Distributed Greedy Approximation to Maximum Weighted Independent Set for Scheduling with Fading Channels

ABSTRACT

Developing scheduling mechanisms that can simultaneously achieve throughput optimality and good delay performance often require solving the Maximum Independent Weighted Set (MWIS) problem. However, under most realistic network settings, the MWIS problem can be shown to be NP-hard. In non-fading environments, low-complexity scheduling algorithms have been provided that converge either to the MWIS solution in time or to a solution that achieves at least a provable fraction of the achievable throughput. However, in more practical systems the channel conditions can vary at faster time-scales than convergence occurs in these lower-complexity algorithms. Hence, these algorithms cannot take advantage of the opportunistic gain, and may no longer guarantee good performance. In this paper, we propose a low-complexity scheduling scheme that performs provably well under fading channels and is amenable to implement in a distributed manner. To the best of our knowledge, this is the first scheduling scheme under fading environments that requires only local information, has a low complexity that grows logarithmically with the network size, and achieves provable performance guarantees (which is arbitrarily close to that of the well-known centralized Greedy Maximal Scheduler). Through simulations we verify that both the throughput and the delay under our proposed distributed scheduling scheme are close to that of the optimal solution to MWIS. Further, we implement a preliminary version of our algorithm in a testbed by modifying the existing IEEE 802.11 DCF. The preliminary experiment results show that our implementation successfully accounts for wireless fading, and attains the opportunistic gains in practice, and hence substantially outperforms IEEE 802.11 DCF.

1. INTRODUCTION

Scheduling is one of the most fundamental functionalities of wireless networks. It determines which links should transmit at what time and at what data rate. It is well-known that solving the scheduling problem is inherently difficult because the interference relationship is often non-convex and even combinatorial in nature. Further, for large networks it is imperative that the scheduling algorithm is of low complexity and can be implemented in a fully distributed manner. Such requirements make it highly challenging to design easy-to-implement scheduling algorithms.

In the literature, it is well-known that the so-called MaxWeight algorithm is throughput optimal [1]. For graph based interference models, where whether two links interfere or not can be specified by a binary parameter, the MaxWeight algorithm corresponds to the solution to a Maximum Weighted Independent Set (MWIS) problem in the conflict (or interference) graph as follows. In the conflict graph, each link is mapped onto a vertex and two vertices (links) that interfere with each other are connected by an edge. A set of non-connected vertices, which is called an independent set, can transmit data simultaneously. Further, each vertex of the conflict graph is given a weight, which is typically the product of the link rate and its queue length, and which varies across time due to changing queue lengths and time-varying channels. The MaxWeight algorithm then computes an independent set that has the largest total weight (i.e., solution to the MWIS problem). Although the MaxWeight algorithm is throughput optimal, the MWIS problem is NP-Hard in general [2]. Hence, the MaxWeight algorithm incurs high complexity, and further, it is a centralized algorithm that requires global information. Thus, the MaxWeight algorithm is not amenable to practical implementation.

In the literature, there have been many efforts to develop low-complexity and distributed scheduling algorithms with provably good throughput performance [3–12]. These algorithms differ in terms of their throughput guarantee, complexity, and delay performance. They can be classified into two categories, depending on whether or not they account for channel fading.

There have been many more scheduling solutions for wireless systems *without fading*. Low-complexity scheduling algorithms have been developed with complexity that grows significantly slower than the network size, and can yet guarantee a non-negligible fraction of the optimal system capacity. As a point of comparison, the *Greedy Maximal Scheduling* (GMS) algorithm (also known as *Longest Queue First* (LQF) algorithm) can provably attain a fraction of the optimal capacity, with complexity that grows linearly with the total number of links L [13]. Other algorithms can reduce the complexity even further. For example, the *Maximal Scheduling* algorithm can attain at least $\frac{1}{\Delta}$ of the optimal capacity, with $O(\log N)$ complexity [14], where Δ denotes the maximum conflict degree (see (2) for the definition) and

N denotes the number of nodes. The *Constant-time* scheduling algorithms, instead, can achieve a comparable capacity with $O(1)$ complexity, i.e., the complexity does not grow with the network size [10]. Further, both the *Pick-and-Compare* algorithm [3,4] and *Carrier Sensing Multiple Access (CSMA)* algorithm [5,6] have been shown to achieve the optimal throughput. They incur $O(L)$ and $O(1)$ complexity, respectively. We note that these two algorithms have been observed to lead to poor delay performance [7,8], and hence the utility of the throughput gain may be debatable, especially for delay-sensitive applications. Nonetheless, these results indicate that good throughput performance may be attained for non-fading environments using algorithms with very low complexity.

In practice, however, most wireless systems experience some level of channel fading. When link rates vary across time due to fading, the system throughput can be further improved by scheduling links when their rate are high. This is known as the opportunistic gain [15]. Exploiting opportunistic gain has been extremely popular in cellular systems. For ad hoc wireless networks, the MaxWeight algorithm can exploit this opportunistic gain and in fact achieve the optimal throughput even with fading. However, many of the low-complexity scheduling algorithms described in the previous paragraph cannot exploit the opportunistic gain, and their performance in fading environments will be much worse.

Take *CSMA* and *Pick-and-Compare* algorithms as examples. They reduce complexity by amortizing the computation across many time-slots, and hence need to take many iterations to find a close-to-optimal schedule. In fading environments, the link rates could have changed significantly before these algorithms can find a good schedule. Hence, they will not be able to exploit the opportunistic gain unless the fading is very slow [16]. Similarly, it appears to be difficult for the *Maximal Scheduling* algorithm and the *Constant-time* scheduling algorithm to account for channel fading and still guarantee a provable fraction of the optimal capacity. Recently, there have been a few other low-complexity schemes that are provably efficient with fading channels. However, they are either limited to single-hop networks [12] or their performance guarantees are much lower [11].

An exception is perhaps the GMS scheduling algorithm, which computes an approximation to the MWIS problem by choosing the highest weight vertex first, and can guarantee $\frac{1}{\Delta}$ fraction of the optimal capacity *in both fading and non-fading environment.* Other greedy approximations have also been proposed in [17,18]. However, they require centralized operations and linear complexity $O(L)$. Although distributed greedy approximation algorithms have been developed [19,20], they still incur a worst case time-complexity of $O(L)$. This high complexity has become a major obstacle preventing these algorithms from being used in practical system because the channel conditions can vary at faster time-scales than $O(L)$. Given that fading is a prevalent phenomenon in most modern wireless systems, an interesting open question is how one can develop *distributed* scheduling algorithms with even *lower complexity* and yet *guarantee good performance* .

In this paper, we answer this open question by proposing a novel low-complexity and distributed greedy approximation algorithm, called DistGreedy, for both fading and non-fading environments. In contrast to the known greedy approximations [17–20], our proposed DistGreedy algorithm incurs a low logarithmic complexity $O(\log L)$ that grows slowly with the network size. Further, it requires only local information (such as queue length and link rates of neighboring links), and can be implemented in a distributed fashion. We analytically show that our low-complexity distributed algorithm produces a schedule that is a $\frac{1}{\Delta}$-approximation to the MWIS problem, and show through simulations that DistGreedy often achieves scheduling performance far better than the provable bounds. Indeed, it empirically achieves throughput and delay performance that is close to that of the MaxWeight scheduler. We also conduct preliminary experiments with implementation in a real testbed. We implement a new MAC protocol that captures the essence of DistGreedy by modifying the IEEE 802.11 DCF. Performance comparison with the IEEE 802.11 DCF under channel fading shows that the DistGreedy algorithm can exploit the opportunistic gains and thus substantially outperform IEEE 802.11 DCF in fading environments.

The rest of the paper is organized as follows. The system model is described in Section 2. The DistGreedy algorithm is proposed and analyzed in Section 3. We numerically evaluate its performance in Section 4, and provide preliminary experiment results based on a testbed implementation in Section 5. Then, we conclude.

2. SYSTEM MODEL

We consider a wireless network with N nodes and L directed links. We assume that time is slotted and that a single frequency channel is shared by all the links. Multiple link transmissions at the same time slot may fail due to wireless interference. We assume that there is no link error, i.e., a link transmission is successful if there is no simultaneous interfering transmission.

The link rate of a successful transmission depends on its channel state. We assume that the channel state is fixed during a time slot, and changes across time slots. We denote the (global) channel state by h. when the channel is in state h, link l can transfer r_l^h unit of data if its transmission is successful. Let $\mathcal{H}$ denote the set of all the channel states. We assume that the channel state has a finite space with a stationary distribution π^h, with $\sum_{h\in\mathcal{H}} \pi^h = 1$.

In order to account for wireless fading, we employ a channel-dependent interference model as follows. Let $C^h_{kl} \in \{0,1\}$ denote the interference relationship between link k and link l when the channel state is h. We set $C^h_{kl} = 0$ if link l does not interfere with link k (and therefore they can transmit simultaneously), and $C^h_{kl} = 1$, otherwise. We assume that the interference relationship is symmetric, i.e., $C^h_{kl} = C^h_{lk}$. We note that the dependency on h represents a major departure from existing works for non-fading environments. Specifically, the interference relationship as well as the link rate may change across time in our model. This model is not only a simplified version that captures the fundamental characteristics of the more accurate SINR interference model [21], but also a general model that includes many interference models used in the literature to model FH-CDMA, Bluetooth, and IEEE 802.11 DCF network systems [14, 22], as special cases.

Given a network system, our interference model admits a unique *conflict graph* at each channel state h, which clearly presents the underlying interference constraints. For each link $l \in L$, we draw a vertex in the conflict graph, which is also denoted by the same alphabet l. For every two vertices k, l with $C^h_{kl} = 1$, we connect them with an edge in the conflict graph. Let $G^h = (V, E^h)$ denote the conflict graph with the set V of vertices and the set E^h of edges under channel state h. The conflict graph explicitly shows the interference relationship of any two vertices (i.e., links in the original network). In the sequel, we deal with the conflict graph throughout the paper.

We now formally formulate the Maximum Weighted Independent Set (MWIS) problem. Suppose that the channel state is h at time slot t. We consider the conflict graph G^h constructed from the interference constraints under channel state h. We begin with some definitions. Vertex x is a *neighbor* of vertex v, if they are connected by an edge in the conflict graph. Let $I^h(v)$ denote the set of neighbors of vertex v including v, and let $I^h(A)$ denote the set of neighbors of vertices in A, i.e., $I^h(A) := \cup_{v\in A} I^h(v)$. Let $w_v(t,h)$ denote a weight associated with vertex v. In particular, we define the weight of vertex v as the product of queue length $Q_v(t)$ and transmission rate r^h_v. Let $w^*(t,h)$ denote the largest weight, i.e., $w^*(t,h) := \max_{v\in V} w_v(t,h)$. Further, let $\bar{w}_v(X;t,h)$ denote the largest weight in the neighborhood of vertex v within X, i.e., $\bar{w}_v(X;t,h) := \max_{x\in X\cap I^h(v)} w_x(t,h)$.

We say that a set S of vertices is an *independent set* (or a *feasible schedule*) if no two vertices in the set are neighbors. Further, an independent set is maximal if no extra vertex can be added. Such an independent set is also called a *maximal matching*. Let $\mathbb{S}^h$ denote the collection of all the feasible independent sets that are available in G^h. The MWIS problem can be formulated as finding S^* such that

$$S^* \in \operatorname*{argmax}_{S\in\mathbb{S}^h} \sum_{v\in S} w_v(t,h). \tag{1}$$

It has been known that at each time t, given a channel state h, the solution to the MWIS problem with weight $w_v(t,h) = Q_v(t) \cdot r^h_v$ results in a throughput-optimal MaxWeight scheduling scheme [23]. However, due to the high computational complexity and the requirement of global information, such a MaxWeight algorithm is difficult to implement in practice. On the other hand, it has been shown in [24, 25] that an imperfect scheduling solution that solves (1) within a factor of γ at every time t achieves at least γ fraction of the optimal throughput. To this end, our goal is to develop practical low-complexity scheduling algorithm that can approximately solve (1) with a provable fraction in a distributed fashion.

Remarks: In the above MaxWeight scheduling scheme, we implicitly assume single-hop traffic, i.e., packets are transmitted over a single link and leave the system immediately after the transmission. For multi-hop traffic, the same MaxWeight algorithm can be used by replacing the queue length with a queue differential. (See [1] for the details.) Similarly, our DistGreedy algorithm described in the next section can be extended to multi-hop scenarios in a straightforward manner.

Finally, we define the vertex degree $\delta(h) := \max_{v\in V} |I^h(v)|$, where $|\cdot|$ denotes the cardinality of the set, and the maximum conflict (or interference) degree $\Delta(h)$ as

$$\Delta(h) := \max_{v\in V, S\in\mathbb{S}} |I^h(v) \cap S|. \tag{2}$$

In the network, the maximum conflict degree represents the maximum number of simultaneous transmissions in the neighborhood of any link, which can be upper bounded by a constant in many practical interference models [21, 26]. Also, we define $\Delta^* = \max_{h\in\mathcal{H}} \Delta(h)$.

3. DISTRIBUTED GREEDY APPROXIMATION

In this section, we describe our distributed approximate solution to (1) and analyze its performance. We emphasize that the algorithm operates in a distributed manner and each vertex (link) requires only local information from its neighbors in the conflict graph. Throughout this section, we consider the conflict graph G^h at time t under channel state h, and omit the subscripts t and h if there is no confusion.

3.1 Algorithm description

We assume that each time slot has two parts: contention and transmission. The contention part has several intervals, and each interval is further divided into

Algorithm 1 DistGreedy algorithm.

```
    $V_0 \leftarrow V$, $B_0 \leftarrow \emptyset$
 1: for $i = 1$ to $\log_\alpha \beta|V|$ do
 2:   $V_i \leftarrow V_{i-1} \backslash I(B_{i-1})$
 3:   $A_i \leftarrow \emptyset$
 4:   for each $v \in V_i$ do
 5:     calculate $\bar{w}_v(V_i) := \max_{x \in V_i \cap I(v)} w_x$
 6:     if $w_v \geq \frac{\bar{w}_v(V_i)}{\alpha}$ then
 7:       $A_i \leftarrow A_i \cup \{v\}$
 8:     end if
 9:   end for
10:   $B_i \leftarrow$ dist_maximal_matching$(A_i)$
11: end for
```

mini-slots. We determine a feasible schedule during the contention part, and with the computed schedule, transmits actual data during the transmission part.

At a time slot, let B denote the feasible schedule (independent set of vertices) chosen by our algorithm. We explain our solution, starting with an empty set and add vertices to B by executing an iterative algorithm as shown in Algorithm 1.

At each interval i, some vertices are 'determined' as to whether they belong to set B or not. Specifically, vertices in B_i are 'determined' to be in B at interval i, and vertices in $(I(B_i)\backslash B_i)$ are 'determined' not to be in B at interval i. Let V_i denote the set of vertices that have not been determined yet at the beginning of interval i, i.e.,

$$V_i := V \backslash \left(\cup_{j=1}^{i-1} I(B_j) \right),$$

which can be rewritten in a recursive form as

$$V_i = V_{i-1} \backslash I(B_{i-1}).$$

We say that a vertex in V_i is *eligible* at interval i. Let $A_i \subset V_i$ denote the set of vertices that will be 'determined' during interval i, from which we will compute B_i. We will soon see how to find A_i and B_i. From the definitions, it is clear that $V_0 = V$ and $B_0 = \emptyset$. Finally, we have a couple of configuration parameters α, β that will be explained later.

The algorithm can be described as follows. We start with the entire set of vertices $V_0 = V$ and an empty set $B_0 = \emptyset$. Suppose that each vertex v knows its neighbors' weights. In wireless networks, this can be obtained by piggybacking/overhearing the information exchange or by explicitly exchanging control messages.

1. At each interval i, the set of eligible vertices V_i is updated by excluding $I(B_{i-1})$ from V_{i-1} (line 2 in Algorithm 1), where B_{i-1} denotes the set of vertices that are chosen during interval $i-1$ and $I(B_{i-1})$ denotes the set of neighbors for B_{i-1} (including B_{i-1} itself). For this purpose, each vertex that belongs to B_{i-1} should notify its neighbors by broadcasting a control message during interval $i-1$. (See Step 4 below.)

2. Each vertex v in V_i calculates its local maximum weight $\bar{w}_v(V_i) := \max_{x \in V_i \cap I(v)} w_x$ from the weight information of its neighbors (in V_i). Then each vertex v sets itself as one of A_i if $w_v \geq \bar{w}_v(V_i)/\alpha$, where $\alpha > 1$. (Lines $5-8$.)

3. On the set A_i, we compute a maximal matching in a distributed manner (line 10), which requires $O(\delta \log^2 |V|)$ complexity [27] or $O(\delta)$ complexity with precomputation [9], where δ is the vertex degree. Let B_i denotes the obtained set.

4. In the process of computing the maximal matching, each neighbor of vertex $v \in B_i$ should be informed that v belongs to B_i. Hence, the vertices in $I(B_i)$ will not participate in the next interval.

5. The above procedure repeats for J times, where $J := \lceil \log_\alpha \beta|V| \rceil$. The set $B (= \cup_i B_i)$ of vertices will be returned as the final result. This is the set of links that will transmit data packets during the time slot.

We have two configuration parameters α and β that will be further discussed in the next section.

Note that our distributed greedy (DistGreedy) algorithm requires $O(\delta)$ complexity at each interval and will run for $O(\log|V|)$ intervals (using the algorithm in [9]). Hence, the worst-case complexity will be $O(\delta \log|V|)$. In some applications, e.g., regular topologies, δ is a fixed constant. Thus, DistGreedy can be implemented with $O(\log|V|)$ complexity (or with polylogarithmic complexity in random networks[1]), which is much lower than the $O(|V|)$ complexity of the known distributed implementation of GMS [19, 20].

Remarks: Note that the previous greedy approximations to the MWIS problem shown in [17] have a similar iterative procedure as DistGreedy. However, their algorithm works vertex-by-vertex sequentially, which results in linear complexity in the worst case (e.g., consider a ring topology such that, starting from a link, the link weights decrease in a clockwise direction). Further, they have a different rule for selecting a vertex at each interval, e.g., they select vertex v with the largest $\frac{w_v}{|V_i \cap I(v)|}$ or with $w_v \geq \sum_{x \in V_i \cap I(v)} \frac{w_x}{|V_i \cap I(x)|}$. This selection rule is the key to achieve the provable approximation ratio of $\frac{1}{\delta}$. Unlike this previous work [17], DistGreedy reduces the complexity significantly by considering multiple vertices in parallel and do not follow the strict sequential ordering. Further, the procedure stops after a certain number of intervals. At each interval, DistGreedy selects the vertices v with $w_v \geq \max_{x \in V_i \cap I(v)} w_x/\alpha$. The

[1]In random networks, it is well-known that $\delta \sim O(\log|V|)$ to ensure connectivity [21].

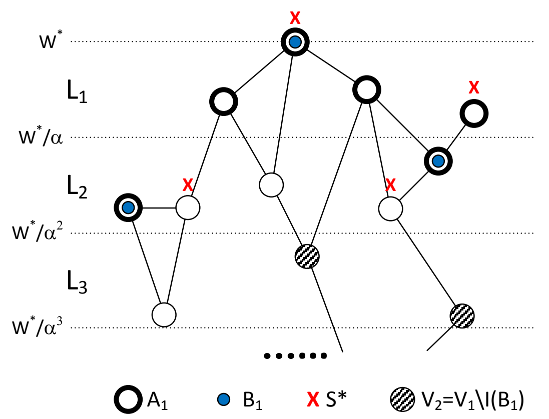


Figure 1: Conflict graph with vertices and edges in the layered format, where vertices are partitioned into layers according to their weights.

end result is a much better approximation ratio ($\approx \frac{1}{\Delta}$) and a much better complexity ($O(\log|V|)$). However, the parallel processing also makes it more difficult to analyze the performance of DistGreedy. Nonetheless, in the next section, we show that due to the selection rule of DistGreedy, it can achieve the approximation ratio arbitrarily close to $\frac{1}{\Delta}$.

3.2 Performance Analysis

We evaluate the performance of our distributed greedy (DistGreedy) algorithm, and show that it is in fact a $\frac{1}{\Delta}$-approximation to the optimal solution. Motivated by [18], we divide the vertices into layers $L_1, L_2, \ldots$ based on the ratio of their weight to the maximum weight w^*, as [2]

$$L_i = \left\{ v \in V \;\Big|\; \frac{w^*}{\alpha^i} < w_v \le \frac{w^*}{\alpha^{i-1}} \right\}. \tag{3}$$

Fig. 1 illustrates an example conflict graph in the layered format.

We start our analysis with the following lemmas.

Lemma 1. *For $i \le \log_\alpha \beta|V|$, if vertex $v \in L_i$, then $v \in I(\cup_{j=1}^{i} B_j)$, and thus*

$$L_i \subset I(\cup_{j=1}^{i} B_j). \tag{4}$$

Proof. If each vertex $v \in L_i$ selects itself for distributed maximal matching no later than the i-th interval, i.e., if $v \in L_i$ implies $v \in \cup_{j=1}^{i} A_j$, then we can obtain the lemma, since

$$v \in L_i \Rightarrow v \in \cup_{j=1}^{i} A_j \Rightarrow v \in \cup_{j=1}^{i} I(B_j) \Rightarrow v \in I(\cup_{j=1}^{i} B_j), \tag{5}$$

where the second step comes from the fact $A_j \subset I(B_j)$, since B_j is a maximal matching on A_j.

Now what remains to be shown is that $v \in L_i$ implies $v \in \cup_{j=1}^{i} A_j$. We show this by induction. It is clear that when $i = 1$, all vertices $v \in L_1$ belong to A_1, because $w_v > \frac{w^*}{\alpha}$. Suppose that the statement is true for all $i \le c$. Note that all vertices in $\cup_{j=1}^{c} A_j$ are not eligible at interval $c+1$ since each vertex in A_j belongs to $I(B_j)$ for $j = 1, 2, \ldots, c$ under our algorithm. Hence, at interval $c+1$, no vertex in $\cup_{j=1}^{c} A_j$ is eligible, which immediately implies that no vertex in $\cup_{j=1}^{c} L_j$ is eligible since $L_i \subset \cup_{j=1}^{i} A_j$ for all $i \le c$. Now, if there is a vertex $v \in L_{c+1}$ eligible at interval $c+1$, i.e., $v \in V_{c+1}$, then vertex v should be included in A_{c+1}, since all vertices e with $w_e > \frac{w^*}{\alpha^c}$ (i.e., $e \in \cup_{j=1}^{c} L_j$) are not eligible. Hence, the induction hypothesis must hold for $i = c+1$. This completes the proof. $\square$

Lemma 1 states that under DistGreedy, any vertex in layer L_i will be 'determined' after interval i ends. In the following lemma, we show that each vertex in layer i must have a neighboring vertex that has a similar or higher weight and that is chosen by DistGreedy after interval i ends. Combining these two lemmas, we can show that after interval i, every vertex in L_i or above is a neighbor of a vertex that is already chosen by DistGreedy.

Lemma 2. *For each vertex $x \in L_i$ (with $i \le \log_\alpha \beta|V|$), there exists $v \in \cup_{j=1}^{i} B_j$ such that $x \in I(v)$ and $\alpha \cdot w_v \ge w_x$.*

Proof. From Lemma 1, we have $x \in I(\cup_{j=1}^{i} B_j)$, and thus there exists $v \in \cup_{j=1}^{i} B_j$ such that $x \in I(v)$. Let $k \le i$ be the smallest index such that $x \in I(B_k)$. Then, $x \notin I(\cup_{j=1}^{k-1} B_j)$ and there exists $v \in B_k$ with $x \in I(v)$. Since $v \in B_k \subset A_k$, it should satisfy $\bar{w}_v(V_k)/w_v \le \alpha$ from line 6 of Algorithm 1. Also since $x \in I(v)$ and x is eligible at interval k (because $x \notin I(\cup_{j=1}^{k-1} B_j)$), we have $\bar{w}_v(V_k) \ge w_x$. Hence, we obtain that $\alpha \cdot w_v \ge w_x$. $\square$

Recall that S^* denotes the maximum weighted independent set over V. We define $D_i(v)$ as the set of vertices in $S^* \cap L_i$ that are connected to v by an edge in the conflict graph, i.e.,

$$D_i(v) := \{x \mid x \in S^* \cap L_i, \text{ and } x \in I(v)\}. \tag{6}$$

Then $|D_i(v)|$ denotes the number of vertices selected by the MWIS solution in layer L_i that conflicts with v. The following lemma shows that the weight sum for S^* within layer L_i can be bounded by the weight sum for the independent set chosen by DistGreedy up to interval i, multiplied by $\alpha \cdot |D_i(v)|$.

[2]The algorithm in [18] computes a maximal matching for each layer, and thus requires for each node to know which layer it belongs to, or equivalently to know w^*. However, knowing the maximum weight w^* may take $O(|V|)$ time to propagate in the worst case. In contrast, DistGreedy works with local weight information and the layering structure is only for the purpose of analysis.

Lemma 3. *At each interval i, we have*

$$\sum_{v \in \cup_{j=1}^{i} B_j} \alpha \cdot |D_i(v)| \cdot w_v \geq \sum_{x \in L_i \cap S^*} w_x. \tag{7}$$

Proof. From Lemma 2, we have that for each vertex $x \in S^* \cap L_i$, there exists $v \in \cup_{j=1}^{i} B_j$ such that $x \in I(v)$ and $\alpha \cdot w_v \geq w_x$. However, multiple x may map to the same v. Nonetheless, for each of such v, at most $|D_i(v)|$ vertices in $S^* \cap L_i$ can potentially be neighbor of v in the conflict graph. Therefore, we can obtain the result. □

By summing both sides of (7) for all i, we can bound the maximum weight sum by the weight sum of the vertices chosen by DistGreedy within a constant factor $\alpha\Delta$. (See the proof of Lemma 4 below.) However, if we were to terminate after all vertices are considered, it would have resulted in $O(|V|)$ complexity (e.g., consider a fully connected graph with vertices whose weights are $1, \frac{1}{\alpha+\epsilon}, \frac{1}{(\alpha+\epsilon)^2}, \ldots$). In the next lemma, we show that even if DistGreedy stops after $O(\log |V|)$ intervals, the performance loss would still be negligible.

Lemma 4. *By setting $\alpha \to 1$ and β sufficiently large, Algorithm 1 is a $\frac{1}{\Delta(h)}$-approximation algorithm.*

Proof. Let $B := \cup_{j=1}^{J} B_j$, where $J := \lceil \log_\alpha \beta |V| \rceil$. By summing (7) from $i = 1$ to J, we can obtain that

$$\begin{aligned}\sum_{i=1}^{J} \sum_{x \in L_i \cap S^*} w_x &\leq \sum_{i=1}^{J} \sum_{v \in \cup_{j=1}^{i} B_j} \alpha \cdot |D_i(v)| \cdot w_v \\ &\leq \sum_{v \in B} \sum_{i=1}^{J} \alpha \cdot |D_i(v)| \cdot w_v \\ &\leq \sum_{v \in B} \alpha \cdot \Delta \cdot w_v.\end{aligned} \tag{8}$$

Also, for $i > J$, we can obtain that

$$\sum_{i=J+1}^{\infty} \sum_{x \in L_i \cap S^*} w_x \leq \sum_{i=J+1}^{\infty} \sum_{x \in L_i} w_x \leq |V| \cdot \frac{w^*}{\alpha^J} \leq \frac{w^*}{\beta}, \tag{9}$$

where w^* denotes the largest weight among all the vertices. The last inequality holds since $J = \lceil \log_\alpha \beta |V| \rceil$.

Combining (8) and (9), we can obtain that

$$\alpha\Delta \sum_{v \in B} w_v + \frac{w^*}{\beta} \geq \sum_{x \in S^*} w_x. \tag{10}$$

Thus our result follows. □

It has been shown in non-fading environments that a scheduling solution that is a γ-approximation to the MWIS problem at each time slot can achieve at least γ fraction of the optimal throughput [24, 25]. It is straightforward to extend the result to fading environment: a scheduling solution that is a $\gamma(h)$-approximation to the MWIS problem under channel state h at each time slot can achieve at least $\min_{h \in \mathcal{H}} \gamma(h)$ fraction of the optimal throughput. Combining it with Lemma 4, we can obtain the following Proposition.

Proposition 5. *A scheduling solution that executes DistGreedy at each time slot can achieve $\frac{1}{\Delta^*}$ fraction of the optimal throughput, where $\Delta^* = \max_{h \in \mathcal{H}} \Delta(h)$.*

Remarks: In developing a distributed low-complexity GMS algorithm, one of the main difficulties lies in the requirement of the strict global ordering of selected vertices. For example, if the conflict graph is a linear graph, where the weights of vertices monotonically decrease from the left to the right, then the selection of the right-most vertex v with the smallest weight can be made only after the selection of its left-side neighbor u, which again can be made only after the selection of the left-side neighbor of vertex u due to the linear topology. This implies that the selection of the right-most vertex v needs to be made after $O(|V|)$ time.

Our result implies that the strict global ordering in the GMS algorithm is not required for high performance. A loose ordering would be sufficient, which can result in significant complexity reduction with negligible performance degradation. We highlight that the state-of-the-art “distributed” $\frac{1}{\Delta(h)}$-approximation algorithm requires $O(|V|)$ complexity [19, 20], while our local greedy algorithm significantly lowers the complexity to $O(\delta(h) \log |V|)$.

4. NUMERICAL RESULTS

We evaluate DistGreedy, Greedy, and MaxWeight through simulations, where MaxWeight is the optimal scheduler that solves the MWIS problem at each time slot. We simulate two networks: one with a grid topology and the other with a randomly generated topology.

We first consider a grid topology as shown in Fig. 2. Each link has an average transmission rate of one, two, or three packets per time slot, which are signified in the figures by the number of lines between two nodes, e.g., one line implies one packet per time on average. At each time slot, actual link rate changes and is chosen uniformly at random from the range $[0, 2(\text{Avg. rate})]$. Since DistGreedy approximates the optimal solution to the MWIS problem at each time slot, we focus on the behavior of DistGreedy under static interference models, where the interference relationship does not change across time. In particular, we use one-hop (or primary) interference model, under which two links that share a node cannot transmit at the same time. We impose single-hop traffic of load ρ on every link: at each time slot, each link has a packet arrival with probability ρ. The arrivals are i.i.d. across time slots and links. We set DistGreedy to have $\lceil \log_\alpha \beta |V| \rceil$ intervals at each time slot. We use a link-coloring technique to find a maximal matching, under which $(\delta + 1)$ mini-slots are sufficient [9]. Since $\delta = 6$ in our grid topology, we use 10

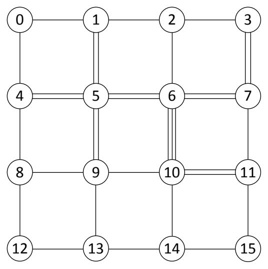


Figure 2: Grid network topology.

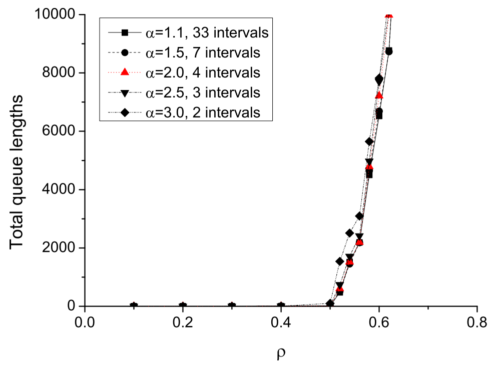


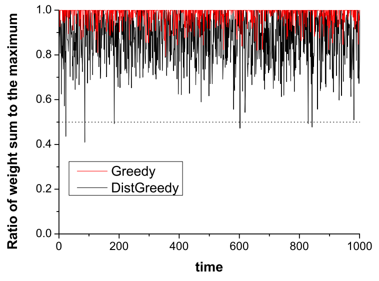
Figure 3: Performance of DistGreedy with different α.

mini-slots at each interval. The number of mini-slots are not taken into account in the performance measurements. Each result is an average of 10 simulation runs for 10^6 time slots.

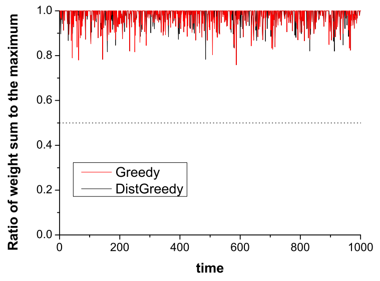
Fig. 3 illustrates the performance of DistGreedy in terms of total queue lengths with different α settings and $\beta = 1$. Sharp increases of queue lengths imply the boundary of the capacity region. Note that a larger α means thicker layers and thus a smaller number of intervals. The results show that the performance is not sensitive to the value of α in a range $[1.1, 3]$ and a small number of intervals (e.g., $\alpha = 2.0$) would be sufficient for high throughput performance.

While running DistGreedy, we also trace the maximum weight sum, the weight sum of the schedule selected by DistGreedy, and the weight sum of the schedule that would be chosen by Greedy, at each time slot. Fig. 4 depicts the ratio of each weight sum (from DistGreedy and Greedy) to the maximum weight sum. It shows that both GMS and DistGreedy typically achieve much higher ratios than the analytical bound, which is $\frac{1}{2}$ in the one-hop interference model and shown in the figure using a dotted line. Also, as α gets closer to 1, DistGreedy algorithm approaches Greedy algorithm because layers become narrower and the number of intervals increases.

In Fig. 5, we compare the performances of MaxWeight, Greedy, DistGreedy (with $\alpha = 2$, $\beta = 1$), and Q-CSMA, where Q-CSMA is a CSMA algorithm known to be throughput-optimal in non-fading environments. For Q-CSMA, we use the version shown in [8], i.e., each link v sets its access probability for the decision vector to $\frac{1}{|I(v)|}$ and sets its weight for link activity to $\log(Q_l(t) \cdot r_l^h(t))$. The results in Fig. 5 illustrate that Q-CSMA, which has the lowest complexity $O(1)$ among the scheduling schemes, has much poor throughput and delay performance than the others. In particular, its delay grows quickly at an offered load much lower than other algorithms, which suggests that it is not throughput optimal in fading environments. In contrast, DistGreedy has similar queue lengths to MaxWeight and Greedy. In other words, it empirically achieves similar throughput and delay performance to the optimal. Further, the simulation results suggest that the actual performance of DistGreedy could significantly outper-



(a) $\alpha = 2.0$



(b) $\alpha = 1.1$

Figure 4: Ratio of the achieved weight sum to the maximum weighted sum.

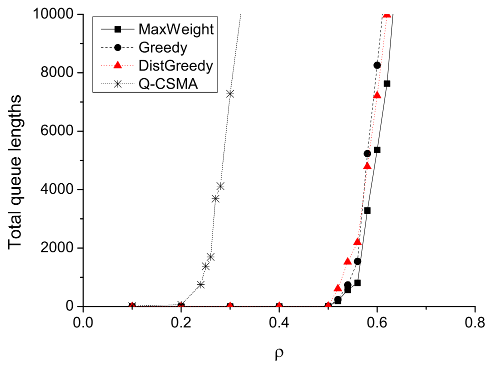


Figure 5: Performance comparison of scheduling schemes.

forms the analytical lower bounds.

Next we consider a network that is randomly generated. We place a total of 32 nodes at random within 1×1 area. We connect two nodes with a link if they are within a distance of 0.25. Each link has a time-varying link rate, which is randomly chosen in the range of $[0, 10]$ packets per slot, and i.i.d. across links and time. We generated single-hop traffic over 24 links, which are chosen at random. Each source injects a random number of packets in the range of $[1, 5]$ at each time slot, with probability ρ. The results are similar to the grid network and are omitted due to lack of space. We can observe that the throughput and delay performance of DistGreedy are close to those of MaxWeight and Greedy, even though it has a significantly lower complexity.

5. PRELIMINARY EXPERIMENT RESULTS

In this section, we provide preliminary experimental results and show that the opportunistic gains of wireless fading can be achieved in practice. We have implemented a version of DistGreedy in hardware driver by modifying the medium access control of IEEE 802.11 DCF. Specifically, we use the `ath5k` device driver [28] over Voyage Linux [29] installed into the Alix 2D2 system board [30]. We modified it such that each node maintains information of recent queue lengths and transmission rates (link capacity) of neighboring nodes, and that a header in the data frame includes the queue length information of the transmitter. The transmission rate information can also be obtained from the device if the frame is successfully received. Each node who receives or overhears the frame updates the queue and rate information of the transmitter from the header.

Unlike IEEE 802.11 DCF, DistGreedy in Algorithm 1 requires time synchronization between nodes. We avoid the synchronization overheads by implementing an approximation of our algorithm while capturing the essential feature of the algorithm. Further, while the DistGreedy scheme is a link-based algorithm, current implementation of IEEE 802.11 DCF operates on nodes. Due to this difference, we use $w'_n(t)$ and $\bar{w}'_n(t)$ to denote node n's weight and its local maximum weight at time t, respectively. Our DistGreedy implementation works just like IEEE 802.11 DCF, except that when the transmitter n has a frame to send (asynchronously with other nodes) at time t, it calculates the local maximum weight $\bar{w}'_n(t)$ from the information that it has received from its neighbors, and estimates $\theta := \lfloor \frac{\bar{w}'_n(t)}{\alpha \cdot w'_n(t)} \rfloor$. Then, it chooses the contention window size at random within $[0, 7]$ if $\theta = 0$, within $[0, 31]$ if $\theta = 1$, within $[0, 63]$ if $\theta = 2$, and within $[0, 127]$ otherwise[3]. In this way, our implementation effectively has four intervals. Moreover, our implementation does not compute a maximal matching, which reduces the complexity further (compared with Algorithm 1, line 10). We set $\alpha = 10/9$, the maximum buffer size to 100 frames, and the IP packet size to 512 bytes. We remark that finding an optimal number of intervals and the contention window sizes is an interesting open question but beyond the scope of the paper.

[3]Non-overlapped windows for each θ seems to be a more intuitive choice. Unfortunately, we are unable to select a window that starts with non-zero value due to configuration restriction in our device firmware.

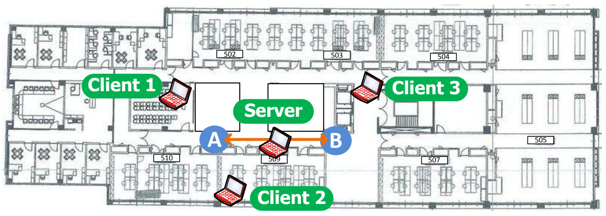
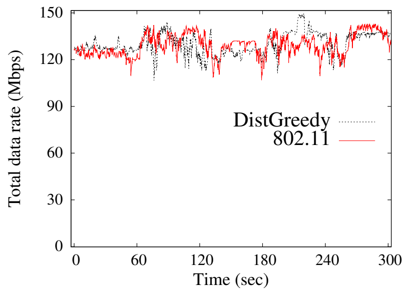


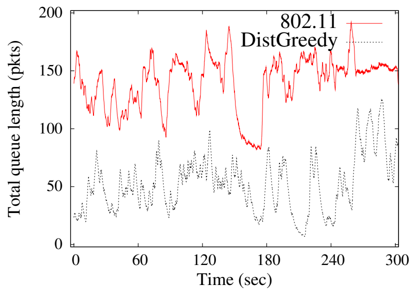
Figure 6: Experiment setup.

Fig. 6 shows our experiment setting at the 5th floor of the ECE department building in UNIST, South Korea. We set three stationary clients and let each client transmit data at 3 Mbps to the single mobile server. All the clients can overhear each other's transmissions. The server moves between two positions A and B as shown in Fig. 6, at every 60 seconds, starting from A. Clearly, Client 1 has a good channel state when the server locates in Position A, and Client 3 performs well when the server locates in Position B. We conduct our experiments for 5 minutes and measure transmission rate, queue lengths, and throughput of each client.

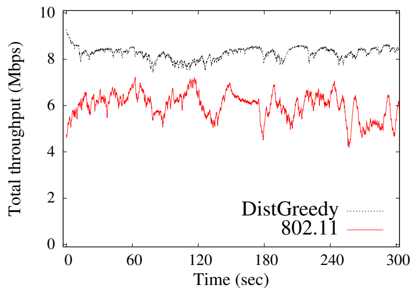
Figs. 7 and 8 illustrate the experiment results. Due to high measurement variations, we show time-averaged values using exponential weighted moving average. Figs. 7(a) and 8(a) show that the variations of the total (average) capacity of three clients is less than the variations of the transmission rate of an individual link, which is significant especially when the server moves. Note that transmission rate of a link has a discrete value of $\{6, 9, 12, 24, 36, 48, 56\}$ Mbps. Instantaneous link rate changes very frequently across time though these changes are not shown in the figures due to averaging. Further the link rate is chosen by the hardware physical layer depending on the channel state at a given time. We have observed that transmission rate changes frequently and often in an unpredictable manner. Since the transmission rate is not under our control, it is difficult to maintain exactly the same channel environment when we compare two different MAC protocols. This is the reason that DistGreedy and IEEE 802.11 DCF have different link rates in Figs. 7(a) and 8(a). Nonetheless, the figures show that overall link rates are similar for both cases. We may force the hardware to use a fixed transmission rate at the transmitter. However, in such a case, we may see frequent transmission failures due to insufficient SINR at the receiver.



(a) Total transmission rate

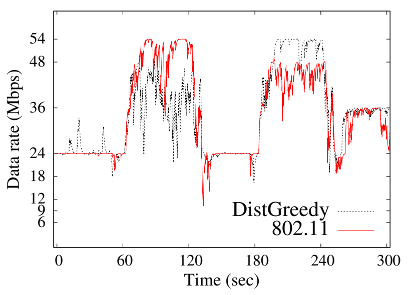


(b) Total queue length

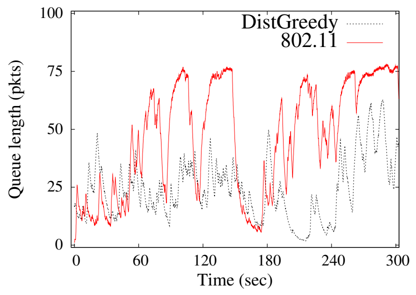


(c) Total throughput

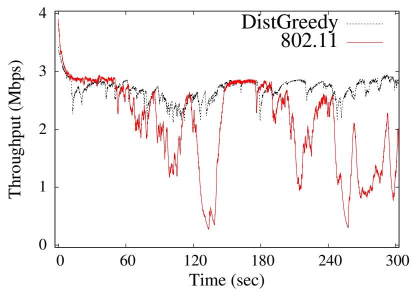
Figure 7: Overall performance across all clients. Total transmission rate (a) shows that the total (average) link rate sum is similar. However, total queue lengths (b) and throughputs (c) clearly show that DistGreedy outperforms the IEEE 802.11 DCF.



(a) Transmission rate



(b) Queue length



(c) Throughput

Figure 8: Performance of Client 3. Similar results as in Fig. 7 are observed when we focus on the performance of a client.

Fig. 7 shows that our DistGreedy implementation achieves better network performance in terms of reduced total queue length (Fig. 7(b)) and total throughput performance (Fig. 7(c)) under similar channel states. Indeed, DistGreedy implementation maintains queue lengths low for all the three clients, and IEEE 802.11 DCF results in many drops due to buffer overflows. For example, the queue length of Client 3 frequently increases up to 100 under IEEE 802.11 DCF even through it does not show up in Fig. 8 due to exponential weighted time averaging.

In Figs. 8(a) and 8(b), we can observe that under IEEE 802.11 DCF, high link rate (i.e., in $[60, 120]$ and $[180, 240]$) does not lead to low queue lengths, since IEEE 802.11 DCF does not opportunistically exploit wireless fading. In contrast, it shows that the DistGreedy implementation successfully keeps the queue length low, especially when the link rate is high. This implies that DistGreedy can account for the channel variations, and thus takes the advantage of the opportunistic gains. Another interesting observation is that in Fig. 8(c), IEEE 802.11 DCF often suffers from poor throughput performance when the server moves, i.e., after 120 and 240 seconds, while DistGreedy implementation maintains high throughput performance during the moves.

6. CONCLUSION

In this paper, we develop a distributed scheduling scheme that is provably efficient under wireless fading. By taking a local greedy approach, we prove that our scheme is a $\frac{1}{\Delta^*}$-approximation to the Maximum Weighted Independent Set problem, where Δ^* is the maximum conflict degree, and has $O(\log |V|)$ complexity (or polylogarithmic complexity), where $|V|$ is the number of links.

We evaluate our scheme through simulations in grid networks and random networks. The results show that our distributed scheduling scheme is insensitive to parameter settings, and achieves throughput and delay performance similar to those of the optimal solution. We also implement our scheme with hardware by modifying the existing IEEE 802.11 DCF. The experimental results show that our modification results in better throughput performance with low queue length by taking into account time-varying link capacities.

7. REFERENCES

[1] L. Tassiulas and A. Ephremides, “Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximal Throughput in Multihop Radio Networks,” *IEEE Trans. Autom. Control*, vol. 37, no. 12, Dec. 1992.

[2] C. Joo, G. Sharma, N. B. Shroff, and R. R. Mazumdar, “On the Complexity of Scheduling in Wireless Networks,” *EURASIP Journal of Wireless Communications and Networking*, 2010.

[3] E. Modiano, D. Shah, and G. Zussman, “Maximizing Throughput in Wireless Networks via Gossiping,” *Sigmetrics Performance Evaluation Review*, vol. 34, no. 1, 2006.

[4] L. Bui, S. Sanghavi, and R. Srikant, “Distributed Link Scheduling with Constant Overhead,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, Oct. 2009.

[5] L. Jiang and J. Walrand, “A Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 13, June 2010.

[6] J. Ni, B. Tan, and R. Srikant, “Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, June 2012.

[7] M. Lotfinezhad and P. Marbach, “Delay Performance of CSMA Policies in Multihop Wireless Networks: A New Perspective,” in *Information Theory and Application Workshop*, Feb. 2010.

[8] J. Ghaderi and R. Srikant, “Effect of Access Probabilities on the Delay Performance of Q-CSMA Algorithms,” in *IEEE INFOCOM*, April 2012.

[9] C. Joo and N. B. Shroff, “Local Greedy Approximation for Scheduling in Multi-hop Wireless Networks,” *IEEE Trans. Mobile Computing*, vol. 11, no. 3, March 2012.

[10] X. Lin and S. Rasool, “Constant-Time Distributed Scheduling Policies for Ad Hoc Wireless Networks,” *IEEE Trans. Autom. Control*, vol. 54, no. 2, Feb. 2009.

[11] C. Joo, “On Random Access Scheduling for Multimedia Traffic in Multi-hop Wireless Networks,” 2012, to appear in IEEE Trans. Mobile Computing.

[12] B. Li and A. Eryilmaz, “A Fast-CSMA Algorithm for Deadline Constraint Scheduling over Wireless Fading Channels,” in *Workshop on Research Allocation and Cooperation in Wireless Networks (RAWNET)*, May 2011.

[13] B. Birand, M. Chudnovsky, B. Ries, P. Seymour, G. Zussman, and Y. Zwols, “Analyzing the Performance of Greedy Maximal Scheduling via Local Pooling and Graph Theory,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, Feb. 2012.

[14] X. Wu and R. Srikant, “Scheduling Efficiency of Distributed Greedy Scheduling Algorithms in Wireless Networks,” in *IEEE INFOCOM*, 2006.

[15] X. Liu, E. K. P. Chong, and N. B. Shroff, “A Framework for Opportunistic Scheduling in Wireless Networks,” *Computer Networks*, vol. 41, no. 4, March 2003.

[16] S. Yun, J. Shin, and Y. Yi, “Medium Access over Time-varying Channels with Limited Sensing Cost,” *CoRR*, vol. abs/1206.5054, 2012.

[17] S. Sakai, M. Togasaki, and K. Yamazaki, “A Note on Greedy Algorithms for the Maximum Weighted Independent Set Problem,” *Discrete Appl. Math.*, vol. 126, no. 2-3, March 2003.

[18] P.-J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang, “Wireless Link Scheduling under Physical Interference Model,” in *IEEE INFOCOM*, April 2011.

[19] J.-H. Hoepman, “Simple Distributed Weighted Matchings,” eprint, Oct. 2004. [Online]. Available: http://arxiv.org/abs/cs/0410047v1

[20] S. Basagni, “Finding a Maximal Weighted Independent Set in Wireless Networks,” *Telecommunication Systems*, vol. 18, 2001.

[21] P. Gupta and P. R. Kumar, “The Capacity of Wireless Networks,” *IEEE Trans. Inf. Theory*, vol. 46, no. 2, March 2000.

[22] S. Sarkar and L. Tassiulas, “End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Ad-hoc Networks,” in *IEEE CDC*, Dec. 2003.

[23] M. J. Neely, E. Modiano, and C. E. Rohrs, “Dynamic Power Allocation and Routing for Time-varying Wireless Networks,” *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, 2005.

[24] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, “On the Stability of Input-Queued Switches with Speed-Up,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 1, Feb. 2001.

[25] X. Lin and N. B. Shroff, “The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, April 2006.

[26] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar, “Throughput and Fairness Guarantees Through Maximal Scheduling in Wireless Networks,” *IEEE Trans. Inf. Theory*, vol. 54, no. 2, Feb. 2008.

[27] G. Sharma, C. Joo, N. B. Shroff, and R. R. Mazumdar, “Joint Congestion Control and Distributed Scheduling for Throughput Guarantees in Wireless Networks,” *ACM Trans. Model. and Comput. Simul.*, vol. 21, no. 1, Dec. 2010.

[28] “Atheros Linux Wireless Drivers,” http://wireless.kernel.org/en/users/Drivers/ath5k.

[29] “Voyage Linux,” http://linux.voyage.hk.

[30] “ALIX system boards,” http://pcengines.ch/alix.htm.