right) \\ &= \frac{k}{k+2} \{ \pi^\star(\mathbf{w}) \cdot \mathbf{w} - (\pi^\star(\mathbf{w}) \cap \pi) \cdot \mathbf{w} \} \\ &\quad - \{ \pi \cdot \mathbf{w} - (\pi^\star(\mathbf{w}) \cap \pi) \cdot \mathbf{w} \} \\ &\geq \frac{k}{k+2} \pi^\star(\mathbf{w}) \cdot \mathbf{w} - \pi \cdot \mathbf{w} \end{aligned}$$

where $\pi^\star(\mathbf{w}) \cap \pi$ is the matching consisting of links that are in both $\pi^\star(\mathbf{w})$ and π.

Similarly, if $k \geq \lceil \frac{l_{\max}-1}{2} \rceil$, then

$$\text{gain}(\mathcal{A}^*) = \pi^\star(\mathbf{w}) \cdot \mathbf{w} - \pi \cdot \mathbf{w}.$$

The lemma's proof follows from the fact that

$$\text{gain}(\mathcal{A}^*) = (\pi \oplus \mathcal{A}^*) \cdot \mathbf{w} - \pi \cdot \mathbf{w}$$

by definition. ∎

Now that we have shown the existence of a suitable set $\mathcal{A}^*$, all we need to do to prove Theorem 1 is to uniformly lower bound the probability of the algorithm generating $\mathcal{A}^*$.

Proof of Theorem 1: Recall that in our algorithm a disjoint set of augmentations $\mathcal{A}$ is created, and the ones with positive gain are switched to obtain $\pi[t]$ from $\pi[t-1]$. Thus

$$\pi[t] \cdot \mathbf{w}[t] \geq (\pi[t-1] \oplus \mathcal{A}) \cdot \mathbf{w}[t].$$

Therefore, it suffices to lower bound

$$P[\mathcal{A} = \mathcal{A}^* | \mathbf{w}[t], \pi[t-1]]$$

by a quantity that is independent of $(\mathbf{w}[t], \pi[t-1])$ but can depend on graph structure and k. Note, of course, that $\mathcal{A}^*$ is not independent of $(\mathbf{w}[t], \pi[t-1])$.

We will now provide a very naive lower bound to the above probability. Let $l = |\mathcal{A}^*|$ be the number of disjoint augmentations in $\mathcal{A}^*$. Choose one node in each of these augmentations as follows. If the augmentation is a path, choose one of its endpoints, and if it is a cycle, choose any node. Let $\mathcal{E}$ be the event that *all* of the following are true:

- the algorithm generates $\mathcal{A} = \mathcal{A}^*$;
- the only seeds active in phase 1 are the nodes chosen above, and
- each augmentation's intended length (chosen at its seed) is equal to the actual length of that augmentation in $\mathcal{A}^*$ (i.e., no augmentation is a result of a cutoff due to contention).

Clearly $P[\mathcal{A} = \mathcal{A}^* | \mathbf{w}[t], \pi[t-1]] \geq P[\mathcal{E} | \mathbf{w}[t], \pi[t-1]]$

We now lower-bound the right-hand side. The probability that the nodes turn active or remain inactive as specified by $\mathcal{E}$ is $p^l(1-p)^{n-l}$. Furthermore, the probability that the intended lengths are exactly as chosen is k^{-l} since they are chosen uniformly from $1, \ldots, k$. Finally, we need each of the random link-

addition choices (step 2 of the algorithm) of each of the above l augmentations to exactly parallel the corresponding ones in $\mathcal{A}^*$. There are a total of at most n such choices to be made, and the probability of each being the correct one is at least $\frac{1}{\Delta}$, where Δ is the maximum degree of the graph. Thus, the probability of correct link choices is at least Δ^{-n}.

Putting everything together, we have that

$$\begin{aligned} &P[\mathcal{E}|\mathbf{w}[t], \pi[t-1]] \\ &\quad \geq p^l(1-p)^{n-l}k^{-l}\Delta^{-n} \\ &\quad \geq \min\left\{1, \left(\frac{p}{1-p}\right)^n\right\}\left(\frac{1-p}{k\Delta}\right)^n. \end{aligned}$$

The right-hand side of the last inequality is now independent of $(\mathbf{w}[t], \pi[t-1])$, providing us with the required uniform lower bound. Setting $\delta = \min\{1, (\frac{p}{1-p})^n\}(\frac{1-p}{k\Delta})^n$ completes the proof of the theorem. ■

VII. Simulations

In this section, we investigate the performance of our class of algorithms via simulations. The primary purpose of this investigation is to look at the delay performance of our algorithms. In these simulations, we compare our algorithm to the following algorithms.

- Maximal Matching (MM): At every time step, a maximal matching[6] is generated, and links in this matching are made active while others are made inactive.
- (Delayed) Max-Weight Matching-t: Every t time steps, the max-weight matching, with weights being the queue lengths, is chosen as the schedule. This schedule is then used for the subsequent t steps.

Note that both of the above algorithms, at least as stated here and as simulated by us, are centralized.

It is natural to compare our algorithms to the existing constant-overhead algorithms [1]–[3]. Each of these algorithms essentially tries to emulate maximal matching via enhanced contention resolution. Therefore, for our simulations, we compare our algorithm directly with the maximal matching algorithm itself, with the implicit understanding that the performance of MM would be better than that of the existing constant-time protocols. All of these algorithms, including the centralized MM algorithm, can guarantee at best half of the capacity region, and hence, from the capacity viewpoint, they are *not* comparable; our algorithms can capture any fraction $k/(k+2)$ of the capacity region for any $k \geq 1$. We emphasize that, in this section, we compare only the delay performance.

Our algorithm has constant overhead in each scheduling cycle, and, hence, the long-term overhead is also constant. If, however, one is only interested in the *weaker* requirement that the long-run overhead be low, one could use a potentially more complex algorithm to find a schedule, and then use the schedule for a long time to *amortize* the complexity. Such an approach may be feasible in high-speed router switches, but it presents significant implementation and coordination challenges for wireless networks. We discuss these in Section VIII; in the present section, we only consider the delay performance of one such high-complexity algorithm, MWM, when it is delayed in this fashion. Max-weight matching has been observed, both in theory and practice, to have very good delay performance.

[6]Recall that a *maximal* matching is any matching to which no link can be added without removing an existing link.

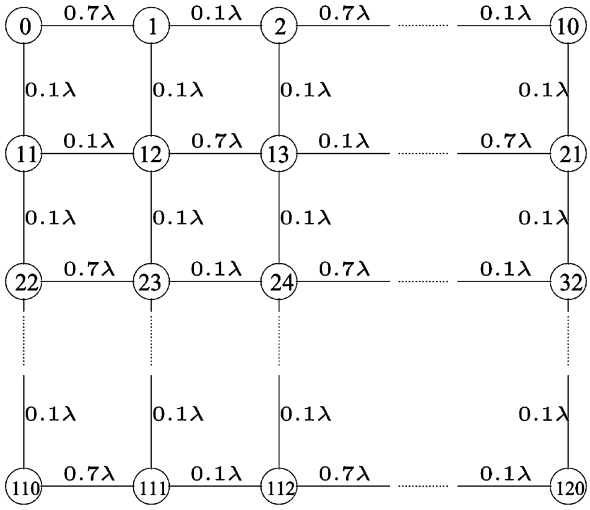

Fig. 6. Network Topology.

We ran our simulations on a simple grid network, shown in Fig. 6. The network is an 11×11 grid with 121 nodes (represented by circles) and 220 links (represented by lines). Each link has the unit capacity—i.e., it can transmit one unit of data in one time slot—when active. The data arrival model is as follows: On each link, in each time slot, one unit of data arrives with probability equal to the load on the link; otherwise, no data arrives.

Link loads in our example are parameterized by λ and are either 0.1λ or 0.7λ, as marked in the figure. It is clear, from the node-exclusive interference model, that the capacity region of this network corresponds to $\lambda < 1$. The numbers 0.1 and 0.7 thus specify the *direction* in which we are investigating the capacity region, while λ specifies the *extent* of the capacity region that is captured along that direction.

Fig. 7 compares the performance of our algorithm with that of MM. We plot the average maximum queue backlog of all links in the networks, in terms of λ, after 48 000 time slots. Our algorithm has been run for the cases $\{k = 2, p = 0.2\}$, $\{k = 3, p = 0.2\}$, and $\{k = 3, p = 0.1\}$.

Notice first of all that *even small-k implementations of our algorithm can stabilize higher loads than MM in reasonable networks like the one in the simulation*. In particular, while both MM and our algorithm for $k = 2$ both guarantee only half the capacity region in the worst case, our $k = 2$ algorithm achieves close to 100% throughput in the simulation, while MM achieves only close to 85%. Also, the fact that $k = 2$ is already so close to 100% means that $k = 3$ performs similar to $k = 2$ in terms of achievable capacity. It, however, does have a slightly better delay performance.

Also observe that, for small loads, our algorithm has higher queue sizes as compared to MM. This is because MM ensures that every link will be scheduled, or there will be an interfering link scheduled instead. When loads are low, the number of interfering links is small because the corresponding queues are empty for large portions of the time. This leads to good utilization. Our algorithms are randomized and do not ensure maximal usage. However, our simulations indicate that the queue lengths of our algorithms are not unreasonable.

The parameter p of a node becoming a seed is a free parameter that can be set by the designer. Our simulations seem to indicate

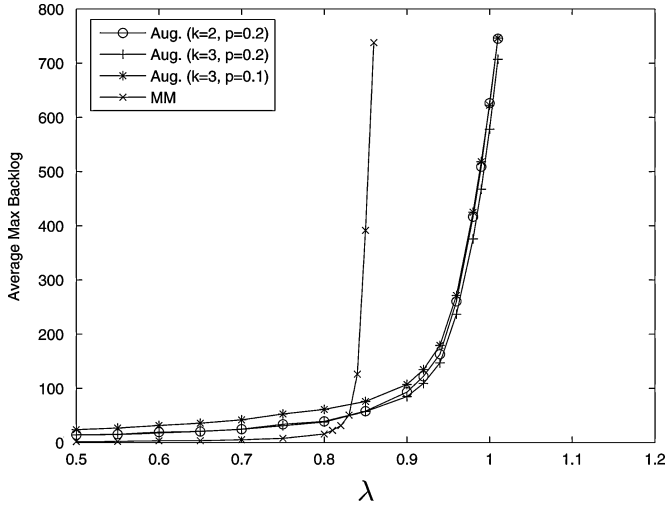

Fig. 7. Performance comparison to MM.

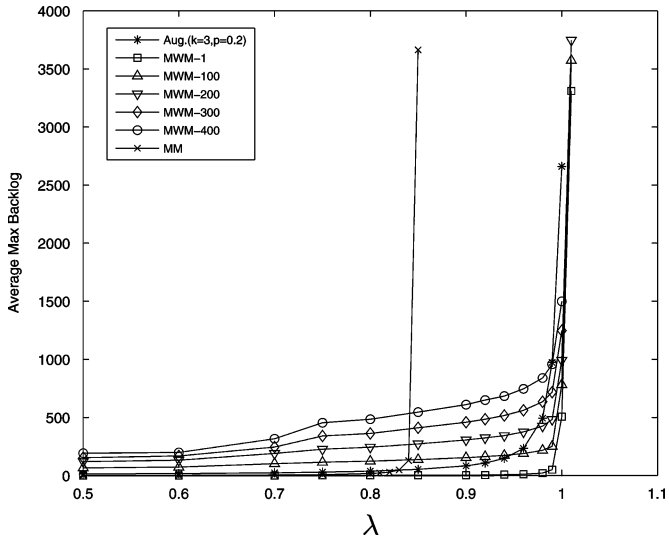

Fig. 8. Performance comparison to delayed versions of MWM.

that for reasonable values of p, the exact value does not seem to have an overwhelming effect on performance. This seems to suggest that performance may not be too sensitive to the *exact* choice of the parameter.

In Fig. 8, we compare our algorithm to delayed versions of the MWM algorithm. For example, MWM-100 corresponds to the algorithm where a max-weight matching is found every 100 time slots, and the resulting schedule is used for the subsequent 100 slots. MWM-1 corresponds to finding the max-weight matching in every time slot; this is a very high-complexity algorithm and, as can be seen, has very good delay performance even close to capacity. The simulation running time in this case is a million slots. As expected, average delays increase when the delay t of MWM-t increases; conversely, the amortized complexity decreases as t increases.

Finding the max-weight matching can take as long as $O(n^3)$ time in the worst case. Note that $n = 121$ in our simulations, which makes n^3 quite a large number. Hence, amortizing to get constant long-run overhead means that schedules would be updated after very long intervals. In practice, finding the MWM may be much faster; in our simulations, we compare our algorithm with parameters arbitrarily fixed at $k = 3$ and $p = 0.2$ to MWM-t, with delay values $t = 1, 100, 200, 300,$ and 400 (values much smaller than n^3).

Each MWM-t is a capacity-achieving algorithm, and, hence, has lower delays when the load is very close to capacity. However, for most moderate values of loads, our algorithm has lower delays than all except the undelayed MWM-1. Also, higher (but fixed) values of k in our algorithm may lead to smaller delays even at rates closer to the capacity. These simulations suggest that the probability δ in Theorem 1 of forming a matching of requisite weight is quite conservative.

VIII. Discussion

The algorithms in this paper pertain to the primary interference constraint, a widely used model for wireless scheduling. However, it is not the most general model, and indeed it would be very interesting to see if the ideas in this paper can be extended to cover more general interference models like 2-hop or k-hop. The crucial innovation would have to be a way to generate "local" updates to the existing schedule such that the probability of a local update achieving any given fraction β of the max-weight schedule will be nonzero. For primary interference, we used the augmentation structure of matchings to do this. Augmentations are an alternating structure, and one would need to find similar alternating structures for more general interference cases (e.g., one can find similar alternating sets between two independent sets in a graph).

The main motivation of this paper was to develop scheduling algorithms that can be implemented without explicit regard for the network on which it is deployed. A crucial ingredient of that is the requirement of constant overheads *every time* a schedule needs to be developed. However, if one is only concerned about long-run *average* overhead, one could conceive running a high-overhead algorithm infrequently, thus amortizing the complexity but raising the delay. This poses significant problems because, unlike in switch scheduling, in a wireless implementation: 1) the computations are decentralized, and 2) the *same* resource is used for the overhead transmissions and the data service. For example, an implementation of infrequent max-weight matching would need to detect when the global max-weight matching has been reached, then globally disseminate this information to all nodes. A little thought shows that these steps need network-wide coordination whose implementation would require complicated protocols. Finally, Section VII shows that our algorithm, run every slot, compares favorably in terms of delay, as compared to infrequently run max-weight matching (except for rates extremely close to capacity).

Our algorithm is based on the idea of making local changes to an existing schedule; each local change results in an increase in the weight. This idea has its origins in the work of Tassiulas [7]. In that paper, a new schedule is globally compared with the existing one, and the better one is chosen. This global comparison leads to a complexity growing in network size but allows the full capacity to be achieved. In our paper, the crucial step is to ensure that local changes are guaranteed to have some nonzero probability of capturing the required fraction of the maximum weight.

IX. Conclusion

This paper presents a novel class of simple distributed wireless scheduling algorithms. Our algorithms can achieve any fraction of the capacity region for data transmission and require constant overheads that do not scale with network size. Our results are also interesting from a practical standpoint due to the simple structure of our algorithms and our explicit accounting of the tradeoff between overhead and scheduling performance.

The objective of this paper is to design simple algorithms with appealing properties, not to announce a complete wireless protocol ready for implementation. In particular, careful studies need to be done to arrive at the value of k that would give the best performance in practice. This will likely depend on the maximum length a scheduling cycle can have, given mobility and other aspects of the network that have been abstracted away in our model. An interesting avenue for possible algorithmic investigation is to see how our ideas adapt to designs for more general interference constraints.

Appendix A
Proof of Theorem 2

In this Appendix, we prove Theorem 2. The proof is along the lines of [15] and [27], the main difference being that the generalization of the back-pressure algorithm requires additional work to complete the proof.

Note that the packets travel hop by hop, i.e., packets arrive to the next queue after departing from the previous queue. Hence, we assume the following convention for the dynamics of the queues: At any time slot t, based on the schedule $\pi[t-1]$, the departures from queues happen at the beginning of the time slot, then the arrivals at the end of the time slot. Thus, the evolution of queue lengths is governed by the following rules:

$$q_n^d[t+1] = \left(q_n^d[t] - \sum_{m:(n,m)\in\mathcal{L}} c_{nm}\pi_{nm}^d[t-1] \right)^+ + \sum_{f\in F(n,d)} a_f[t] + \sum_{k:(k,n)\in\mathcal{L}} \nu_{kn}^d[t-1] \quad (6)$$

where $a_f[t]$ is the number of flow f's arrivals in time slot t, and $\nu_{nm}^d[t-1]$ is the number of packets, destined to destination d, which are actually sent over link (n,m). Notice that

$$\nu_{nm}^d[t-1] \le c_{nm}\pi_{nm}^d[t-1].$$

We have that $\mathbf{y}[t] \triangleq (\mathbf{q}[t], \pi[t-1])$ defines an irreducible and aperiodic Markov chain. Then, let us consider the following Lyapunov function:

$$V(\mathbf{y}) = \underbrace{\frac{1}{1+\alpha}\sum_{d\in D}\sum_{n\in\mathcal{N}}(q_n^d)^{\alpha+1}}_{V_1(\mathbf{y})} + \underbrace{[(\mathbf{w}\cdot(\beta\pi^\star(\mathbf{w})-\pi))^+]^2}_{V_2(\mathbf{y})}.$$

We first state two lemmas, and then prove them. The proof of Theorem 2 follows thereafter.

Lemma 4: There exist positive constants γ and B_2 such that

$$\begin{aligned}\Delta_1(\mathbf{y}[t]) &\triangleq \mathbb{E}[V_1(\mathbf{y}[t+1]) - V_1(\mathbf{y}[t])|\mathbf{y}[t]] \\ &\le -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} + \sqrt{V_2(\mathbf{y}[t])} + B_2 \\ &\quad + L\sum_{f\in\mathcal{F}} U_f(x_f[t]) - L\sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)).\end{aligned}$$

Lemma 5: There exist constants B_3 and B_4 such that

$$\begin{aligned}\Delta_2(\mathbf{y}[t]) &\triangleq \mathbb{E}[V_2(\mathbf{y}[t+1]) - V_2(\mathbf{y}[t])|\mathbf{y}[t]] \\ &\le -\delta V_2(\mathbf{y}[t]) + B_3\sqrt{V_2(\mathbf{y}[t])} + B_4\end{aligned}$$

where δ is defined in Section IV-A.

Proof of Lemma 4: First, consider any queue q with the following updating rule:

$$\begin{aligned}q[t+1] &= (q[t]-\mu[t])^+ + a[t] \\ &= q[t] \underbrace{-\mu[t] + a[t] + u[t]}_{x[t]}\end{aligned}$$

where $u[t] = -q[t]+\mu[t]$ if $q[t] < \mu[t]$, and $u[t] = 0$, otherwise.

We would like to get the bound of $(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}$ as follows:

$$\begin{aligned}(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1} &= (q[t]+x[t])^{\alpha+1} - (q[t])^{\alpha+1} \\ &= (\alpha+1)(\tilde{q}[t])^\alpha x[t]\end{aligned}$$

where $\tilde{q}[t]$ is real number satisfying that $q[t] \le \tilde{q}[t] \le q[t]+x[t]$ if $x[t] \ge 0$, or $q[t]+x[t] \le \tilde{q}[t] \le q[t]$ if $x[t] < 0$. Hence, $(q[t]-a_{\max})^+ \le \tilde{q}[t] \le q[t]+a_{\max}$, where $a_{\max}$ is the upper bound of $a[t]$ for all t. Also, let $\mu_{\max}$ be the upper bound of $\mu[t]$ for all t.

One can show (by considering separately two cases when $q[t] \ge \mu_{\max}$ and when $q[t] < \mu_{\max}$) that

$$\frac{(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}}{\alpha+1} \le (q[t])^\alpha(a[t]-\mu[t]) + m_2 + m_1\frac{(q[t])^\alpha}{q[t]}\mathcal{I}_{\{q[t]\ge\mu_{\max}\}} \quad (7)$$

where $m_1 = \alpha a_{\max}^2 \max\{1, (1+\frac{a_{\max}}{\mu_{\max}})^{\alpha-1}\}$ and $m_2 = a_{\max}(a_{\max}+\mu_{\max})^\alpha + \mu_{\max}^{\alpha+1}$.

Now, to simplify the notation, let us define

$$\begin{aligned}\pi_{\text{in}(n)}^d &\triangleq \sum_{k:(k,n)\in\mathcal{L}} c_{kn}\pi_{kn}^d, \pi_{\text{out}(n)}^d \\ &\triangleq \sum_{m:(n,m)\in\mathcal{L}} c_{nm}\pi_{nm}^d.\end{aligned}$$

Since $\mathbb{E}[a_f^2[t]]$, $\pi_{\text{in}(n)}^d[t]$, and $\pi_{\text{out}(n)}^d[t]$ are all upper-bounded, we apply (7) for the queue (q_n^d) with updating the rule in (6) to get

$$\begin{aligned}\Delta_1(\mathbf{y}[t]) &\le \sum_{d\in D}\sum_{n\in\mathcal{N}} \left(q_n^d[t]\right)^\alpha \\ &\quad \times \left[\sum_{f\in F(n,d)} x_f[t] + \pi_{\text{in}(n)}^d[t-1] - \pi_{\text{out}(n)}^d[t-1]\right] \\ &\quad + \sum_{d\in D}\sum_{n\in\mathcal{N}} m_1\frac{\left(q_n^d[t]\right)^\alpha}{q_n^d[t]}\mathcal{I}_{\{q_n^d[t]\ge\mu_{\max}\}} \\ &\quad + |D||\mathcal{N}|m_2. \quad (8)\end{aligned}$$

Manipulating (8) by adding and subtracting terms, we have

$$\begin{aligned}&\Delta_1(\mathbf{y}[t]) \\ &\le \sum_{f\in\mathcal{F}} \left(q_{b(f)}^{e(f)}[t]\right)^\alpha x_f[t] - L\sum_{f\in\mathcal{F}} U_f(x_f[t]) \quad (9) \\ &\quad - \sum_{f\in\mathcal{F}} \left(q_{b(f)}^{e(f)}[t]\right)^\alpha x_f^\star(\beta) + L\sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)) \quad (10) \\ &\quad + \sum_{d\in D}\sum_{n\in\mathcal{N}} \left(q_n^d[t]\right)^\alpha \beta\left[\pi_{\text{in}(n)}^{\star d}(\mathbf{w}[t]) \right. \quad (11)\end{aligned}$$

$$-\pi^{\star d}_{\text{out}(n)}(\mathbf{w}[t])\Big] + \sum_{f\in\mathcal{F}} \left(q^{e(f)}_{b(f)}[t]\right)^{\alpha} x^{\star}_f(\beta) \tag{12}$$

$$+ \sum_{d\in D}\sum_{n\in\mathcal{N}} \left(q^d_n[t]\right)^{\alpha}\left[\pi^d_{\text{in}(n)}[t-1] - \pi^d_{\text{out}(n)}[t-1]\right] \tag{13}$$

$$- \sum_{d\in D}\sum_{n\in\mathcal{N}} \left(q^d_n[t]\right)^{\alpha}\beta\left[\pi^{\star d}_{\text{in}(n)}(\mathbf{w}[t]) - \pi^{\star d}_{\text{out}(n)}(\mathbf{w}[t])\right]$$
$$+ L\sum_{f\in\mathcal{F}}(U_f(x_f[t]) - U_f(x^{\star}_f(\beta)))$$
$$+ \sum_{d\in D}\sum_{n\in\mathcal{N}} m_1\frac{\left(q^d_n[t]\right)^{\alpha}}{q^d_n[t]}\mathcal{I}_{\{q^d_n[t]\geq\mu_{\max}\}} + |D||\mathcal{N}|m_2 \tag{14}$$

where $\pi^{\star d}_{\text{in}(n)}(\mathbf{w}[t])$ and $\pi^{\star d}_{\text{out}(n)}(\mathbf{w}[t])$ are defined similarly to $\pi^d_{\text{in}(n)}$ and $\pi^d_{\text{out}(n)}$, respectively.

First, by the congestion control algorithm (Section IV-B), we have that (9) + (10) ≤ 0.

Next, note that $\mathbf{x}^{\star}(\beta)$ is within the region $\beta\Lambda$. By definition of the capacity region, there exists a vector $\{\mu^d_{nm}\}^{d\in D}_{(n,m)\in\mathcal{L}}$ such that:

- $\sum_{f\in F(n,d)} x^{\star}_f(\beta) = \beta(\mu^d_{\text{out}(n)} - \mu^d_{\text{in}(n)})$, where $\mu^d_{\text{in}(n)}$ and $\mu^d_{\text{out}(n)}$ are defined similarly to $\pi^d_{\text{in}(n)}$ and $\pi^d_{\text{out}(n)}$, respectively, and
- $\{\sum_{d\in D}\mu^d_{nm}\} = \sum_{i=1}^{|\mathcal{M}|}\alpha_i\pi^i, \sum_{i=1}^{|\mathcal{M}|}\alpha_i < 1, \quad \alpha_i \geq 0, \forall i.$

Therefore, we have

$$\begin{aligned}(11)+(12) &= -\beta\sum_{d\in D}\sum_{n\in\mathcal{N}}\left(q^d_n[t]\right)^{\alpha}\Big(\pi^{\star d}_{\text{out}(n)}(\mathbf{w}[t]) \\ &\quad -\pi^{\star d}_{\text{in}(n)}(\mathbf{w}[t]) - \mu^d_{\text{out}(n)} + \mu^d_{\text{in}(n)}\Big) \\ &\leq -\beta\left(1 - \sum_{i=1}^{|\mathcal{M}|}\alpha_i\right)\sum_{(n,m)\in\mathcal{L}}\pi^{\star}_{nm}(\mathbf{w}[t]) \\ &\quad \times c_{nm}\max_d\Big(\left(q^d_n[t]\right)^{\alpha} - \left(q^d_m[t]\right)^{\alpha}\Big)\end{aligned}$$

where the last inequality is because $\pi^{\star}(\mathbf{w}[t])$ is the optimal matching given $\mathbf{q}[t]$. Following similar steps as in [5, p. 1947], we can get

$$\sum_{(n,m)\in\mathcal{L}} c_{nm}\pi^{\star}_{nm}(\mathbf{w}[t])\max_d\Big(\left(q^d_n[t]\right)^{\alpha} - \left(q^d_m[t]\right)^{\alpha}\Big) \geq \frac{1}{l_{\max}}\left[(1+\alpha)\frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|}\right]^{\frac{\alpha}{\alpha+1}}\min_{(n,m)\in\mathcal{L}} c_{nm}.$$

Thus, there exists a positive constant ϵ such that

$$(11)+(12) \leq -\epsilon|D||\mathcal{N}|\left[(1+\alpha)\frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|}\right]^{\frac{\alpha}{\alpha+1}}.$$

Finally, we can rewrite (13) + (14) as

$$\begin{aligned}(13)+(14) &= \sum_{d\in D}\sum_{(n,m)\in\mathcal{L}}\left[\beta\pi^{\star d}_{nm}(\mathbf{w}[t]) - \pi^d_{nm}[t-1]\right] \\ &\quad \times\left[\left(q^d_n[t]\right)^{\alpha} - \left(q^d_m[t]\right)^{\alpha}\right] \\ &\leq \sum_{(n,m)\in\mathcal{L}}(\beta\pi^{\star}_{nm}(\mathbf{w}[t]) - \pi_{nm}[t-1])w_{nm}[t] \\ &\leq \sqrt{V_2(\mathbf{y}[t])}.\end{aligned}$$

Combining these results yields

$$\begin{aligned}\Delta_1(\mathbf{y}[t]) &\leq -\epsilon|D||\mathcal{N}|\left[(1+\alpha)\frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|}\right]^{\frac{\alpha}{\alpha+1}} \\ &\quad + L\sum_{f\in\mathcal{F}}U_f(x_f[t]) - L\sum_{f\in\mathcal{F}}U_f(x^{\star}_f(\beta)) \\ &\quad + \sum_{d\in D}\sum_{n\in\mathcal{N}} m_1\frac{\left(q^d_n[t]\right)^{\alpha}}{q^d_n[t]}\mathcal{I}_{\{q^d_n[t]\geq\mu_{\max}\}} \\ &\quad + \sqrt{V_2(\mathbf{y}[t])} + |D||\mathcal{N}|m_2. \end{aligned}\tag{15}$$

Now, fix a constant $\tilde{\epsilon}$ such that $0 < \tilde{\epsilon} < \epsilon$. It is easy to see that if $q > K_1$, where $K_1 := \max\{\mu_{\max}, \frac{m_1}{\epsilon-\tilde{\epsilon}}\}$,, then $\frac{m_1}{q}\mathcal{I}_{\{q\geq\mu_{\max}\}} < \epsilon - \tilde{\epsilon}$. Therefore, we have

$$(15) \leq (\epsilon-\tilde{\epsilon})\sum_{d\in D}\sum_{n\in\mathcal{N}}\left(q^d_n[t]\right)^{\alpha}\mathcal{I}_{\{q^d_n[t]>K_1\}} + \frac{|D||\mathcal{N}|K_1}{\mu_{\max}}.$$

Also, note that Jensen's inequality yields

$$\frac{1}{|D||\mathcal{N}|}\sum_{d\in D}\sum_{n\in\mathcal{N}}\left(q^d_n[t]\right)^{\alpha} \leq \left[\frac{1+\alpha}{|D||\mathcal{N}|}V_1(\mathbf{y}[t])\right]^{\frac{\alpha}{\alpha+1}}.$$

Therefore, after some manipulation, we get the result

$$\begin{aligned}\Delta_1(\mathbf{y}[t]) &\leq -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} + \sqrt{V_2(\mathbf{y}[t])} + B_2 \\ &\quad + L\sum_{f\in\mathcal{F}}U_f(x_f[t]) - L\sum_{f\in\mathcal{F}}U_f(x^{\star}_f(\beta))\end{aligned}$$

where

$$\gamma = \tilde{\epsilon}(|D||\mathcal{N}|)^{\frac{1}{1+\alpha}}(1+\alpha)^{\frac{\alpha}{1+\alpha}}$$
$$B_2 = |D||\mathcal{N}|\left(m_2 + \frac{K_1}{\mu_{\max}}\right).$$

■

Proof of Lemma 5: Given $\mathbf{y}[t] = (\mathbf{q}[t], \pi[t-1])$, let $\mathcal{E}_\beta$ be the event

$$\{\mathbf{w}[t]\cdot\pi[t] \geq \beta\mathbf{w}[t]\cdot\pi^{\star}(\mathbf{w}[t])\}.$$

Then

$$\begin{aligned}&\mathbb{E}[V_2(\mathbf{y}(t+1)|\mathbf{y}(t)] \\ &\quad = P[\mathcal{E}_\beta|\mathbf{y}(t)]\,\mathbb{E}[V_2(\mathbf{y}(t+1))|\mathbf{y}(t),\mathcal{E}_\beta] \\ &\qquad + P\left[\mathcal{E}^c_\beta|\mathbf{y}(t)\right]\mathbb{E}[V_2(\mathbf{y}(t+1))|\mathbf{y}(t),\mathcal{E}^c_\beta] \\ &\quad \leq 0.P[\mathcal{E}_\beta|\mathbf{y}(t)] + (1-\delta)\mathbb{E}\left[V_2(\mathbf{y}(t+1))|\mathbf{y}(t),\mathcal{E}^c_\beta\right] \\ &\quad = (1-\delta)\mathbb{E}\left[\{(\beta\pi^{\star}(\mathbf{w}[t+1]) - \pi[t]) \right. \\ &\qquad \left. \times\mathbf{w}[t+1]\}^2|\mathbf{y}[t],\mathcal{E}^c_\beta\right].\end{aligned}$$

Next, we write the evolution of the weight vector $\mathbf{w}[t]$ as

$$\mathbf{w}[t+1] = \mathbf{w}[t] + \mathbf{r}[t] - \mathbf{z}[t]$$

where $\mathbf{r}[t]$ denotes the vector of packets added to $\mathbf{w}[t]$, and $\mathbf{z}[t]$ denotes the vector of packets subtracted from $\mathbf{w}[t]$. Note that both of them are upper-bounded. Hence, we can write

$$\begin{aligned}&\mathbb{E}[V_2(\mathbf{y}(t+1)|\mathbf{y}(t)] \\ &\quad \leq (1-\delta)\mathbb{E}\left[\{\mathbf{w}[t](\beta\pi^{\star}(\mathbf{w}[t+1]) - \pi[t]) \right. \\ &\qquad \left. +(\mathbf{r}[t]-\mathbf{z}[t])(\beta\pi^{\star}(\mathbf{w}[t+1]) - \pi[t])\}^2|\mathbf{y}[t],\mathcal{E}^c_\beta\right].\end{aligned}$$

We have that the latter term is upper-bounded by a finite constant. To bound the first term, we have the following observations:

- $\mathbf{w}[t]\cdot\pi^\star(\mathbf{w}[t+1]) \le \mathbf{w}[t]\cdot\pi^\star(\mathbf{w}[t])$ since $\pi^\star(\mathbf{w}[t])$ is the optimal weighted matching corresponding to the weight vector $\mathbf{w}[t]$;
- $\mathbf{w}[t]\cdot\pi[t] \ge \mathbf{w}[t]\cdot\pi[t-1]$ by the inequality (4).

Combining these, we can find finite positive constants B_3, B_4 such that

$$\mathbb{E}[V_2(\mathbf{y}(t+1))|\mathbf{y}(t)] \le (1-\delta)V_2(\mathbf{y}[t]) + B_3\sqrt{V_2(\mathbf{y}[t])} + B_4$$

which completes the proof. ∎

Proof of Theorem 2: Combining the results of Lemmas 4 and 5, we have

$$\begin{aligned}\Delta(\mathbf{y}[t]) &\triangleq \mathbb{E}[V(\mathbf{y}[t+1]) - V(\mathbf{y}[t])|\mathbf{y}[t]] \\ &\le -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) \\ &\quad + L\sum_{f\in\mathcal{F}} U_f(x_f[t]) - L\sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)) \\ &\quad + (1+B_3)\sqrt{V_2(\mathbf{y}[t])} + B_2 + B_4. \end{aligned} \tag{16}$$

Since $x_f[t] \le M, \forall f\in\mathcal{F}, \forall t$, there exists a constant $U_{\max}$ such that $U_f(x_f[t]) \le U_{\max}, \forall f\in\mathcal{F}, \forall t$. Thus

$$\begin{aligned}\Delta(\mathbf{y}[t]) \le -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) + L|\mathcal{F}|U_{\max} \\ +(1+B_3)\sqrt{V_2(\mathbf{y}[t])} + B_2 + B_4.\end{aligned}$$

Let us consider the set $\{\mathbf{y} : V(\mathbf{y}) < K\}$, which is a finite subset of the state space of $\mathbf{y}$. We will show that $V(\mathbf{y}[t])$ has negative drift outside this set for K large enough.

If $V(\mathbf{y}[t]) \ge K$, then $V_1(\mathbf{y}[t]) \ge (K - V_2(\mathbf{y}[t]))^+$, and, hence, the above inequality becomes

$$\begin{aligned}\Delta(\mathbf{y}[t]) \le -\frac{\gamma}{2}[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \frac{\gamma}{2}[(K - V_2(\mathbf{y}[t]))^+]^{\frac{\alpha}{\alpha+1}} \\ +(1+B_3)\sqrt{V_2(\mathbf{y}[t])} - \delta V_2(\mathbf{y}[t]) + B_5\end{aligned}$$

where $B_5 = B_2 + B_4 + L|\mathcal{F}|U_{\max}$. Now, for K large enough, we can get the bound

$$\begin{aligned}0 \ge -\frac{\gamma}{2}[(K - V_2(\mathbf{y}[t]))^+]^{\frac{\alpha}{\alpha+1}} \\ +(1+B_3)\sqrt{V_2(\mathbf{y}[t])} - \delta V_2(\mathbf{y}[t]) + B_5.\end{aligned}$$

Thus, $\Delta(\mathbf{y}[t]) \le -\frac{\gamma}{2}[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}}$ if $V(\mathbf{y}[t]) \ge K$. Given this condition, Foster's criterion implies the stability of the Markov chain $\mathbf{y}[t]$, and, thus, of the queues $q_n^d[t]$'s.

Next, note that we can easily find a constant B_1 such that

$$\begin{aligned}B_1 \ge -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) \\ +(1+B_3)\sqrt{V_2(\mathbf{y}[t])} + B_2 + B_4.\end{aligned}$$

Then, inequality (16) becomes

$$\Delta(\mathbf{y}[t]) \le B_1 + L\sum_{f\in\mathcal{F}} U_f(x_f[t]) - L\sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)).$$

Taking expectations of both sides and summing over $t = 0, 1, \ldots, T-1$, we get

$$\begin{aligned}&\mathbb{E}[V(\mathbf{y}(T)) - V(\mathbf{y}(0))] \\ &\le L\sum_{t=0}^{T-1}\sum_{f\in\mathcal{F}} \mathbb{E}[U_f(x_f[t])] \\ &\quad - LT\sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)) + TB_1\end{aligned}$$

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}\sum_{f\in\mathcal{F}} \mathbb{E}[U_f(x_f[t])] \\ &\ge \sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)) + \frac{\mathbb{E}[V(\mathbf{y}(T))]}{LT} \\ &\quad - \frac{\mathbb{E}[V(\mathbf{y}(0))]}{LT} - \frac{B_1}{L}.\end{aligned}$$

Taking the limit as T goes to infinity, and applying Jensen's inequality, we get the result

$$\sum_{f\in\mathcal{F}} U_f(\bar{x}_f) \ge \sum_{f\in\mathcal{F}} U_f(x_f^\star(\beta)) - \frac{B_1}{L}.$$

∎

REFERENCES

[1] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Proc. IEEE Conf. Decision Control*, San Diego, CA, Dec. 2006, pp. 1258–1263.

[2] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 1631–1639.

[3] C. Joo and N. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 19–27.

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

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

[6] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, vol. 1, pp. 296–302.

[7] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998, vol. 2, pp. 533–539.

[8] P. Giaccone, B. Prabhakar, and D. Shah, "Towards simple, high-performance schedulers for high-aggregate bandwidth switches," in *Proc. IEEE INFOCOM*, New York, NY, Jun. 2002, vol. 3, pp. 1160–1169.

[9] T. Weller and B. Hajek, "Scheduling non uniform traffic in a packet switching system with small propagation delay," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 813–823, Dec. 1997.

[10] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000, vol. 2, pp. 556–564.

[11] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 3, pp. 1804–1814.

[12] S. Sarkar and K. Kar, "Achieving 2/3 throughput approximation with sequential maximal scheduling under primary interference constraints," in *Proc. Allerton Conf. Commun., Control Comput.*, Monticello, IL, Sep. 2006, pp. 729–740.

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

[14] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks: A partitioning approach," in *Proc. ACM MOBICOM*, Los Angeles, CA, Sep. 2006, pp. 26–37.

[15] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial complexity algorithms for full utilization of multi-hop wireless networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 499–507.

[16] Y. Yi, G. de Veciana, and S. Shakkottai, "Learning contention patterns and adapting to load/topology changes in a MAC scheduling algorithm," in *Proc. WiMesh*, 2006, pp. 23–32.

[17] X. Wu and R. Srikant, "Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing," in *Proc. IEEE Conf. Decision Control*, Dec. 2005, pp. 5342–5347.
[18] P. Chaporkar, K. Kar, and S. Sarkar, "Achieving queue length stability through maximal scheduling in wireless networks," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2006.
[19] X. Wu, R. Srikant, and J. Perkins, "Queue length stability of maximal greedy schedules in wireless networks," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2006.
[20] S. Ray and S. Sarkar, "Arbitrary throughput versus complexity tradeoffs in wireless networks using graph partitioning," in *Proc. Inf. Theory Appl. Workshop*, Jan. 2007.
[21] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.
[22] D. Shah and D. Wischik, "Optimal scheduling algorithms for input-queued switches," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.
[23] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, San Francisco, CA, 1990, pp. 434–443.
[24] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Syst.*, vol. 50, no. 4, pp. 401–457, Aug. 2005.
[25] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1794–1803.
[26] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," in *IEEE J. Sel. Areas Commun.*, Aug. 2006, vol. 24, no. 8, pp. 1514–1524.
[27] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1723–1734.
[28] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE Conf. Decision Control*, Paradise Island, Bahamas, Dec. 2004, pp. 1484–1489.
[29] S. Pettie and P. Sanders, "A simpler linear time 2/3-ϵ approximation for maximum weight matching," *Inf. Process. Lett.*, vol. 91, no. 6, pp. 271–276, 2004.

Loc X. Bui received the B.S. degree in electronics and telecommunications from the Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam, in 2003, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, in 2006 and 2008, respectively.

Since October 2008, he has been with Airvana Inc., Chelmsford, MA, where he currently is a Software Engineer. His research interests include communication networks, wireless communications, network control, and optimization.

Sujay Sanghavi received the B.Tech. degree from the Indian Institute of Technology, Bombay, India, in 2000, and the M.S. degree in electrical and computer engineering (ECE), the M.S. degree in mathematics, and the Ph.D. degree in ECE from the University of Illinois at Urbana-Champaign in 2002, 2005, and 2006, respectively.

He joined the faculty of the Department Electrical and Computer Engineering at Purdue University, West Lafayette, IN, in Fall 2008 as an Assistant Professor. Prior to that, he spent two years as a Post-Doctorate Fellow in the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge. His research interests lie in probability, algorithms, and optimization, as applied to large-scale problems in communications, networking, and signal processing.

R. Srikant (S'90–M'91–SM'01–F'06) received the B.Tech. degree from the Indian Institute of Technology, Madras, India, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 1988 and 1991, respectively.

He was a Member of Technical Staff at AT&T Bell Laboratories from 1991 to 1995. He is currently with the University of Illinois at Urbana-Champaign, where he is a Professor in the Department of Electrical and Computer Engineering and a Research Professor in the Coordinated Science Laboratory. His research interests include communication networks, stochastic processes, queueing theory, information theory, and game theory.

Prof. Srikant was an Associate Editor of *Automatica*, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, and IEEE/ACM TRANSACTIONS ON NETWORKING. He has also served on the Editorial Boards of special issues of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and IEEE TRANSACTIONS ON INFORMATION THEORY. He was the Chair of the 2002 IEEE Computer Communications Workshop, Santa Fe, NM, and a program Co-Chair of IEEE INFOCOM 2007.