Distributed Scheduling Schemes for Throughput Guarantees in Wireless Networks

Gaurav Sharma, Changhee Joo, and Ness B. Shroff

Abstract— **Wireless networks differ from their wireline counterparts in that a group of simultaneously active wireless links interfere with each other. This interference between links significantly complicates the scheduling component in wireless systems. This is especially problematic in multi-hop wireless networks, where the number of nodes can be quite large and one cannot assume the presence of a central authority. The extent of the interference varies across networks as it is physical layer dependent, and with it varies the complexity of scheduling. In this work, we discuss some commonly used interference models in the literature along with distributed scheduling schemes that can provide provable throughput guarantees under those models. A detailed analysis of the communication overhead is provided for the scheduling schemes, and the results indicate that this overhead can significantly impact the network throughput.**

I. Introduction

Wireless communication has seen a phenomenal growth over the last decade. However, many challenges still need to be overcome before we can fully utilize the potential of wireless communication. One of the main challenges is the optimal design of multihop wireless networks, which is far more difficult than is wireline counterpart. The reason is mainly due to the phenomenon of interference that introduces coupling across various layers of the protocol stack, including the physical, medium access control (MAC), network, and transport layers. This has spurred recent interest in developing cross-layer design algorithms (see, for example, [2], [4], [10], [11], [14], [15], [16], [18], [21], [22], [25], [27], [29], [30]) for such networks.

The cross-layer design problems have been shown to exhibit a nice decoupling property (see, for example, [11], [29]). More precisely, a cross-layer design problem can be decomposed into multiple subproblems, where each subproblem requires optimization across a single layer. The subproblems are coupled through parameters that correspond to congestion prices or queue lengths at the links or nodes. This structure of the cross-layer design implies that it has roughly the same complexity as the layered design, provided the congestion price information can be properly maintained at each link or node.

Gaurav Sharma is with Center for Wireless Systems and Applications, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA gsharma@purdue.edu

Changhee Joo is with Center for Wireless Systems and Applications, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA cjoo@purdue.edu

Ness B. Shroff is with Center for Wireless Systems and Applications, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA shroff@purdue.edu

The main component of any optimal cross-layer design scheme is the throughput-optimal scheduler that is required to solve a very difficult global optimization problem of the form:

$$\textbf{(OPT-SCHED)} \quad \textbf{maximize} \sum_{l \in E} p_l r_l \quad \textbf{subject to } r \in \Delta,$$

where E denotes the set of wireless links; r is the vector of link rates r_l, $l \in E$; p_l, $l \in E$, is the congestion price or possibly some function of queue length at link l; and Δ is the capacity region of the network. Readers familiar with the seminal work in [26] will note that OPT-SCHED is a generalization of the problem that the maximum throughput policy needs to solve. The generality of OPT-SCHED is a consequence of the more generic physical layer model used in its formulation as compared to the one used in [26].

OPT-SCHED requires a network wide optimization, which is difficult to perform in a distributed environment. In fact, OPT-SCHED can be shown to be NP-Complete and Non-Approximable under most settings (see [20], [21]). Thus, it is hard to carry out the required optimization in OPT-SCHED even with centralized control. To overcome this difficulty, some alternative techniques have been developed in the literature that either allow one to obtain long term throughput guarantees without having to solve OPT-SCHED at every slot or impose a certain structure on the underlying connectivity graph of the wireless network.

In this paper, we first discuss the complexity of OPT-SCHED under various commonly used interference models, with or without imposing a structure on the underlying connectivity graph of the wireless network. We then discuss some alternative schemes that allow one to provide long term throughput guarantees without having to solve OPT-SCHED at every slot. The emphasis will be on a rigorous analysis of the time complexity for practical implementations of such schemes.

II. Complexity of OPT-SCHED

In this section, we discuss various interference models studied in the literature and discuss the complexity of scheduling under each of them. We start with a description of our system model.

A. System Model

We consider a set V of nodes, communicating with each other using wireless means. All transmissions are carried out

over the same frequency band, and therefore, interfere with each other. Each node uses the same power level for all its transmission; this power level can be different for different nodes. A (undirected) link is said to exist between two nodes if each of them can successfully receive from the other, provided no other node in the network transmits at the same time. The set of (undirected) links so formed is denoted by E. Note that the existence of a link between two nodes is dependent on many factors including the power allocated to the nodes; noise variances at the nodes; and coding and modulation schemes used at the nodes.

B. Interference Models

A general form of interference model studied in the literature is the *contention matrix based interference model*. In such an interference model, any given link can interfere with any other link in the network. The structure of the link interference is specified using a contention matrix, whose entries are set to 1 or 0 depending on whether the corresponding set of links interfere or not. Given a set of links, all links in the set that have no other interfering link within the set are allowed to transmit at some predetermined rate. Using the results in [20], OPT-SCHED can be shown to be NP-Complete and Non-Approximable under a general contention matrix based interference model.

A more restrictive class of interference models is the class of K-hop interference models studied in [20], [21]. In the case of a K-hop interference model, links that are within K-hops of each other are said to interfere with each other. Note that a K-hop model restricts the structure of interference constraints that can be realized using a contention matrix based interference model.

Of particular interest within the class of K-hop interference models are the 1-hop and 2-hop interference models studied in many related works [1], [5], [6], [12], [27] that are applicable in case of FH-CDMA and IEEE 802.11 DSSS based networks, respectively. Simulation results in [21] suggest that the 2-hop interference model can effectively capture the interference constraints in many different networks.

The OPT-SCHED problem corresponds to a weighted matching problem under the 1-hop interference model and can be solved in polynomial time. A distributed 2-hop approximation is also possible in this case. However, under a K-hop interference model with $K \geq 2$, OPT-SCHED is NP-Complete and Non-Approximable (see [20], [21]).

C. Graph Models

A possible way of dealing with the complexity of OPT-SCHED is to impose a certain structure on the underlying connectivity graph G. Indeed, under an SINR threshold based setting where nodes experience identical noise variance and have the same maximum transmit power, the graph G can be assumed to be a geometric graph. In [21], it is shown that one can obtain centralized $(1+\varepsilon)$-approximation algorithms for OPT-SCHED under all K-hop interference models in the case of geometric graphs. However, it is not clear whether one can develop distributed approximation algorithms for OPT-SCHED with a good approximation ratio and low communication complexity in the case of geometric graphs.

III. Long Term Throughput Guarantees

We now discuss how one can obtain long term throughput guarantees without having to solve OPT-SCHED at every slot. The approaches followed in the literature can fall into three main categories described below.

A. Pick and Compare Approach

In this approach, a set of links satisfying the interference constraints is picked at each time slot and the weight of the chosen set of links is compared with the set of links chosen to transmit during the previous slot; and the one with maximum weight is chosen for transmission during the current slot* [13], [24], [31]. This approach was first proposed and analyzed in [24], where it was shown to provide maximum possible throughput. Note that the analysis carried out in [24] does not account for the loss in throughput due to the time complexity of implementing such an algorithm in practice. However, comparing the weights of two given matchings requires network wide computation and message exchange and may incur substantial overhead in terms of time, even under the simplest of interference models, such as the 1-hop interference model.

More precisely, observe that an algorithm that compares the weight of any two given matchings will at least require control message exchanges of the order of the network diameter, henceforth denoted by $D(G)$. If such an algorithm is run at every slot, then the actual throughput obtained will be at least a factor of $D(G)$ away from the optimal. In the case of most wireless networks, the diameter of the network can be of the order of $\sqrt{|V|}$ (e.g., random geometric graphs with communication radius of each node set to the minimum communication radius required for the almost sure connectivity of the network, grid networks). In such a case, the throughput provided by any Pick and Compare scheme could be substantially smaller than the optimal throughput.

B. Maximal Scheduling Based Approach

We discuss Maximal scheduling based schemes, which require only local message exchange and can provide much better throughput in practice than Pick and Compare schemes.

We construct the constraints of a maximal schedule as follows. Let $\mathscr{Q}$ denote the set of links l with $q_l > 1$, where q_l is the congestion price of link l, and let $I(l)$ be the set of links that interfere with link l, inclusive of l. Any schedule $\mathscr{S}$ that satisfies the following two conditions is a maximal matching.

- For each $l \in \mathscr{S}$, $\mathscr{S} \cap I(l) = \{l\}$.
- For all $l \in \mathscr{Q}$, $\mathscr{S} \cap I(l) \neq \emptyset$.

The first constraint states that if link l is scheduled, no other link in its interference range can be scheduled. The second

*The ties, if any, can be resolved in some arbitrary manner.

constraint ensures that there is at least a link scheduled in the interference range of link l with $q_l > 1$.

A Maximal scheduling policy is a scheduling policy under which the set of links chosen for transmission for slot t is a maximal schedule $\mathscr{S}(t)$. Note that Maximal scheduling does not require one to solve or even approximate within a constant factor the problem OPT-SCHED [11], [21], [23], [28]. The Maximal scheduling policy imposes a very weak set of constraints on the set of links chosen for transmission during any slot and this makes it amenable to distributed implementation.

In [21], it is shown that the Maximal scheduling policy can support at least $1/d_I(G)$ of the maximum achievable throughput under any scheduling policy, where $d_I(G)$ denotes the interference degree of the graph G. Here, $d_I(G)$ is the largest one between the maximum number of simultaneous transmissions within $I(l)$ for each l[†]. The Maximal scheduling based approach is particularly attractive because $d_I(G)$ is usually small in most practical cases. For example, in the case of geometric graphs and K-hop interference models, following bounds on $d_I(G)$ have been shown in [3], [21] as

$$d_I(G) \leq \begin{cases} 2 & \text{for } K = 1 \\ 8 & \text{for } K = 2 \\ \dfrac{(2K+1)^2}{\lfloor K/2 \rfloor^2} & \text{for } K \geq 3. \end{cases}$$

C. Constant-Time Scheduling Approach

While Maximal scheduling is quite amenable to distributed implementation, the number of rounds of computation and local (control) message exchange required in each slot may scale with the number of nodes (the dependence is usually polylogarithmic). This is not desirable to the large networks, in which such behavior can lead to significant performance decrease.

A Constant-Time scheduling scheme is a scheduling scheme with a constant, independent of $|V|$, number of computation and local message exchange rounds at every slot. The usefulness of Constant-Time scheduling stems from the fact that in spite of allowing for a constant number of communication and local message exchange rounds per slot, the throughput guarantee they provide is comparable to Maximal scheduling [7], [9]. We will discuss their performance in the next section.

IV. A Comparison of Maximal Scheduling, Pick and Compare, and Constant-Time Scheduling Approaches

In this section we compare the performance of Maximal scheduling, Constant-Time scheduling, and Pick and Compare scheduling. Due to space constraints, we limit our discussion to the 1-hop and 2-hop interference models.

We begin with the Maximal scheduling. Although, it has been considered in many of the previous works [3], [11], [21], the difficulties of its implementation in a distributed fashion have not been dealt with explicitly. We now provide two randomized distributed algorithms for implementing the Maximal scheduling policy under the 1-hop and 2-hop interference models. Both these algorithms are inspired by a classical algorithm for constructing maximal independent set in [17].

We first describe our distributed computing model. As in [5] and other related works, we assume a *synchronous message passing distributed computing model*, which is a variation of the standard models used in the distributed computing literature. The main point of difference is the broadcast nature of the model which is typical of wireless networks. More precisely, we model the distributed computing architecture as a graph with undirected edges (we assume bidirectional links between the nodes as specified in the 1-hop and 2-hop interference models). Each node has a unique ID. The clocks at all the nodes are synchronized and the communication takes place in rounds, each occupying a mini-slot. A packet transmission from node u is heard by all nodes v in its neighborhood, unless the node v itself transmits or some other neighbor of node v also transmits. This interference between simultaneous transmissions makes it difficult to implement a Maximal scheduling policy in wireless networks.

A. Randomized Maximal Scheduling under 1-Hop Interference

We now propose a randomized distributed algorithm, namely MaxScheduleOneHop, that implements the Maximal scheduling policy under the 1-hop interference model.

We need to define some terminology before we can give a description of MaxScheduleOneHop:

- uv: An undirected link between nodes u and v.
- $N(u)$: The set of neighbors of node u, i.e., nodes $v \in V$ such that $uv \in E$.
- $N^{\mathcal{Q}}(u)$: The set of nodes $v \in N(u)$ such that $q_{uv} > 1$.
- $d(u)$: $\max\limits_{v \in N(u)} |N(v)|$.
- $d^{\mathcal{Q}}(u)$: $\max\limits_{v \in N^{\mathcal{Q}}(u)} |N^{\mathcal{Q}}(v)|$.
- Δ: The maximum node degree, i.e., $\max\limits_{v \in V} |N(v)|$.

MaxScheduleOneHop uses three subroutines, namely UpdatePrices, CompAndDistNeighborhoods, and UpdateAndDistNeighborhoods. As the names suggest, these subroutines are used for the following purpose:

i) **UpdatePrices:** The subroutine allows each node in the network to update the congestion prices of its outgoing links.
ii) **CompAndDistNeighborhoods:** The subroutine allows each node v to calculate $N^{\mathcal{Q}}(v)$ and $d^{\mathcal{Q}}(v)$ at every slot based on current congestion prices.
iii) **UpdateAndDistNeighborhoods:** The subroutine allows each node to remove those nodes from its current neighborhood that were scheduled to transmit or receive during the current phase of MaxScheduleOneHop. More precisely, each node computes $N^{\mathcal{Q}}(v)$ and

[†]Precisely, $d_I(G) = \max_{l \in E} d_I(l)$, where $d_I(l)$ is the maximum number of simultaneous transmissions without interference within $I(l)$.

$d^{\mathcal{Q}}(v)$ by considering only those nodes that have as yet not been scheduled to transmit or receive.

We are now ready to give a description of MaxSceduleOneHop in Fig. 1. The subroutines used by MaxScheduleOneHop

MaxScheduleOneHop$(G, q(t))$

i) q(t+1) := **UpdatePrices(**$G, q(t)$**)**;
ii) **CompAndDistNeighborhoods(**$G, q(t+1)$**)**;
iii) $\mathscr{S}_0(t+1) := \phi$ and $b(u) = -1$ for all $u \in V$.
iv) **for** $p = 1$ to $\lceil C_P \log |V| \rceil$ **do**
v) $\quad \mathscr{S}_p(t+1) := \mathscr{S}_{p-1}(t+1)$;
vi) $\quad$ **for** $i = 1$ to $C_I \log |V|$ **do**
vii) $\quad\quad$ Each node u with $b(u) = -1$ chooses to transmit with probability $\frac{1}{d^{\mathcal{Q}}(u)+1}$. Upon deciding to transmit, it chooses a node v at random from $N^{\mathcal{Q}}(u)$ and sends a RTS message to node v.
viii) $\quad\quad$ If the RTS packet is successfully received by node v, it responds with a CTS message and sets $b(v) = 1$.
ix) $\quad\quad$ Upon receiving the CTS packet, node u sets $b(u) = 1$. $\mathscr{S}_p(t+1) := \mathscr{S}_p(t+1) \cup uv$.
x) $\quad$ **end for**
xi) **UpdateAndDistNeighborhoods(**$G, \mathscr{S}_{p-1}(t+1), \mathscr{S}_p(t+1)$**)**;
xii) **end for**

Fig. 1. MaxScheduleOneHop$(G, q(t))$

will be described next, beginning with UpdateAndDistNeighborhoods in Fig. 2.

UpdateAndDistNeighborhoods(G, S_{pr}, S_{cr})

i) **for each** $v \in V$ **do**
ii) $\quad$ **if there exists** $uv \in S_{cr} \backslash S_{pr}$
iii) $\quad\quad$ **ReliablyBroadcast**$(v,$ matched to $u)$;
iv) $\quad$ **end if there exists**
v) $\quad$ Compute $N^{\mathcal{Q}}(v)$ considering only those nodes in $N(v)$ that are currently unmatched.
vi) $\quad$ **ReliablyBroadcast**$(v, |N^{\mathcal{Q}}(v)|)$;
vii) $\quad$ Compute $d^{\mathcal{Q}}(v)$.
viii) **end for each**

Fig. 2. UpdateAndDistNeighborhoods(G, S_{pr}, S_{cr})

UpdateAndDistNeighborhoods allows each node to update $N^{\mathcal{Q}}(v)$ and $d^{\mathcal{Q}}(v)$ at the end of each phase in MaxScheduleOneHop. If a node was matched during the current phase, it broadcasts this information to its neighbors using the subroutine ReliablyBroadcast. With this knowledge, each node v updates $N^{\mathcal{Q}}(v)$ by deleting those neighboring nodes which were matched during the current phase. Once this is done, each node v uses the subroutine ReliablyBroadcast to broadcast $|N^{\mathcal{Q}}(v)|$ to its neighbors; and based on this knowledge each node v updates $d^{\mathcal{Q}}(v)$. The subroutine ReliablyBroadcast is described in Fig. 3.

ReliablyBroadcast$(v, data)$

i) **for** $k = 1$ to $\lceil C_K \Delta \log |V| \rceil$ **do**
ii) $\quad$ Broadcast **data** with probability $\frac{1}{d(v)+1}$ to your neighbors.
iii) **end for**

Fig. 3. ReliablyBroadcast$(v, data)$

Next, we describe the subroutine CompAndDistNeighborhoods in Fig. 4.

CompAndDistNeighborhoods(G, q)

i) **for each** $v \in V$ **do**
ii) $\quad$ Compute $N^{\mathcal{Q}}(v)$.
iii) $\quad$ **ReliablyBroadcast**$(v, |N^{\mathcal{Q}}(v)|)$;
iv) $\quad$ Compute $d^{\mathcal{Q}}(v)$.
v) **end for each**

Fig. 4. CompAndDistNeighborhoods(G, q)

CompAndDistNeighborhoods allows each node to compute $N^{\mathcal{Q}}(v)$ and $d^{\mathcal{Q}}(v)$. The computation of $N^{\mathcal{Q}}(v)$ is straightforward. Once this is done, each node v uses the subroutine ReliablyBroadcast to broadcast $N^{\mathcal{Q}}(v)$ to its neighbors; and based on this knowledge each node v computes $d^{\mathcal{Q}}(v)$.

UpdatePrices allows each node to compute the congestion prices of its outgoing links based on the local knowledge of the schedule $\mathscr{S}(t)$. In the case where the congestion price of each link is only a function of its own backlog, it does not involve any message exchange[‡].

In [19], it is shown that if the constants C_P, C_I, and C_K are chosen appropriately, then MaxScheduleOneHop returns a subset of links that satisfies the constraints imposed by the maximal scheduling under 1-hop interference with high probability (in $|V|$).

We examine the time complexity of MaxScheduleOneHop. Each execution of UpdateAndDistNeighborhoods involves $\Theta(\Delta \log |V|)$ rounds of computation and local message exchange. Since MaxScheduleOneHop has $\Theta(\log |V|)$ phases and UpdateAndDistNeighborhoods is run at the end of each phase, MaxScheduleOneHop involves $\Theta(\Delta \log^2 |V|)$ rounds of computation and local message exchange. Note, however, that if the maximum node degree in the network is significantly smaller than $|V| - 1$, then one can reduce the number of phases in MaxScheduleOneHop to $\lceil C_P \log \Delta \rceil$. MaxScheduleOneHop would then require $\Theta(\Delta \log \Delta \log |V|)$ rounds of computation and local message exchange. For example, if the maximum node degree in the network is $\Theta(\log |V|)$, as in the case of random geometric graphs, then

[‡]Note that in the setting considered in [19], the congestion price of a link depends not only on its own backlog, but also on the backlogs at the links that interfere with the given link. In this paper, however, we do not consider such a dependence.

by reducing the number of phases in MaxScheduleOneHop from $\Theta(\log|V|)$ to $\Theta(\log\log|V|)$, we can reduce its running time from $\Theta(\log^3|V|)$ to $\Theta(\log^2|V|\log\log|V|)$.

B. Randomized Maximal Scheduling under 2-Hop Interference

In this subsection, we propose a randomized distributed algorithm for implementing the Maximal scheduling policy under the 2-hop interference model. Recall that under the 2-hop interference model no two links that are within two hops of each other can be scheduled to transmit or receive at the same time. The distributed computing model we adopt is the same as before. In particular, the model of interference between control packets is still the same. The reason for having a different interference model for data and control packets is that in most networks (e.g., IEEE 802.11 based networks) the control packets are usually much smaller in size than the data packets and are often transmitted at a much smaller rate than the data packets. Correspondingly, successful reception of a control packet requires much less SINR as compared to that of a data packet; thereby motivating the use of different interference models for each of them.

The algorithm we propose in this section, named MaxScheduleTwoHop, is conceptually very similar to MaxScheduleOneHop. However, there are some additional difficulties that arise in case of 2-hop interference and are dealt with in MaxScheduleTwoHop. Indeed, a distinguishing feature of MaxScheduleTwoHop is the exchange of *collision* (COL) packets to ensure that no two links that are within two hops of each other decide to transmit or receive at the same time. More precisely, if a sender node detects an ongoing transmission while transmitting the *request to send* (RTS) packet, it sends a subsequent COL packet. Successful reception of an RTS packet by the receiver guarantees that no other transmitter can be within one hop of the receiver. Further, no collision packet being sent guarantees that no two nodes that are within one hop of each other can decide to transmit at the same time. If the receiver does not hear a COL packet or a collision due to multiple such packets, it sends a *clear to send* (CTS) packet. If it detects an ongoing transmission while transmitting the CTS packet, it subsequently sends a COL packet. No collision packet being sent guarantees that no two nodes within one hop can decide to receive at the same time.

We need the following notation in order to facilitate our discussion of MaxScheduleTwoHop:

- $N_2(u)$: The set of two hop neighbors of node u, i.e., $\cup_{v\in N(u)}N(v)$.
- q_{uv}: $\max(q_{(u,v)},q_{(v,u)})$.
- $N_2^Q(u)$: The set of undirected links $vw\in E$ such that $v\in N(u)$ and $q_{uw}>1$.
- $d_2^Q(u)$: $\max\limits_{v\in N_2(u)}|N_2^Q(v)|$.

We provide a detailed description of MaxScheduleTwoHop in Fig. 5.

MaxScheduleTwoHop uses three subroutines, namely UpdatePrices, CompAndDistTwoHopNeighborhoods, and UpdateAndDistTwoHopNeighborhoods. UpdatePrices allows each node to update the congestion price of its outgoing links and is essentially the same as in the case of MaxScheduleOneHop[§]. The latter two subroutines are similar to their counterparts in the case of MaxScheduleOneHop. The main difference, however, is that unlike their counterparts, these subroutines must broadcast information over two hops in order for each node v to be able to (i) compute $N_2^Q(v)$, which requires the knowledge of congestion prices of the links within two hops; and (ii) compute $d_2^Q(v)$ using the knowledge of $N_2^Q(w)$ for $w\in N_2(v)$. Broadcasting information over two hops is accomplished by using the subroutine ReliablyBroadcast twice in a series, once with your local information and then with the information obtained from your one hop neighbors. The detailed descriptions of these subroutines are provided in Fig. 6.

UpdateAndDistTwoHopNeighborhoods allows each node

MaxScheduleTwoHop$(G,q(t))$

i) q(t+1) := **UpdatePrices**$(G,q(t))$;
ii) **CompAndDistTwoHopNeighborhoods**$(G,q(t+1))$;
iii) $\mathscr{S}_0(t+1):=\phi$ and $b(u)=-1$ for all $u\in V$.
iv) **for** $p=1$ to $C_P\log|V|$ **do**
v) $\mathscr{S}_p(t+1):=\mathscr{S}_{p-1}(t+1)$;
vi) **for** $i=1$ to $C_I\log|V|$ **do**
vii) Each node u with $b(u)=-1$ chooses to transmit with probability $\frac{1}{d_2^Q(u)+1}$. Upon deciding to transmit, it chooses a node v at random from $N^Q(u)$ and sends a RTS message to node v.
viii) If a transmitting node detects any other transmission while transmitting, then it sends a COL packet immediately after the RTS packet.
ix) If a receiver node successfully receives the RTS packet and does not subsequently hear a COL packet (or a collision involving multiple Cpackets), then it sends a CTS packet.
x) If a receiver node v detects any other transmission while transmitting the CTS packet, then it sends a COL packet immediately after the CTS packet. Otherwise, it sets $b(v)=1$.
xi) If a sender node u hears the CTS packet from its intended receiver but no subsequent COL packet, it sets $b(u)=1$. $\mathscr{S}_p(t+1):=\mathscr{S}_p(t+1)\cup uv$.
xii) **end for**
xiii) **UpdateAndDistTwoHopNeighborhoods**$(G,\mathscr{S}_{p-1}(t+1),\mathscr{S}_p(t+1))$;
xiv) **end for**

Fig. 5. MaxScheduleTwoHop$(G,q(t))$.

[§]Note that when one considers a setting in which the congestion price of a link depends on the backlogs of all the links that interfere with it (see [19]), then UpdatePrices requires a different implementation under the 1-hop and 2-hop interference models.

UpdateAndDistTwoHopNeighborhoods(G, S_{pr}, S_{cr})

i) **for each** $v \in V$ **do**
ii) **if there exists** $uv \in S_{cr} \backslash S_{pr}$
iii) **ReliablyBroadcast**$(v,$ matched to $u)$;
iv) **else**
v) Wait for $\lceil C_K \Delta \log |V| \rceil$ control slots to collect the information from your neighbors.
vi) **end if there exists**
vii) **ReliablyBroadcast**$(v,$ collected information$)$;
viii) Compute $N_2^Q(v)$.
ix) **ReliablyBroadcast**$(v, |N_2^Q(v)|)$;
x) **ReliablyBroadcast**$(v, |N_2^Q(w)|$ for all $w \in N(v))$;
xi) Compute $d_2^Q(v)$.
xii) **end for each**

Fig. 6. UpdateAndDistTwoHopNeighborhoods(G, S_{pr}, S_{cr}).

CompAndDistTwoHopNeighborhoods(G, q)

i) **for each** $v \in V$ **do**
ii) Info := Queue length of all outgoing links.
iii) **ReliablyBroadcast**$(v,$ Info$)$;
iv) Compute q_{vw} for $w \in N(v)$.
v) **ReliablyBroadcast**$(v, q_{vw}$ for all $w \in N(v))$;
vi) Compute $N_2^Q(v)$.
vii) **ReliablyBroadcast**$(v, |N_2^Q(v)|)$;
viii) **ReliablyBroadcast**$(v, |N_2^Q(w)|$ for all $w \in N(v))$;
ix) Compute $d_2^Q(v)$.
x) **end for each**

Fig. 7. CompAndDistTwoHopNeighborhoods(G, q)

to update $N_2^Q(v)$ and $d_2^Q(v)$ at the end of each phase in MaxScheduleTwoHop. If a node was matched during the current phase, it broadcasts this information to its neighbors using the subroutine ReliablyBroadcast. Each node then broadcasts the information it obtains from its one hop neighbors further to its one hop neighbors. This allows each node v to calculate $N_2^Q(v)$. Once this is done, each node v uses the subroutine ReliablyBroadcast to send $|N_2^Q(v)|$ to its neighbors; which further broadcast this information to their one hop neighbors, enabling each node v to calculate $d_2^Q(v)$.

CompAndDistTwoHopNeighborhoods works in a fashion similar to UpdateAndDistTwoHopNeighborhoods, with the only difference being that the neighborhoods are updated due to the change in the congestion prices, instead of some links being scheduled for transmission.

As in the case of MaxScheduleOneHop, it can be shown that (see [19]) if the constants C_P, C_I, and C_K are chosen appropriately, then MaxScheduleTwoHop returns a subset of links that satisfies the constraints imposed by the maximal scheduling under 2-hop interference with high probability (in $|V|$). Moreover, it has the same time complexity as MaxScheduleOneHop, namely $\Theta(\Delta \log^2 |V|)$ rounds of computation and local message exchange.

C. Pick and Compare under 1-hop and 2-hop Interference

In this subsection, we discuss the time complexity of implementing a Pick and Compare scheme under the 1-hop and 2-hop interference models. As discussed in Section III-A, a Pick and Compare scheme has two components:

- **Pick Algorithm:** The pick algorithm is responsible for choosing a set of links that satisfies the interference constraints and has a probability of at least δ, for some $\delta > 0$, of being the maximum weighted set of links satisfying the interference constraints.
- **Compare Algorithm:** The compare algorithm is responsible for comparing the weights of two given sets of links and choosing the one with the maximum weight.

It is straightforward to design a pick algorithm with the above properties under the 1-hop and 2-hop interference models. Let us first consider the 1-hop interference model. In this case, we have the simple implementation of a pick algorithm as shown in Fig. 8.

Pick(G)

i) **for each** $v \in V$ **do**
ii) Choose to transmit with probability $\frac{1}{d(v)+1}$. Upon deciding to transmit, choose a node u at random from $N(v)$ and send a RTS packet to node u.
iii) If RTS packet is successfully received from some node u, then respond with a CTS packet.
iv) **end for**

Fig. 8. Pick(G).

Following the line of analysis in [19], one can show that given any arbitrary link, the probability that the link gets activated by the above algorithm is at least $e^{-2}/|V|$. Since no matching contains more than than $|V|/2$ links, the probability that the above algorithm activates a maximum weight matching is at least $(e^{-2}/|V|)^{\frac{|V|}{2}} > 0$.

The algorithm illustrated in Fig. 9, named PickTwoHop, is an extension of the Pick algorithm to a setting with 2-hop interference.

Applying the same analysis method, it can be shown that any given set of edges satisfying the 2-hop interference constraints would be activated by PickTwoHop with probability at least $(e^{-2}/|V|)^{\frac{|V|}{2}} > 0$.

From these results, we see that the pick operation can be performed in a distributed fashion and in a constant time under both 1-hop and 2-hop interference models. The difficulty in using a Pick and Compare scheme, however, is placed on the compare operation. It is quite difficult to implement in a distributed fashion and has large time complexity. Note that in order for each node to be able to compare the weights of two given sets of links, it must first

PickTwoHop(G)

i) **for each** $v \in V$ **do**
ii) Choose to transmit with probability $\frac{1}{d_2(v)+1}$. Upon deciding to transmit, choose a node u at random from $N(v)$ and send a RTS packet to node u.
iii) Send a COL packet if you detect any other transmission while transmitting.
iv) If you hear a RTS packet but no subsequent COL packet (or a collision involving multiple COL packets), then send a CTS packet.
v) Send a COL packet if you detect any other transmission while transmitting.
vi) **end for**

Fig. 9. PickTwoHop(G)

obtain the link weight information from across the network, which requires number of slots of the order of the network diameter[¶] $D(G)$. In the worst case, $D(G)$ can be of the order of $|V|$. For networks whose connectivity graph is a 2-D grid or a random geometric graph (with each node having the minimum communication radius required for almost sure connectivity), the diameter $D(G)$ is of the order of $\sqrt{|V|}$ and $\sqrt{|V|/\log|V|}$, respectively.

Thus, we would expect any Pick and Compare scheme to have a time complexity of $\Theta(\sqrt{|V|})$ or more in practical scenarios. Since Δ is usually of the order of $\log|V|$ or less, a Maximal scheduling based scheme, with time complexity of $\Theta(\Delta\log^2|V|)$, would be expected to offer a much better throughput in practice as compared to a Pick and Compare scheme.

D. Constant-Time Scheduling under 1-hop and 2-hop interference

We now discuss the performance of Constant-Time scheduling, in which 'constant time' implies that it requires only a *finite* number contending rounds.

We define the following;

- M : The number of mini-slots (or rounds; fixed),
- q_l : The queue length of link l,
- c_l : The capacity of link l,
- $E(i)$: The set of links connected to node i,
- $N(l)$: The set of links neighboring link l.

Assuming that each node can overhear transmission within its interference range, the detailed algorithm is given in Fig. 10.

The attempt probability p_l is given as
i) 1-hop interference case:

$$p_l = \frac{\sqrt{M}-1}{2M} \cdot \frac{q_l/c_l}{\max\left(\sum_{k\in E(i)}\frac{q_k}{c_k}, \sum_{k\in E(j)}\frac{q_k}{c_k}\right)}, \qquad \text{(IV.1)}$$

[¶]Note that we assume throughout the paper that the network is connected.

ConstantTime(G)

i) **for each** $l \in E$ **do**
ii) Check if any link $k \in N(l)$ attempts transmission before this round.
iii) If no one attempts transmission, attempts transmission with probability p_l.
iv) **end for**

Fig. 10. ConstTime(G).

where i and j are two nodes consisting of link l.
ii) 2-hop interference case:

$$p_l = \frac{\sqrt{M}-1}{\Delta M} \cdot \frac{q_l/c_l}{\max_{k\in N(l)}\left(\sum_{h\in N(k)}\frac{q_h}{c_h}\right)}. \qquad \text{(IV.2)}$$

It is proven in [7] that the Constant-Time scheduling scheme with p_l given by (IV.1) achieves the performance within a factor of $\frac{1}{2}\left(1-\frac{2}{\sqrt{M}}\right)$ of the optimal for the 1-hop interference model, and with p_l given by (IV.2), $\frac{1}{\Delta}\left(1-\frac{2}{\sqrt{M}}\right)$ for the 2-hop interference model, respectively.

While the performance of the Constant-Time scheduling is within a factor of $\left(1-\frac{2}{\sqrt{M}}\right)$ of the performance of Maximal scheduling, the algorithm requires the weight information of neighboring links. Therefore, it could have time complexity up to $O(\Delta\log|V|)$.

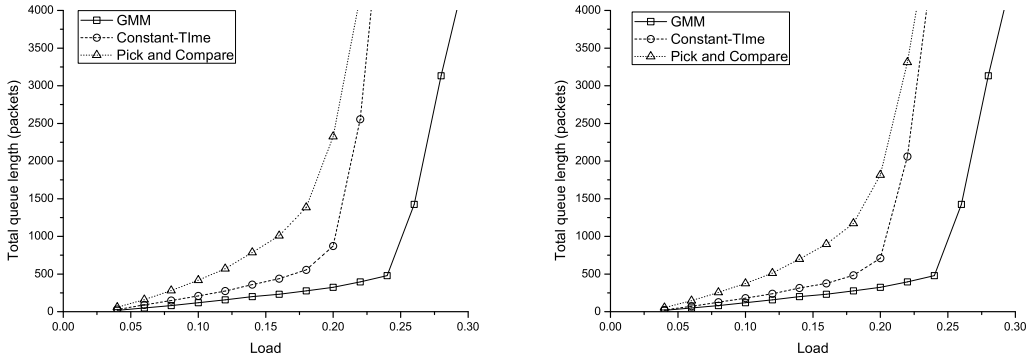
V. Simulation

We compare the performance of 'Greedy Maximal Matching (GMM)', 'Pick and Compare', and 'Constant-Time' scheduling in terms of the capacity region and their practical aspects of convergence time and queue occupation.

GMM constructs a maximal schedule by choosing the link of the largest weight first (without violating the interference constraint). It starts with an empty schedule $\mathscr{S}$ and the set of available links A, which is initially set to all links that have nontrivial amount of workload, that is, $\mathscr{Q}$. GMM chooses the link with the largest weight in A, say l, then adds it to $\mathscr{S}$, and removes all links in $I(l)$ from A. This procedure repeats until the available set A becomes empty. GMM is a centralized scheme and achieves the performance within a factor of $\frac{1}{2}$ of the optimal for the 1-hop interference model. We compare the performance of GMM for the reference, since it is often observed that it empirically performs as well as an optimal scheduling[‖].

We generate a random topology with 36 nodes in an 1x1 rectangular space and connect a link between two nodes if their distance is less than 0.3, which result in 128 *directed* links. The capacity of link (i,j), $c_{(i,j)}$ is assigned randomly in the range of [5, 10] packets per slot and it is symmetric, i.e., $c_{(i,j)} = c_{(j,i)}$. The traffic load at each link $\rho_{(i,j)}$ is Poisson distributed with mean of $\{0,\ \rho,\ 2\rho\}$ packets per slot, which is chosen randomly with probability $\{0.2,\ 0.6,$

[‖]The optimal scheduling scheme is too complex to simulate.



(a) $M = 32$ (b) $M = 64$

Fig. 11. Capacity region of scheduling schemes.

$0.2\}$, respectively. Load ρ is a controlling factor. Note that the load is not symmetric, i.e., $\rho_{(i,j)} \neq \rho_{(j,i)}$.

We use the 1-hop interference model and assume that node i has the weight information of its neighborhood, that is, node i is supposed to know $q_{(i,j)}$ and $q_{(j,i)}$ for all $j \in N(i)$, where $N(i)$ is the set of nodes connected to node i by a single hop. Since both Pick and Compare and Constant-Time scheduling require this local information, the comparison between them remains valid.

A slot is divided into two periods; one for control message exchange and the other for actual data transmissions. The period for message exchange consists of smaller mini-slots (rounds). Two messages can be exchanged in a single mini-slot, i.e., a single set of RTS and CTS transmission can happen. The overhead of message exchanges is counted by the number of rounds. Throughout the simulations, we set the cost of a mini-slot is 0.002^{**} compared to the actual data transmission time. Hence, letting the number of rounds denote M, the ratio of the overhead to the slot time is $\frac{1}{0.002 \cdot M+1}$. However, we do not take into account the overhead due to the message size. Especially, the state-of-the-art distributed algorithm for Pick and Compare [13] used in our simulations demands a large message in order to obtain quick estimation of the weights of the matchings for the Compare procedure.

Fig. 11 presents the sum of queue length over all nodes in the network. Since the capacity region of a scheduling scheme is defined as the amount of load while keeping the sum of queue length finite, a scheduling scheme with the smaller sum might achieve the better capacity region. In this context, it appears that the performance of the schemes is in decreasing order: GMM, Constant-Time, and Pick and Compare.

However, we also observe that there is a performance crossover between Pick and Compare and Constant-Time scheduling as shown in Fig. 12. It makes sense in that Pick and Compare could achieve 100% capacity region (without considering the overhead) and Constant-Time scheduling guarantees only 50%. Note that our simulation is done for a finite time. If we simulate for a longer time, the crossover may occur with a lighter load. With loads over 0.2, we observed that for Pick and Compare and Constant-Time scheduling, the sum of queue length is still increasing when the simulation ends.

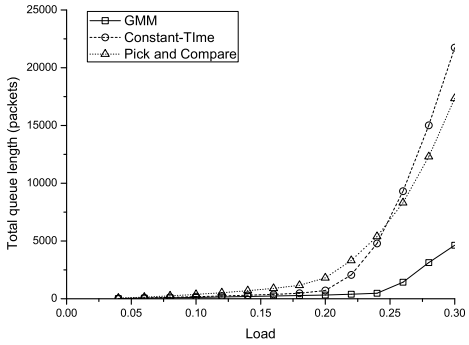
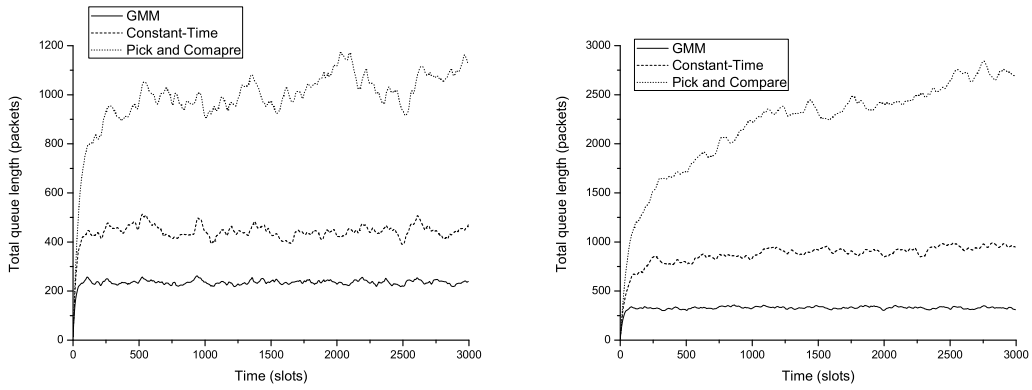


Fig. 12. Performance crossover of Pick and Compare and Constant-Time scheduling; $M = 64$.

Fig. 13 shows two instances of increasing queue lengths when 32 mini-slots are used. With a load of 0.16, all three schemes have the sum of queue length bounded by certain levels. However, with a load of 0.2, while GMM and Constant-Time have bounded queue lengths, the queue length of Pick and Compare is still increasing when the simulation ends. It may eventually stop increasing, but what is unknown is when and where. Since Pick and Compare takes a relatively long time to achieve optimality, it is less attractive for practical implementation.

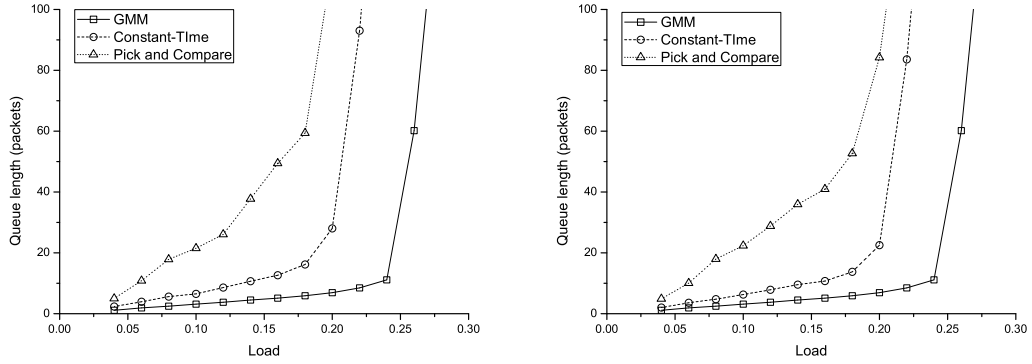
The importance of the queue length needs to be emphasized, especially when a node has finite buffer space. Fig. 14 illustrates that Pick and Compare demands a much larger buffer size than GMM and Constant-Time scheduling. This is

**In the standard IEEE 802.11 networks, a contention slot is 20us. If we assume that a packet size is 1000bytes, the transmission of 5 packets (the minimum link capacity in our simulation is 5 packets per slot.) over 2Mbps takes 20ms. Hence, the ratio of two slots, for an RTS and a CTS, to the data transmission period is 0.002.



(a) Load = 0.16 (b) Load = 0.2

Fig. 13. Convergence of scheduling schemes; $M = 32$.



(a) $M = 32$ (b) $M = 64$

Fig. 14. Maximal queue occupation of scheduling schemes.

because while GMM and Constant-Time scheduling instantly prefer a link with a larger weight, Pick and Compare takes some time to determine the link. It takes longer for Pick and Compare to get the optimal schedule in more connected networks, which results in larger queue lengths.

VI. Concluding Remarks and Future Research Directions

There is a plethora of directions for future research in this area. We now discuss some of them.

A. Graph Models

As we discussed before, imposing a structure on the wireless network graph considerably impacts the complexity of the scheduling problem OPT-SCHED. In particular, although the scheduling problem OPT-SCHED is NP-Complete and Non-Approximable under general graph models, PTAS can be developed for it in the case of geometric graphs. However, it is still unclear whether low-complexity constant factor distributed solutions can be developed for OPT-SCHED in this case. This remains an interesting problem for future research.

Another open problem is to consider more general graph models as compared to geometric graphs. We believe that results similar to geometric graphs are straightforward to obtain for (r,s)-civilized graphs and disk graphs. Another class of graph models to consider are graphs formed by nodes on the plane, with the probability of an edge between two nodes being some non-increasing function of the distance between them. Note that such a class of graph models includes all geometric graphs. It will be interesting to investigate whether OPT-SCHED is approximable under such a general class of graph models.

B. Distributed Scheduling Schemes

Maximal scheduling and Constant-Time scheduling can both be implemented in a distributed fashion under most commonly used interference models. However, the performance guarantee they provide is rather loose. It is not known whether one can design low-complexity distributed scheduling schemes with better performance guarantees as compared to these two schemes. Recall that a Pick and Compare scheme provides 100% throughput guarantee, but have a very large time complexity. Can one design a low-

complexity scheduling scheme with close to 100% throughput guarantee? Answering this question in the positive seems to be challenging; and answering it in the negative seems even more challenging given that we have had few "negative results" or "lower bounds" in the distributed computing literature.

C. Physical Layer and Interference Models

Previous research in this area has focused on contention matrix based interference models. In fact, explicit distributed solutions have only been developed under the 1-hop and 2-hop interference models. It will be interesting to consider SINR based interference models. More realistic signal propagation models and fading also need to be incorporated into the cross-layer design framework.

Most current works are for single-interface single-antenna type of systems. However, in the future one is likely to see a large number of devices with multiple interfaces and multiple antennas. The main question is whether the insights gained from the design of single-interface single-antenna systems can be applied to design such multi-interface multi-antennae systems. A preliminary investigation of this problem has been conducted in [8].

REFERENCES

[1] D. J. Baker, J. Wieselthier, and A. Ephremides. A Distributed Algorithm for Scheduling the Activation of Links in a Self-organizing Mobile Radio Network. In *IEEE ICC*, pages 2F.6.1–2F.6.5, 1982.

[2] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu. Joint Asynchronous Congestion Control and Distributed Scheduling for Multi-Hop Wireless Networks. In *IEEE INFOCOM*, 2006.

[3] P. Chaporkar, K. Kar, and S. Sarkar. Throughput Guarantees Through Maximal Scheduling in Wireless Networks. 43rd Annual Allerton Conf. on Communications, Control, and Computing, September 2005.

[4] R. L. Cruz and A. V. Santhanam. Optimal routing, link scheduling and power control in multihop wireless networks. In *INFOCOM*, 2003.

[5] H. Balakrishnan et al. The Distance-2 Matching Problem and its Relationship to the MAC-layer Capacity of Ad Hoc Wireless Networks. *IEEE JSAC*, 22(6):1069–1079, August 2004.

[6] B. Hajek and G. Sasaki. Link Scheduling in Polynomial Time. *IEEE Transactions on Information Theory*, 34(5):910–917, September 1988.

[7] C. Joo and N. B. Shroff. Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks. Technical Report, School of ECE, Purdue University, 2006.

[8] X. Lin and S. B. Rasool. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks. Preprint.

[9] X. Lin and S. B. Rasool. Constant-Time Distributed Scheduling Policies for Ad Hoc Wireless Networks. to appear in IEEE CDC, 2006.

[10] X. Lin and N. B. Shroff. Joint Rate Control and Scheduling in Multihop Wireless Networks. In *IEEE CDC*, 2004.

[11] X. Lin and N. B. Shroff. The Impact of Imperfect Scheduling on Cross-Layer Rate Control in Multihop Wireless Networks. In *IEEE INFOCOM*, March 2005.

[12] B. Miller and C. Bisdikian. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice Hall, 2000.

[13] E. Modiano, D. Shah, and G. Zussman. Maximizing Throughput in Wireless Networks via Gossiping. In *ACM Sigmetric*, June 2006.

[14] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *IEEE INFOCOM*, 2005.

[15] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic Power Allocation and Routing for Time Varying Wireless Networks. In *IEEE INFOCOM*, 2003.

[16] M. J. Neely, E. Modiano, and C. E. Rohrs. Power allocation and routing in multibeam satellites with time-varying channels. *IEEE/ACM Trans. Netw.*, 11(1):138–152, 2003.

[17] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Math. Appl., SIAM, Philadelphia, 2000.

[18] S. Sarkar and L. Tassiulas. End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks. *IEEE Trans. on Automatic Control*, 50(9), September 2005.

[19] G. Sharma, R. R. Mazumdar, and N. B. Shroff. Joint Congestion Control and Distributed Scheduling for Throughput Guarantees in Wireless Networks. Technical Report, School of ECE, Purdue University, 2006.

[20] G. Sharma, R. R. Mazumdar, and N. B. Shroff. Maximum Weighted Matching with Interference Constraints. In *IEEE FAWN*, March 2006.

[21] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the Complexity of Scheduling in Wireless Networks. In *ACM MOBICOM*, 2006.

[22] A. L. Stolyar. Maximizing Queueing Network Utility subject to Stability: Greedy Primal-Dual Algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[23] D. Subhadrabandhu, F. Anjum, S. Kannan, and S. Sarkar. Domination and Coverage Guarantees Through Distributed Computation. 43rd Annual Allerton Conf. on Communications, Control, and Computing, September 2005.

[24] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *IEEE INFOCOM*, 1998.

[25] L. Tassiulas and A. Ephremides. Jointly optimal routing and scheduling in packet radio networks. *IEEE Trans. on Info. Theory*, 38(1):165–168, January 1992.

[26] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control*, 37(12):1936–1948, December 1992.

[27] X. Wu and R. Srikant. Bounds on the Capacity Region of Multi-hop Wireless Networks under Distributed Greedy Scheduling. In *IEEE INFOCOM*, 2006.

[28] X. Wu, R. Srikant, and J. R. Perkins. Queue-Length Stability of Maximal Greedy Schedules in Wireless Networks. In *Proc. Workshop on Information Theory and Applications*, February 2006.

[29] L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Trans. on Comm.*, 52(7):1136–1144, July 2004.

[30] Y. Yi and S. Shakkottai. Hop-by-hop Congestion Control over a Wireless Multi-hop Network. In *IEEE INFOCOM*, March 2004.

[31] Y. Yi, G. Veciana, and S. Shakkottai. Learning Contention Patterns and Adapting to Load/Topology Changes in a MAC Scheduling Aglrithm, September 2006.