

Distributed Link Scheduling with Constant Overhead

Loc Bui, Sujay Sanghavi, and R. Srikant

Abstract—This paper proposes a new class of simple, distributed algorithms for joint congestion control, routing and scheduling in multihop wireless networks. 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. The multihop congestion controller injects packets into the network based only on the queues at the sources of the flows.

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 trade-off in practice.

Index Terms—wireless networks, congestion control, scheduling, distributed algorithms, primary interference, matchings, multihop

I. INTRODUCTION

THIS paper presents novel distributed algorithms for joint congestion control, routing and 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. 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.

This research was supported by a Vodafone fellowship, NSF grants CCF 06-34891, CNS 07-21286 and the DARPA CBMANET project.

An earlier version of this paper appeared in the Proceedings of ACM SIGMETRICS, June 2007.

L. Bui and R. Srikant are with the Department of Electrical and Computer Engineering and Coordinated Science Lab, University of Illinois at Urbana-Champaign, e-mails: {locbui, rsrikant}@uiuc.edu.

S. Sanghavi is with Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, e-mail: sanghavi@mit.edu.

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 signalling, the algorithms perform well with regards to the benchmark in the *remaining* portion used for data transmission.

In our paper we carefully take overheads into account a priori in the performance evaluation. For our algorithms it is clear (i) how efficiently the overall wireless resources are utilized, (ii) what is the tradeoff between scheduling performance and control overhead, and (iii) how a system designer can choose her operating point on this tradeoff. These three aspects are elaborated on 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 doing 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 signalling) part and a service (data-transmission) part. Figure 1 depicts this partition. The partitioning of the scheduling cycle thus explicitly captures the wastage in control signalling: 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

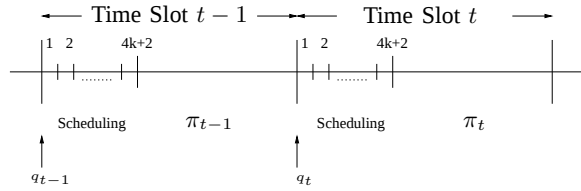


Fig. 1. Each scheduling cycle is divided into a control part and a data part. The control part consists of $4k + 2$ phases – each phase being the length of a round-trip between neighbors – after which a new set π of active links will be decided. This new set is active for the data part of the cycle.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

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.

The 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 $\left(\frac{k}{k+2}\right)$ 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*.

The paper is organized as follows. In Section II we survey some of the existing literature most relevant to our paper, and compare our results. Section III lays out the formal system model for scheduling in the presence of primary interference. Section IV presents the joint congestion control, routing and scheduling algorithm, and states the main results of the paper. Section V describes the scheduling algorithm that forms the core of our protocol, along with examples, illustrations and discussion. Section VI contains the proof of the main result.

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 de-randomized 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 which achieves $2/3$ 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 $1/2$ 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 multi-hop traffic, e.g. [17], or more general

interference constraints, e.g. [18]–[20].

III. SYSTEM MODEL

We now formally describe our system model, which is by 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 which 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., $D = \{d \in \mathcal{N} : d = e(f) \text{ for some } f \in \mathcal{F}\}$. 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 destination¹. 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, non-decreasing, 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, any node only communicates with at most one other node in any time slot. This means 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 $\mathcal{M} = \{\pi^1, \pi^2, \dots, \pi^{|\mathcal{M}|}\}$ 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], [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:

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

where $co(\mathcal{M})$ 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 $\mathbf{x}^*$ satisfying:

$$\begin{aligned} \mathbf{x}^* &\in \arg \max \sum_{f \in \mathcal{F}} U_f(x_f) \\ \text{s.t. } &\mathbf{x} \in \Lambda. \end{aligned} \quad (1)$$

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

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

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 -sub-optimal solution $\mathbf{x}^*(\beta)$ is defined as the optimal solution to the following optimization problem:

$$\begin{aligned} \mathbf{x}^*(\beta) &\in \arg \max \sum_{f \in \mathcal{F}} U_f(x_f) \\ \text{s.t. } &\mathbf{x} \in \beta\Lambda. \end{aligned} \quad (2)$$

In this paper, we propose a joint congestion control, routing, and scheduling mechanism that achieves the β -sub-optimal solution $\mathbf{x}^*(\beta)$, 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

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

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

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

Note that by letting $\alpha = 1$, we get back to the differential backlog weight which is usually used in the literature. Hence, the weight form that we consider here is more general. It is

¹Our 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

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 $\pi^*(\mathbf{w}[t])$ denote the optimal matching corresponding to the weight vector $\mathbf{w}[t]$, i.e.,

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

It is well-known that if the matching $\pi^*(\mathbf{w}[t])$ is used at each step, then the resulting network throughput is optimal. However, computing $\pi^*(\mathbf{w}[t])$ is computationally prohibitive (the complexity in worst-case 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 π^* with a certain probability, has low overhead, low complexity, and can be implemented in distributed manner. The algorithm is parameterized by a parameter k that provides a trade-off 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 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 which 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.,

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

- 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.,

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

- 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:

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

- 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., $q_{bf}^{ef}[t]$. Then, the

instantaneous mean data rates $x_f[t]$ of flow f is set as follows:

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

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 non-negative real number, the number of packets injected into the network has to be a non-negative integer which 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 allow 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:

$$\sum_{f \in \mathcal{F}} U_f(\bar{x}_f) \geq \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) - \frac{C_1}{L},$$

where $\bar{x}_f = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[x_f[t]]$, and C_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 [24]–[27]. The reader is also referred to the related work in [28]. For details, see Appendix A. ■

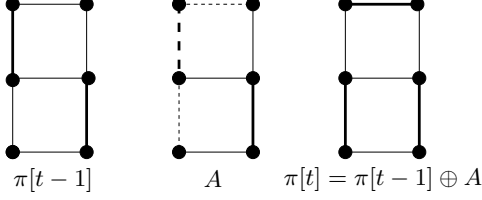
V. THE MATCHING AUGMENTATION SCHEDULING ALGORITHM

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

²One 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.

in $A \cap \pi$ are removed from π and all links in $A \setminus \pi$ are added³ 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 the example below.



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

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

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 augmentations is the sum of the gains 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 Figure 2, which depicts our algorithm operating in phases on a simple graph. The bold links in the top left graph of Figure 2 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:

³Here $A \setminus \pi$ 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.

⁵The 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.

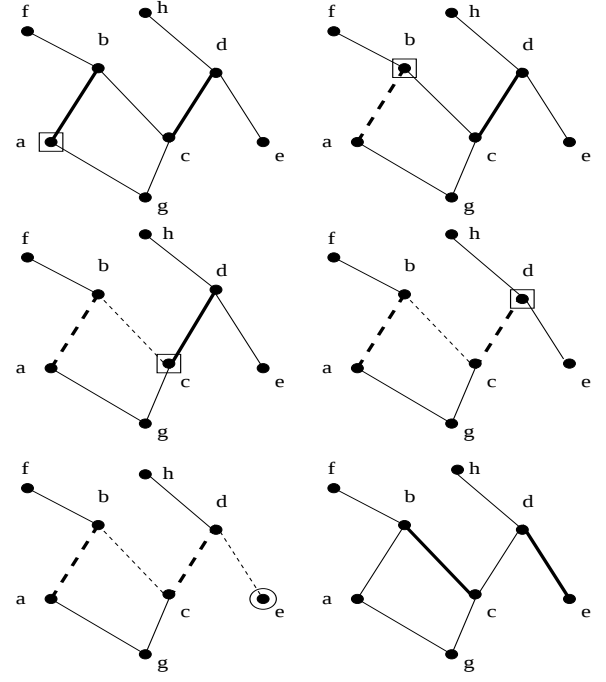


Fig. 2. Augmentation building in our algorithm (Example 1). The $\square$ indicates active nodes. Dashed lines are links in A . Bold lines in the first 5 graphs are links in $\pi[t-1]$, and in the last graph are links in $\pi[t-1] \oplus A$. $\circ$ indicates terminus.

- 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.

Node b can choose to add any one of the links (b,c) or (b,f) . Say it chooses (b,c) . So, 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. So 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 above, where bold links are the ones in $\pi[t-1] \oplus A$. ■

The above example illustrates the simple basic idea underlying our algorithm, namely: (i) randomly seed and grow disjoint augmentations, and (ii) 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 makes $\pi[t]$ from $(\mathbf{w}[t], \pi[t-1])$. A more realistic, and more detailed, example and a brief discussion follows 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 $aug(v)$ 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 $aug(v)$. For any active v except the seed, one of its links will be in $aug(v)$ 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 .

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 $\{1, \dots, k\}$.

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 $aug(s)$ 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 $aug(v)$ as follows:
 - a) If $aug(v)$ 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 $aug(v)$. Also, v sends a REQ to u along (v, u) . If no such link exists, v becomes *terminus*.
 - b) If $aug(v)$ needs a new link outside $\pi[t-1]$, but $size(aug(v))$ 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 and

- a) any another active u also sends REQ to w in that phase, 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 $aug(v)$. 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 $aug(w)$ began and ended with links in $\pi[t-1]$,
- c) if $aug(w)$ has not reached its intended size.

If all of above are true, link (w, v) is added to $aug(w)$. Also, in either case, every terminus w evaluates $gain_t(aug(w))$ and makes the decision of switching if and only if $gain_t(aug(w)) > 0$.

- 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 non-interfering. After phase $4k+2$, all nodes in augmentation implement decision.

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* 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 $aug(v)$ 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 being already a member of $aug(v)$ 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. Thus all augmentations are consistent and disjoint. Note of course that any link (z, u) that is in $\pi[t-1]$ *can* be added to another augmentation. This however does not hurt the disjointness.

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 Figure 3. 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 also 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 $\pi[t-1]$

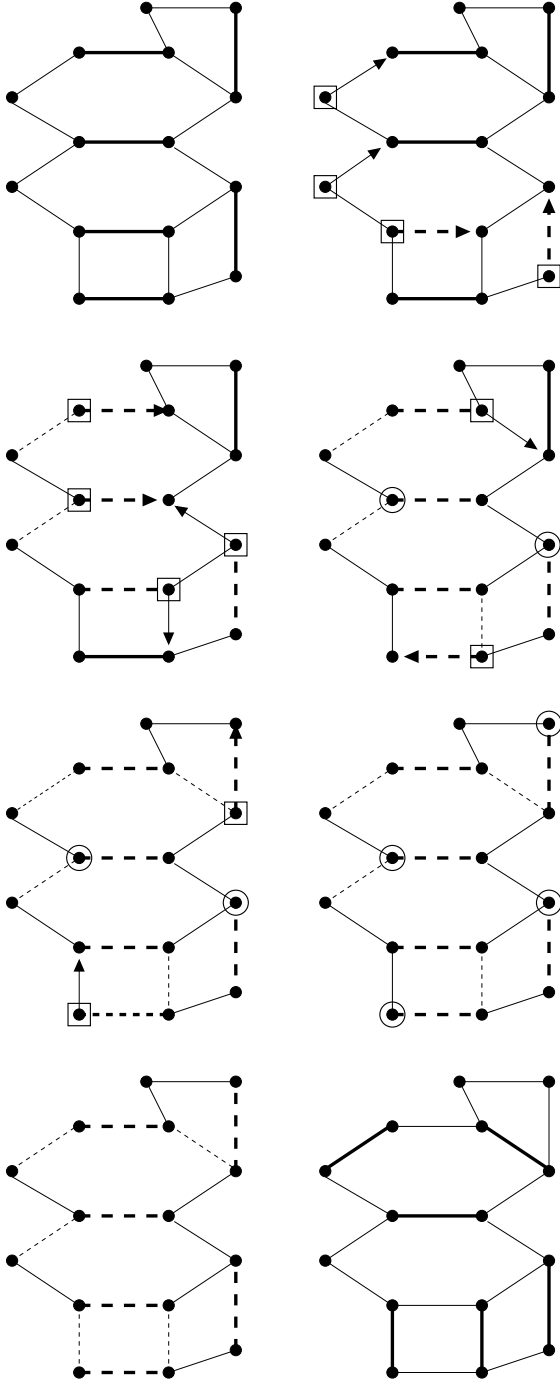


Fig. 3. The working of our algorithm in phases (Example 2). Bold lines in first seven graphs are links in $\pi[t-1]$, in the last graph are links in $\pi[t]$. Dashed lines are links in augmentations. $\square$ denotes active nodes and $\circ$ denotes terminus nodes. Arrows depict the REQ signals in each phase.

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 possible 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, 2 and 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

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

- If $k < \lceil \frac{L_{max}-1}{2} \rceil$, then

$$(\pi \oplus \mathcal{A}^*) \cdot \mathbf{w} \geq \left(\frac{k}{k+2} \right) \pi^*(\mathbf{w}) \cdot \mathbf{w}$$

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

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 towards 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* $S := \pi \triangle \pi^*(\mathbf{w})$ of matchings π and $\pi^*(\mathbf{w})$ is the set of links that are in exactly one of π or $\pi^*(\mathbf{w})$. 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 $C_1 := C \cap \pi^*(\mathbf{w})$ 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 $e_1, e_2, \dots$ 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 $(k+1)^{\text{th}}$ 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 $k+1$. Figure 4 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 $\mathcal{A}_1, \dots, \mathcal{A}_{k+1}$, made from the links $e_1, \dots, e_{k+1}$ above, 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,

$$\sum_{i=1}^{k+1} \text{gain}(\mathcal{A}_i) = k C_1 \cdot \mathbf{w} - (k+1) C_2 \cdot \mathbf{w}$$

which means that there exists at least one j such that

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

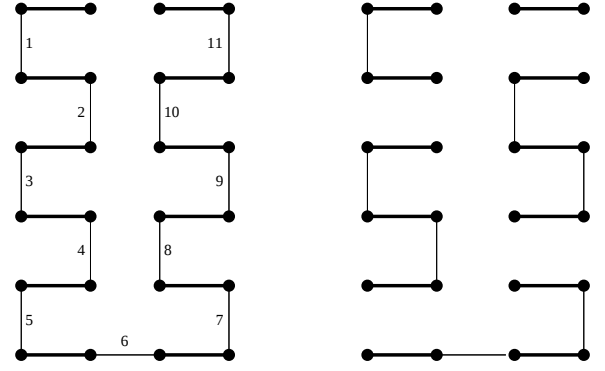


Fig. 4. The path on the left is a path component C of S , with non-bold links denoting $C_1 = C \cap \pi^*(\mathbf{w})$ and bold ones for $C_2 = C \cap \pi$. Links in C_1 are numbered, and $k = 2$. C is cut up starting at link number 2, going anti-clockwise, to obtain the set $\mathcal{A}_2$ on the right. Note that $\mathcal{A}_2$ contains disjoint augmentations of size at most $k = 2$.

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

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 $\hat{C} = C - e$, and define $\hat{C}_1 := \hat{C} \cap \pi^*(\mathbf{w})$ and $\hat{C}_2 := \hat{C} \cap \pi$ as before. Obviously, $C_2 = \hat{C}_2$. Also, since e was chosen to be the link in C_1 with smallest weight and $|C_1| = \frac{\text{size}(C)}{2} \geq k+1$,

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

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

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

Setting $\mathcal{A}_C = \mathcal{A}_{\hat{C}}$ 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:

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

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

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

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

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. $\blacksquare$

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,

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

So, 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 $p^l(1-p)^{n-l}$. Further, the probability that the intended lengths are exactly as chosen is k^{-l} , since they are chosen uniformly from $1, \dots, k$. 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

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

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 $\delta = \min \left\{ 1, \left(\frac{p}{1-p} \right)^n \right\} \left(\frac{1-p}{k\Delta} \right)^n$ completes the proof of the theorem. $\blacksquare$

VII. CONCLUSIONS

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 the due to 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 PERFORMANCE ANALYSIS

In this appendix, we prove Theorem 2. The proof is along the lines of [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

time slot, then the arrivals at the end of 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

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

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:

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

The outline of the proof of Theorem 2 is as follows: we first show that this $V(\mathbf{y})$ has (a) bounded drift when $\mathbf{y}$ belongs to a compact region containing the origin, and also (b) has drift strictly smaller (i.e. more negative) than a fixed negative number whenever $\mathbf{y}$ is outside this compact region. By the Foster-Lyapunov criterion, this implies that the Markov chain $\mathbf{y}[t]$ is stable, and hence the queues are also stable. After establishing this, we show that the average flow rates are as large as specified in the statement of the theorem.

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

Lemma 4: There exist positive constants γ and c_1 such that

$$\begin{aligned} \Delta_1(\mathbf{y}[t]) &\triangleq \mathbb{E}[V_1(\mathbf{y}[t+1]) - V_1(\mathbf{y}[t]) | \mathbf{y}[t]] \\ &\leq -\gamma[V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} + \sqrt{V_2(\mathbf{y}[t])} + C_2 \\ &\quad + L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)). \end{aligned}$$

Lemma 5: There exist constants C_3 and C_4 such that

$$\begin{aligned} \Delta_2(\mathbf{y}[t]) &\triangleq \mathbb{E}[V_2(\mathbf{y}[t+1]) - V_2(\mathbf{y}[t]) | \mathbf{y}[t]] \\ &\leq -\delta V_2(\mathbf{y}[t]) + C_3 \sqrt{V_2(\mathbf{y}[t])} + C_4. \end{aligned}$$

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 $q[t] \leq \tilde{q}[t] \leq q[t] + x[t]$ if $x[t] \geq 0$, or $q[t] + x[t] \leq \tilde{q}[t] \leq q[t]$ if $x[t] < 0$. Hence, $(q[t] - a_{max})^+ \leq \tilde{q}[t] \leq q[t] + a_{max}$ where a_{max} is the upper-bound of $a[t]$ for all t . Also, let μ_{max} be the upper-bound of $\mu[t]$ for all t .

Case 1: $q[t] \geq \mu_{max}$. Then $u[t] = 0$ and $x[t] = a[t] - \mu[t]$.

$$\begin{aligned} &\frac{1}{\alpha+1} [(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}] \\ &= (\tilde{q}[t])^\alpha x[t] \\ &\leq (q[t])^\alpha \left(1 + \frac{a_{max}}{q[t]} \right)^\alpha (a[t] - \mu[t]) \\ &= (q[t])^\alpha \left(1 + \frac{\alpha a_{max}}{q[t]} \xi^{\alpha-1} \right) (a[t] - \mu[t]), \end{aligned}$$

where ξ is a real number such that $1 \leq \xi \leq 1 + \frac{a_{max}}{q[t]} \leq 1 + \frac{a_{max}}{\mu_{max}}$. Hence, $\xi^{\alpha-1} \leq \max \left\{ 1, \left(1 + \frac{a_{max}}{\mu_{max}} \right)^{\alpha-1} \right\}$, and then

$$\begin{aligned} &\frac{1}{\alpha+1} [(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}] \\ &= (q[t])^\alpha (a[t] - \mu[t]) + m_1 \frac{(q[t])^\alpha}{q[t]}, \end{aligned}$$

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

Case 2: $q[t] < \mu_{max}$. Note that:

$$\begin{aligned} x[t] &\leq a[t] \leq a_{max}, \\ 0 &\leq \tilde{q}[t] \leq q[t] + a_{max} \leq \mu_{max} + a_{max}. \end{aligned}$$

Therefore,

$$\begin{aligned} &\frac{1}{\alpha+1} [(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}] \\ &= (\tilde{q}[t])^\alpha x[t] \\ &\leq a_{max} (a_{max} + \mu_{max})^\alpha \\ &\leq a_{max} (a_{max} + \mu_{max})^\alpha + \mu_{max}^{\alpha+1} \\ &\quad + (q[t])^\alpha (a[t] - \mu[t]). \end{aligned}$$

Combining both cases, we have the following bound:

$$\begin{aligned} &\frac{1}{\alpha+1} [(q[t+1])^{\alpha+1} - (q[t])^{\alpha+1}] \\ &\leq (q[t])^\alpha (a[t] - \mu[t]) \\ &\quad + c_1 \frac{(q[t])^\alpha}{q[t]} \mathcal{I}_{\{q[t] \geq \mu_{max}\}} + m_2. \end{aligned} \quad (7)$$

where $m_2 = a_{max} (a_{max} + \mu_{max})^\alpha + \mu_{max}^{\alpha+1}$.

Now, to simplify the notation, let us define:

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

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

in (6) to get:

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

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

$$\Delta_1(\mathbf{y}[t]) \leq \sum_{f \in \mathcal{F}} (q_{b(f)}^{e(f)}[t])^\alpha x_f[t] - L \sum_{f \in \mathcal{F}} U_f(x_f[t]) \quad (9)$$

$$- \sum_{f \in \mathcal{F}} (q_{b(f)}^{e(f)}[t])^\alpha x_f^*(\beta) + L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) \quad (10)$$

$$+ \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \beta \left[\pi_{in(n)}^{*d}(\mathbf{w}[t]) - \pi_{out(n)}^{*d}(\mathbf{w}[t]) \right] \quad (11)$$

$$+ \sum_{f \in \mathcal{F}} (q_{b(f)}^{e(f)}[t])^\alpha x_f^*(\beta) \quad (12)$$

$$+ \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \left[\pi_{in(n)}^d[t-1] - \pi_{out(n)}^d[t-1] \right] \quad (13)$$

$$- \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \beta \left[\pi_{in(n)}^{*d}(\mathbf{w}[t]) - \pi_{out(n)}^{*d}(\mathbf{w}[t]) \right]$$

$$+ L \sum_{f \in \mathcal{F}} (U_f(x_f[t]) - U_f(x_f^*(\beta)))$$

$$+ \sum_{d \in D} \sum_{n \in \mathcal{N}} m_1 \frac{(q_n^d[t])^\alpha}{q_n^d[t]} \mathcal{I}_{\{q_n^d[t] \geq \mu_{max}\}} + |D||\mathcal{N}|m_2.$$

where $\pi_{in(n)}^{*d}(\mathbf{w}[t])$ and $\pi_{out(n)}^{*d}(\mathbf{w}[t])$ are defined similarly to $\pi_{in(n)}^d$ and $\pi_{out(n)}^d$, respectively.

First, by the Congestion Control algorithm (Section IV-B), we have that:

$$\begin{aligned} &L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - \sum_{f \in \mathcal{F}} (q_{b(f)}^{e(f)}[t])^\alpha x_f[t] \\ &\geq L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) - \sum_{f \in \mathcal{F}} (q_{b(f)}^{e(f)}[t])^\alpha x_f^*(\beta), \end{aligned}$$

and hence, (9) + (10) ≤ 0 .

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

- For all $d \in D$, $n \in \mathcal{N}$, and $n \neq d$,
$$\sum_{f \in F(n,d)} x_f^*(\beta) = \beta \mu_{out(n)}^d - \beta \mu_{in(n)}^d,$$

where $\mu_{in(n)}^d$ and $\mu_{out(n)}^d$ are defined similarly to $\pi_{in(n)}^d$ and $\pi_{out(n)}^d$, respectively;

$$\bullet \left\{ \sum_{d \in D} \mu_{nm}^d \right\} = \sum_{i=1}^{|\mathcal{M}|} \alpha_i \pi^i, \sum_{i=1}^{|\mathcal{M}|} \alpha_i < 1, \alpha_i \geq 0, \forall i.$$

Therefore, we have:

$$\begin{aligned} (11) + (12) &= -\beta \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \left(\pi_{out(n)}^{*d}(\mathbf{w}[t]) - \pi_{in(n)}^{*d}(\mathbf{w}[t]) - \mu_{out(n)}^d + \mu_{in(n)}^d \right) \\ &= -\beta \sum_{d \in D} \sum_{(n,m) \in \mathcal{L}} \left(\pi_{nm}^{*d}(\mathbf{w}[t]) - \mu_{nm}^d \right) \\ &\quad \times c_{nm} \left((q_n^d[t])^\alpha - (q_m^d[t])^\alpha \right) \\ &\leq -\beta \left(1 - \sum_{i=1}^{|\mathcal{M}|} \alpha_i \right) \sum_{(n,m) \in \mathcal{L}} \pi_{nm}^*(\mathbf{w}[t]) \\ &\quad \times c_{nm} \max_d \left((q_n^d[t])^\alpha - (q_m^d[t])^\alpha \right). \end{aligned}$$

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

$$\begin{aligned} &\sum_{(n,m) \in \mathcal{L}} c_{nm} \pi_{nm}^*(\mathbf{w}[t]) \max_d \left((q_n^d[t])^\alpha - (q_m^d[t])^\alpha \right) \\ &\geq \frac{1}{l_{max}} \left[(1 + \alpha) \frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|} \right]^{\frac{\alpha}{\alpha+1}} \min_{(n,m) \in \mathcal{L}} c_{nm}. \end{aligned}$$

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, notice that we can rewrite the expression (13)+(14) as:

$$\begin{aligned} &\sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \left[\pi_{in(n)}^d[t-1] - \pi_{out(n)}^d[t-1] \right] \\ &- \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \beta \left[\pi_{in(n)}^{*d}(\mathbf{w}[t]) - \pi_{out(n)}^{*d}(\mathbf{w}[t]) \right] \\ &= \sum_{d \in D} \sum_{(n,m) \in \mathcal{L}} \left[\beta \pi_{nm}^{*d}(\mathbf{w}[t]) - \pi_{nm}^d[t-1] \right] \\ &\quad \times \left[(q_n^d[t])^\alpha - (q_m^d[t])^\alpha \right] \\ &\leq \sum_{(n,m) \in \mathcal{L}} \left(\beta \pi_{nm}^*(\mathbf{w}[t]) - \pi_{nm}[t-1] \right) w_{nm}[t] \\ &\leq \sqrt{V_2(\mathbf{y}[t])}. \end{aligned}$$

Combining these results together yields:

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

Now, fix a constant $\tilde{\epsilon}$ such that $0 < \tilde{\epsilon} < \epsilon$. It is easy to see that if $q > B_1 := \max \left\{ \mu_{max}, \frac{m_1}{\epsilon - \tilde{\epsilon}} \right\}$, then $\frac{m_1}{q} \mathcal{I}_{\{q \geq \mu_{max}\}} <$

$\epsilon - \tilde{\epsilon}$. Therefore, we have:

$$(15) \leq (\epsilon - \tilde{\epsilon}) \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \mathcal{I}_{\{q_n^d[t] > B_1\}} + \frac{|D||\mathcal{N}|B_1}{\mu_{max}}.$$

Also, note that the Jensen's inequality yields:

$$\begin{aligned} & \frac{1}{|D||\mathcal{N}|} \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^\alpha \\ & \leq \left[\frac{1}{|D||\mathcal{N}|} \sum_{d \in D} \sum_{n \in \mathcal{N}} (q_n^d[t])^{\alpha+1} \right]^{\frac{\alpha}{\alpha+1}} \\ & = \left[\frac{1+\alpha}{|D||\mathcal{N}|} V_1(\mathbf{y}[t]) \right]^{\frac{\alpha}{\alpha+1}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \Delta_1(\mathbf{y}[t]) & \leq -\epsilon|D||\mathcal{N}| \left[(1+\alpha) \frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|} \right]^{\frac{\alpha}{\alpha+1}} \\ & \quad + L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) \\ & \quad + (\epsilon - \tilde{\epsilon})|D||\mathcal{N}| \left[(1+\alpha) \frac{V_1(\mathbf{y}[t])}{|D||\mathcal{N}|} \right]^{\frac{\alpha}{\alpha+1}} \\ & \quad \times \mathcal{I}_{\{q_n^d[t] \geq B_1, \forall d, n\}} \\ & \quad + \sqrt{V_2(\mathbf{y}[t])} + |D||\mathcal{N}| \left(m_2 + \frac{B_1}{\mu_{max}} \right) \\ & \leq -\gamma [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} + \sqrt{V_2(\mathbf{y}[t])} + C_2 \\ & \quad + L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) \end{aligned}$$

where

$$\gamma = \tilde{\epsilon} (|D||\mathcal{N}|)^{\frac{1}{1+\alpha}} (1+\alpha)^{\frac{\alpha}{1+\alpha}},$$

and

$$C_2 = |D||\mathcal{N}| \left(m_2 + \frac{B_1}{\mu_{max}} \right).$$

Proof of Lemma 5:

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

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

Then,

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

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:

$$\begin{aligned} \mathbb{E}[V_2(\mathbf{y}(t+1)|\mathbf{y}(t))] & \leq (1-\delta) \mathbb{E}[\{\mathbf{w}[t](\beta \pi^*(\mathbf{w}[t+1]) - \pi[t]) \\ & \quad + (\mathbf{r}[t] - \mathbf{z}[t])(\beta \pi^*(\mathbf{w}[t+1]) - \pi[t])\}^2 | \mathbf{y}[t], \mathcal{E}_\beta^c] \end{aligned}$$

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

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

Combining these, we can find finite positive constants C_3, C_4 such that

$$\mathbb{E}[V_2(\mathbf{y}(t+1)|\mathbf{y}(t))] \leq (1-\delta)V_2(\mathbf{y}[t]) + C_3\sqrt{V_2(\mathbf{y}[t])} + C_4,$$

which completes the proof. $\blacksquare$

We now proceed to use this Lemmas to prove our main result.

Proof of Theorem 2:

Combining those two lemmas, we have:

$$\begin{aligned} \Delta(\mathbf{y}[t]) & \triangleq \mathbb{E}[V(\mathbf{y}[t+1]) - V(\mathbf{y}[t]) | \mathbf{y}[t]] \\ & \leq -\gamma [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) \\ & \quad + L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) \\ & \quad + (1+C_3)\sqrt{V_2(\mathbf{y}[t])} + C_2 + C_4. \end{aligned} \quad (16)$$

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

$$\begin{aligned} \Delta(\mathbf{y}[t]) & \leq -\gamma [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) + L|\mathcal{F}|U_{max} \\ & \quad + (1+C_3)\sqrt{V_2(\mathbf{y}[t])} + C_2 + C_4. \end{aligned}$$

Let us consider the set $\{\mathbf{y} : V(\mathbf{y}) < B\}$ 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 B large enough.

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

$$\begin{aligned} \Delta(\mathbf{y}[t]) & \leq -\frac{\gamma}{2} [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \frac{\gamma}{2} [(B - V_2(\mathbf{y}[t]))^+]^{\frac{\alpha}{\alpha+1}} \\ & \quad + (1+C_3)\sqrt{V_2(\mathbf{y}[t])} - \delta V_2(\mathbf{y}[t]) + C_5, \end{aligned}$$

where $C_5 = C_2 + C_4 + L|\mathcal{F}|U_{max}$. Now, for B large enough, we can get the bound:

$$\begin{aligned} 0 & \geq -\frac{\gamma}{2} [(B - V_2(\mathbf{y}[t]))^+]^{\frac{\alpha}{\alpha+1}} + (1+c_2)\sqrt{V_2(\mathbf{y}[t])} \\ & \quad - \delta V_2(\mathbf{y}[t]) + c_4. \end{aligned}$$

Therefore,

$$\Delta(\mathbf{y}[t]) \leq -\frac{\gamma}{2} [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} \quad \text{if } V(\mathbf{y}[t]) \geq B$$

Given the above inequality, 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 C_1 such that

$$\begin{aligned} C_1 & \geq -\gamma [V_1(\mathbf{y}[t])]^{\frac{\alpha}{\alpha+1}} - \delta V_2(\mathbf{y}[t]) \\ & \quad + (1+C_3)\sqrt{V_2(\mathbf{y}[t])} + C_2 + C_4. \end{aligned}$$

Then, the inequality (16) becomes:

$$\Delta(\mathbf{y}[t]) \leq C_1 + L \sum_{f \in \mathcal{F}} U_f(x_f[t]) - L \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta))$$

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

$$\begin{aligned} \mathbb{E}[V(\mathbf{y}(T)) - V(\mathbf{y}(0))] &\leq L \sum_{t=0}^{T-1} \sum_{f \in \mathcal{F}} \mathbb{E}[U_f(x_f[t])] \\ &\quad - LT \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) + Tc_5 \\ \frac{1}{T} \sum_{t=0}^{T-1} \sum_{f \in \mathcal{F}} \mathbb{E}[U_f(x_f[t])] &\geq \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) \\ &\quad - \frac{V(\mathbf{y}(0))}{LT} - \frac{C_1}{L}. \end{aligned}$$

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

$$\sum_{f \in \mathcal{F}} U_f(\bar{x}_f) \geq \sum_{f \in \mathcal{F}} U_f(x_f^*(\beta)) - \frac{C_1}{L}.$$

■

REFERENCES

- [1] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Proceedings of IEEE Conference on Decision and Control*, San Diego, CA, December 2006, pp. 1258–1263.
- [2] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proceedings of 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 *Proceedings of IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 19–27.
- [4] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910–917, 1988.
- [5] 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*, vol. 37, no. 12, pp. 1936–1948, December 1992.
- [6] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE INFOCOM*, vol. 1, San Francisco, CA, March 1996, pp. 296–302.
- [7] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of IEEE INFOCOM*, vol. 2, San Francisco, CA, March 1998, pp. 533–539.
- [8] P. Giaccone, B. Prabhakar, and D. Shah, "Towards simple, high-performance schedulers for high-aggregate bandwidth switches," in *Proceedings of IEEE INFOCOM*, vol. 3, New York, NY, June 2002, pp. 1160–1169.
- [9] T. Weller and B. Hajek, "Scheduling non uniform traffic in a packet switching system with small propagation delay," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 813–823, December 1997.
- [10] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proceedings of IEEE INFOCOM*, vol. 2, Tel Aviv, Israel, March 2000, pp. 556–564.
- [11] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks," in *Proceedings of IEEE INFOCOM*, vol. 3, Miami, FL, March 2005, pp. 1804–1814.
- [12] S. Sarkar and K. Kar, "Achieving 2/3 throughput approximation with sequential maximal scheduling under primary interference constraints," in *Proceedings of Allerton Conference on Communication, Control and Computing*, Monticello, IL, September 2006.
- [13] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 27–38, June 2006.
- [14] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks: A partitioning approach," in *Proceedings of ACM MOBICOM*, Los Angeles, CA, September 2006, pp. 26–37.
- [15] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial complexity algorithms for full utilization of multi-hop wireless networks," in *Proceedings of 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 *Proceedings of WiMesh*, 2006.
- [17] 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 the IEEE Conference on Decision and Control*, December 2005, pp. 5342–5347.
- [18] P. Chaporkar, K. Kar, and S. Sarkar, "Achieving queue length stability through maximal scheduling in wireless networks," in *Proceedings of Information Theory and Applications Workshop*, UCSD, February 2006.
- [19] X. Wu, R. Srikant, and J. Perkins, "Queue length stability of maximal greedy schedules in wireless networks," in *Proceedings of Information Theory and Applications Workshop*, UCSD, February 2006.
- [20] S. Ray and S. Sarkar, "Arbitrary throughput versus complexity trade-offs in wireless networks using graph partitioning," in *Proceedings of Information Theory and Applications Workshop*, UCSD, January 2007.
- [21] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, January 2005.
- [22] D. Shah and D. Wischik, "Optimal scheduling algorithms for input-queued switches," in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.
- [23] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *Proceedings of ACM-SIAM Symposium on 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 Systems*, vol. 50, no. 4, pp. 401–457, August 2005.
- [25] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proceedings of IEEE INFOCOM*, Miami, FL, March 2005, pp. 1794–1803.
- [26] —, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, August 2006.
- [27] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of IEEE INFOCOM*, Miami, FL, March 2005, pp. 1723–1734.
- [28] 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, pp. 1484–1489.
- [29] S. Pettie and P. Sanders, "A simpler linear time 2/3-ε approximation for maximum weight matching," *Information Processing Letter*, vol. 91, no. 6, pp. 271–276, 2004.

PLACE
PHOTO
HERE

optimization.

PLACE
PHOTO
HERE

Loc X. Bui received the B.Tech. degree in Telecommunications and Electronics Engineering from the Posts and Telecommunications Institute of Technology, Ho Chi Minh City, in 2003. He is currently with the Department of Electrical and Computer Engineering and the Coordinated Science Lab, University of Illinois at Urbana-Champaign, where he is a PhD candidate. He was awarded the Vodafone Fellowship at UIUC in 2006 and 2007. His research interests include communication networks, wireless communications, stochastic processes, network control and

Sujay Sanghavi has a B. Tech in EE from IIT Bombay, and an MS in ECE, MS in Mathematics, and a PhD in ECE all from UIUC. In his graduate research, he worked on the design and analysis of communication network algorithms. After graduating in 2006, Sujay has been a postdoc in MIT, where he works on large-scale signal processing problems.

Sujay's primary research interests are in probability and optimization, as applied to the design of algorithms in communications and signal processing. He has been awarded the Mavis Fellowship and the Perry Fellowship at UIUC.

PLACE
PHOTO
HERE

R. Srikant (S'90-M'91-SM'01-F'06) received his B.Tech. from the Indian Institute of Technology, Madras in 1985, his M.S. and Ph.D. from the University of Illinois in 1988 and 1991, respectively, all in Electrical Engineering. 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 Lab.

He was an associate editor of *Automatica* and the *IEEE Transactions on Automatic Control*, and is currently an associate editor of the *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 in Santa Fe, NM and a program co-chair of IEEE INFOCOM, 2007. His research interests include communication networks, stochastic processes, queueing theory, information theory, and game theory.