Wireless Scheduling with Delayed CSI: When Distributed Outperforms Centralized

Matthew Johnston and Eytan Modiano, *Fellow, IEEE*

Abstract—The performance of wireless scheduling algorithms directly depends on the availability and accuracy of channel state information (CSI) at the scheduler. As CSI updates must propagate across the network, they are delayed as they arrive at the controller. In this paper, we analyze the effect that delayed CSI has on the throughput performance of scheduling in wireless networks. By accounting for the delays in CSI as they relate to the network topology, we revisit the comparison between centralized and distributed scheduling. We explore the tradeoff between optimal centralized scheduling using delayed CSI and imperfect distributed scheduling using timely CSI. In particular, we show that under certain conditions distributed scheduling outperforms the optimal centralized scheduling policy, and we characterize the point at which distributed scheduling outperforms centralized scheduling for tree and clique networks. Lastly, we propose a partially distributed scheme that achieves high throughput amidst delayed CSI.

Index Terms—Centralized networks, distributed networks, wireless link scheduling, delayed CSI, markov processes

1 Introduction

To achieve maximum throughput in a wireless network, a centralized controller must opportunistically schedule transmissions based on the current state of time-varying channels [1]. The channel quality corresponding to a communication link is measured by adjacent nodes, and channel state information (CSI) is forwarded across the network to the scheduler.[1] Due to the transmission and propagation delays over wireless links, a non-negligible amount of time is required to collect CSI throughout the network, and in that time the network state may change. As a result, network controllers often must operate with imperfect, delayed information.

There has been extensive work on wireless scheduling [1], [2], [3], with most solutions based on a centralized algorithms requiring global CSI. Centralized scheduling yields high theoretical performance, since the central entity uses current network-wide CSI to compute a globally optimal schedule; however, maintaining current CSI pertaining to the entire network is impractical, due to the latency in acquiring CSI.

An alternative is a distributed approach, in which each node makes an independent transmission decision based on the CSI available locally at that node. Typically, distributed algorithms achieve only a fraction of the throughput of their ideal centralized counterparts, because they make locally optimal decisions [4]. An example distributed scheme is Greedy Maximal Scheduling [5], [6], which is known to achieve only a fraction of the centralized throughput depending on the topology. Distributed scheduling schemes that approach the centralized throughput region are proposed in [7], [8]; however, these schemes also depend on global CSI, and thus would suffer for any delays to that CSI. Additionally, several authors have applied random-access scheduling approaches to maximize throughput in a fully distributed manner [9], [10].

In practice, the available CSI for centralized scheduling is a delayed view of the network state. Furthermore, the delay in CSI is proportional to the distance of each link to the controller, since CSI updates must traverse the network. These delays reduce the attainable throughput of centralized scheduling [11]. In [12], Ying and Shakkottai study throughput optimal scheduling and routing with delayed CSI. In their work, the authors assume arbitrary delays and do not consider the dependence of delay on the network topology. In contrast, by accounting for the relationship between CSI delay and network topology, we are able to effectively compare centralized and distributed scheduling.

In this paper, we propose a new model for delayed CSI, under which nodes have more accurate CSI pertaining to neighboring links, and progressively less accurate CSI for distant links. We show that as a result of the delays in CSI, in some scenarios distributed scheduling algorithms outperform the optimal centralized scheduling scheme. We develop sufficient conditions under which there exists a distributed scheduling policy that outperforms the optimal centralized policy in tree and clique networks, illustrating the impact of topology on this tradeoff, as we provide simulation results to demonstrate the performance in different topologies.

Finally, we consider a partially distributed scheme, in which a network is partitioned into subgraphs and a

1. CSI updates can be piggy-backed on top of data transmissions

- *The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: matthew.r.johnston@boeing.com, modiano@mit.edu.*

Manuscript received 12 Oct. 2016; revised 15 Aug. 2017; accepted 14 Feb. 2018. Date of publication 27 Feb. 2018; date of current version 1 Oct. 2018. (Corresponding author: Matthew Johnston.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2809739

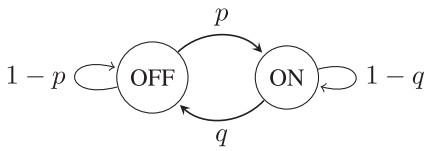


Fig. 1. Markov Chain describing the channel state evolution of each independent channel.

controller is assigned to each subgraph. This approach combines elements from centralized and distributed scheduling in order to trade-off between the effects of delayed CSI and the sub-optimality of local decisions. We show that there exists a regime in which this approach outperforms both the fully distributed and centralized approaches.

The remainder of the paper is outlined as follows. In Section 2, we present the network model and formulate the throughput optimal scheduling problem. In Section 3, we compare centralized and distributed scheduling with delayed CSI. In Sections 4 and 5, we present a detailed analysis comparing centralized and distributed scheduling in tree topologies and clique topologies respectively. In Section 6 we show simulation results confirming our analysis. In Section 7 we extend our approach to consider multiple centralized controllers. Lastly, we conclude in Section 8.

2 Model and Problem Formulation

Consider a network consisting of a set of nodes $\mathcal{N}$, and a set of links $\mathcal{L}$. Time is slotted, and in each slot, a set of links is chosen to transmit. This set of activated links must satisfy an interference constraint. In this work, we use a primary interference model, in which each node is constrained to only activate one neighboring link. In other words, the set of activated links forms a matching.[2]

Each link $l \in \mathcal{L}$ has a time-varying channel state $S_l(t) \in \{0, 1\}$, and is governed by the Markov Chain in Fig. 1. The state of the channel at link l represents the rate at which data can be transmitted over that link. An ON channel ($S_l(t) = 1$) can support a unit throughput (single packet transmission), while transmissions over an OFF channel ($S_l(t) = 0$) fail.

2.1 Delayed Channel State Information

We assume that every node has CSI pertaining to each link, delayed by an amount of time proportional to the distance between the node and the link. Specifically, a node n has k-step delayed information of links in $\mathcal{N}_{k+1}(n)$, where $\mathcal{N}_k(n)$ is the set of links that are k hops away from n. In other words, each node has current CSI pertaining to its adjacent links, 1-hop delayed CSI of its 2-hop neighboring links, and so on, as illustrated in Fig. 2. This models the effect of propagation and transmission delays on the process of collecting CSI.

2.2 Centralized Scheduling

A *centralized scheduling algorithm* consists of a single entity making a global scheduling decision for the entire network. In this work, one node is appointed to be the centralized decision-maker, referred to as the *controller*. The controller has delayed CSI of each link, where the delay is relative to that link's distance from the controller, and makes a scheduling decision based on the delayed CSI. This decision is then broadcasted across the network. Throughout this paper we assume the controller broadcasts the schedule to the other nodes instantaneously. In practice, the decision takes a similar amount of time to propagate from the controller as the time required to gather CSI, which effectively doubles the impact of delay in the CSI. Therefore, the theoretical performance of the centralized scheduling algorithm derived in this work provides an upper bound on the performance achievable in practice.

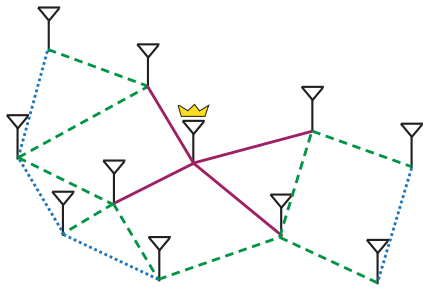


Fig. 2. Delayed CSI structure for centralized scheduling. Controller (denoted by crown) has full CSI of red bold links, one-hop delayed CSI of green dashed links, and two-hop delayed CSI of blue dotted links.

Let $d_r(l)$ be the distance (in hops) of link l from the controller r. The controller has an estimate of $S_l(t)$ based on the delayed CSI. Define the *belief* of a channel to be the probability that a channel is ON given the available CSI at the controller. For link l, the belief $x_l(t)$ is given by

$$x_l(t) = \mathbf{P}\big(S_l(t) = 1 \big| S_l(t - d_r(l))\big). \tag{1}$$

The belief is derived from the k-step transition probabilities of the Markov chain in Fig. 1. Namely,

$$\mathbf{P}\big(S(t) = j \big| S(t-k) = i\big) = p_{ij}^k, \tag{2}$$

where p_{ij}^k is computed as

$$\begin{aligned} p_{00}^k &= \frac{q + p(1-p-q)^k}{p+q}, p_{01}^k = \frac{p - p(1-p-q)^k}{p+q} \\ p_{10}^k &= \frac{q - q(1-p-q)^k}{p+q}, p_{11}^k = \frac{p + q(1-p-q)^k}{p+q}. \end{aligned} \tag{3}$$

Throughout this work, we assume that $1 - p - q \geq 0$, corresponding to channels with "positive memory." The positive memory property ensures that a channel that was ON k slots ago is more likely to be ON at the current time, than a channel that was OFF k slots ago. This allows the transmitter to make efficient scheduling decisions without explicitly obtaining CSI at each time slot. Mathematically, this property is described by the set of inequalities:

$$p_{01}^i \leq p_{01}^j \leq p_{11}^k \leq p_{11}^l \qquad \forall i \leq j \quad \forall l \leq k. \tag{4}$$

As the CSI of a channel grows stale, the probability that the channel is ON is given by the stationary distribution of the chain in Fig. 1, and denoted as π.

$$\lim_{k\to\infty} p_{01}^k = \lim_{k\to\infty} p_{11}^k = \pi = \frac{p}{p+q}. \tag{5}$$

2. A matching is a set of links such that no two links share an endpoint.

In many of the results in this paper, we consider a homogenous model ($p = q$) for ease of analysis. We expect the results presented here to extend easily to the case of ($p \neq q$). Since the objective is to maximize the expected sum-rate throughput, the optimal scheduling decision at each time slot is given by the maximum likelihood (ML) rule, which is to activate the links that are most likely to be ON, i.e., the links with the highest belief. Under the primary interference constraint, a set of links can only be scheduled simultaneously if that set forms a matching. Let $\mathcal{M}$ be the set of all matchings in the network. The maximum expected sum-rate is formulated as

$$\max_{m \in \mathcal{M}} \mathbb{E}\left[\sum_{l \in m} S_l(t) \middle| \{S_l(t - d_r(l))\}_{l \in \mathcal{L}}\right] \tag{6}$$

$$= \max_{m \in \mathcal{M}} \sum_{l \in m} \mathbb{E}\left[S_l(t) \middle| S_l(t - d_r(l))\right] = \max_{m \in \mathcal{M}} \sum_{l \in m} x_l(t). \tag{7}$$

Thus, the optimal schedule is a maximum weighted matching, where the weight of each link is equal to the controller's belief of that link.

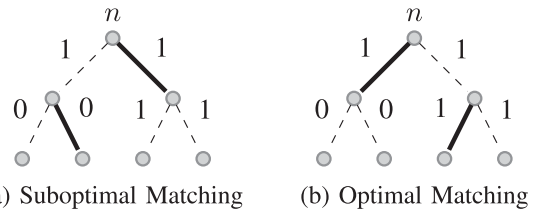
In the above model, we consider an expected throughput metric, as the primary goal in this work is to show that due to the delay in acquiring network state information, centralized schemes, which are commonly assumed to be better than distributed schemes, may suffer due to the information delay. To that end, the consider a non-stochastic model and expected sum-rate objective for ease of exposition. The expected sum-rate is a commonly used metric in the non-stochastic setting.

This work can be extended to a stochastic setting, by adding a weight to each throughput term equal to the queue length at each node. While this would complicate the proofs, we expect to see similar results in the stochastic setting. In the work by Ying and Shakkottai, the authors consider a stochastic model and show that a delay to CSI does reduce the achievable throughput region [12]. Thus, we expect to see a similar point in which the distributed throughput region becomes larger than the centralized throughput region. In related work on controller placement, we considered this stochastic framework and derived the form of a throughput optimal policy [13]. While this paper has a different objective than [13], the formulation there suggests that the current problem can be extended to the in stochastic setting.

2.3 Distributed Scheduling

A *distributed scheduling* algorithm consists of multiple entities making independent decisions without coordination. Each node makes a transmission decision for its neighboring links using only the CSI of adjacent links; hence, the performance of distributed scheduling is unaffected by the delay in CSI. The drawback of such policies is that local scheduling decisions may not be globally optimal.

We consider distributed policies in which transmission decisions are made sequentially to avoid collisions. If a node begins transmission, neighboring nodes detect this transmission and can activate a non-conflicting link rather than an interfering link, in a manner similar to collision avoidance in CSMA/CA. This allows us to focus on the sub-optimality resulting from making a local instead of a global decision, rather than the transmission coordination needed to avoid collisions.[3] In this work we are not concerned with the details of the transmission coordination scheme, but rather the local optimization aspect of this problem.



(a) Suboptimal Matching (b) Optimal Matching

Fig. 3. Example network: All links are labelled by their channel state at the current time. Bold links represent activated links.

As mentioned above, the drawback of distributed scheduling is that local decisions can be suboptimal. For example, in Fig. 3, node n can choose to schedule either of its neighboring links; if it schedules its right child link, then the total sum rate of the resulting schedule is 1, as in Fig. 3a, whereas scheduling the left link results in a sum rate of 2, as in Fig. 3b. In a distributed framework, node n is unaware of the state of the rest of the network, so it makes an arbitrary decision resulting in a throughput loss.

2.4 Partially Distributed Scheduling

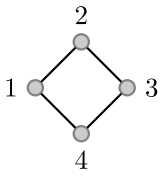
A third class of scheduling algorithms combines elements of centralized and distributed scheduling. These algorithms are referred to as *partially distributed scheduling algorithms*. A partially distributed approach divides the network into multiple control regions, and assigns a controller to schedule the links in each region. Each controller has delayed CSI pertaining to the links in its control region, and no CSI pertaining to links in other regions. This allows for scheduling with fresher information than a purely centralized scheme, and less local sub-optimality than a fully distributed scheme. These policies are explored in Section 7.

3 Centralized versus Distributed Scheduling

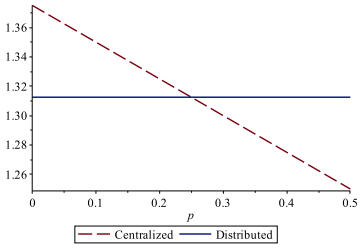
In the previous section, we introduced two primary classes of scheduling policies: distributed and centralized policies. It is known that a centralized scheme using perfect CSI outperforms distributed schemes, due to the aforementioned loss of efficiency in localized decisions. However, these results ignore the effect of delays in collecting CSI. In this section, we revisit the comparison between centralized and distributed scheduling. We show that for sufficiently large CSI delays, there exist distributed policies that perform at least as well as the optimal centralized policy.

As an example, consider the four node network in Fig. 4a, and a symmetric channel state model satisfying $p = q$. Without loss of generality, assume node 1 is the controller. In a centralized scheduling scheme, node 1 chooses a schedule based on current CSI for links $(1, 2)$ and $(1, 4)$, and 1-hop delayed CSI for links $(2, 3)$ and $(3, 4)$. The resulting expected throughput is computed by first conditioning on the state of the links adjacent to the controller, and then

3. Alternative transmission coordination schemes are also possible based on RTS/CTS exchanges [14].



(a) Topology.



(b) Expected throughput as a function of channel transition probability p.

Fig. 4. Four-node ring network example.

on the state of the other two links.

$$\begin{aligned} C(p) = {} & \mathbf{P}(S_{(1,2)}=0, S_{(1,4)}=0)\mathbb{E}[\text{thpt}|S_{(1,2)}=0, S_{(1,4)}=0] \\ & + \mathbf{P}(S_{(1,2)}=1, S_{(1,4)}=1)\mathbb{E}[\text{thpt}|S_{(1,2)}=1, S_{(1,4)}=1] \\ & + \mathbf{P}(S_{(1,2)} \neq S_{(1,4)})\mathbb{E}[\text{thpt}|S_{(1,2)} \neq S_{(1,4)}] \end{aligned} \tag{8}$$

$$\begin{aligned} C(p) &= \frac{1}{4}\left(\frac{3}{4}(1-p)+\frac{1}{4}p\right)+\frac{1}{2}\cdot\frac{3}{2}+\frac{1}{4}\left(1+\frac{3}{4}(1-p)+\frac{1}{4}p\right) \\ &= \frac{11}{8}-\frac{1}{4}p. \end{aligned} \tag{9}$$

The above derivation follows from expanding the expectation terms as follows.

$$\begin{aligned} &\mathbb{E}[\text{thpt}|S_{(1,2)}=0, S_{(1,4)}=0] = \\ &\sum_{s_1,s_2} E[\text{thpt}|S_{(1,2)(t)}=0, S_{(1,4)}(t)=0, S_{(2,3)}(t-1)=s_1, \\ &\quad S_{(3,4)}(t-1)=s_2] \\ &\quad P(S_{(2,3)}(t-1)=s_1, S_{(3,4)}(t-1)=s_2) \\ &= P[S_{(3,4)}(t)=1|S_{(3,4)}(t-1)=0]\left(\frac{1}{4}\right) \\ &\quad + P[S_{(3,4)}(t)=1|S_{(3,4)}(t-1)=1]\left(\frac{1}{4}\right) \\ &\quad + P[S_{(2,3)}(t)=1|S_{(2,3)}(t-1)=1]\left(\frac{1}{4}\right) \\ &\quad + P[S_{(3,4)}(t)=1|S_{(3,4)}(t-1)=1]\left(\frac{1}{4}\right) \\ &= \frac{3}{4}(1-p)+\frac{1}{4}p. \end{aligned} \tag{10}$$

The other terms expand in a similar manner. Now consider a distributed schedule, in which node 1 makes a scheduling decision based on the state of adjacent links $(1,2)$ and $(1,4)$. After this decision is made, node 3 makes a non-conflicting decision based on the state of links $(3,2)$ and $(3,4)$. The resulting expected throughput is given by conditioning on the state of the links adjacent to node 1, and computing throughput accounting for the randomness of the state of the links adjacent to node 3.

$$D = \frac{1}{4}\cdot\frac{3}{4}+\frac{3}{4}\cdot\frac{3}{2}=\frac{21}{16}. \tag{11}$$

The expected throughput for centralized and distributed scheduling in Eqs. (9) and (11) is plotted in Fig. 4b. As the channel transition probability p increases, the memory in the channel decreases, and the expected throughput of a centralized scheduler decreases. The distributed scheduler, on the other hand, is unaffected by the channel transition probability, as it only uses non-delayed local CSI. For channel transition probabilities $p \geq \frac{1}{4}$, distributed scheduling outperforms centralized scheduling over this network.

Next, we extend this result to general topologies. The throughput degradation of the centralized scheme is a function of the memory in the channel state process. Let μ be a metric reflecting this memory. In the case of a two-state Markov chain, we define

$$\mu \triangleq 1-p-q. \tag{12}$$

Note that μ is the second eigenvalue of the state transition matrix for the two-state Markov chain, and thus represents the rate at which the chain converges to its steady state distribution [15].

Theorem 1. *For a fixed steady-state probability π, there exists a threshold μ^* such that if $\mu \leq \mu^*$, there exists a distributed scheduling policy that performs at least as well as the optimal centralized scheduling policy.*

In order to prove Theorem 1, we present several intermediate results pertaining to the expected sum-rate throughput of both the centralized and distributed schemes.

Lemma 1. *For a fixed steady-state probability π, and state transition probabilities p and $q = \frac{\pi}{1-\pi}p$, the expected sum-rate of any distributed policy is independent of the channel memory μ.*

Proof. Since distributed policies are restricted to only use CSI of neighboring links, which is available to each node without delay, the expected sum-rate of a distributed policy only depends on the steady-state probability that links are ON. For fixed π, the expected sum-rate of the distributed policy is constant. □

Lemma 2. *The expected sum-rate of the optimal centralized policy is greater than or equal to that of any distributed policy when $\mu = 1$.*

Proof. When $\mu = 1$, there is full memory in the channel state process, i.e., $p = 0$, and $q = 0$. Thus the centralized policy has perfect CSI throughout the network, and activates the globally optimal schedule. □

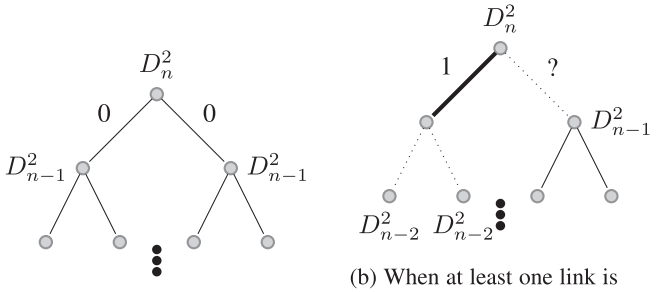
Lemma 3. *There exists a distributed policy with sum rate greater than or equal to the sum rate of the optimal centralized policy when $\mu = 0$.*

The proof of Lemma 3 follows by showing that when $\mu = 0$, a centralized policy only has CSI pertaining to the links adjacent to the controller. Thus, one can construct a distributed policy that returns the same schedule as the centralized policy. The complete proof is given in the Appendix.

Lemma 4. *Let $C(p,q)$ be the sum-rate of the optimal centralized algorithm as a function of the channel transition probabilities p and q. For a fixed value of π, $C(p,q)$ is monotonically increasing in $\mu = 1-p-q$.*

The proof of Lemma 4 is given in the Appendix.

Proof of Theorem 1. Let $C(\mu)$ be the expected sum-rate throughput of the optimal centralized algorithm as a function of the memory in the channel. This theorem is



(a) When both adjacent links are OFF, neither are scheduled. Expected throughput is $2D_{n-1}^2$.

(b) When at least one link is ON, it is scheduled. Dotted links cannot be activated, so expected throughput is $1 + D_{n-1}^2 + 2D_{n-2}^2$.

Fig. 5. Recursive distributed scheduling over binary trees.

proved by showing that there exists a distributed policy with expected sum-rate $D(\pi)$, such that the relationship between $C(\mu)$ and $D(\pi)$ is similar to that in Fig. 4b for fixed π[4]. Since $C(\mu)$ is monotonically increasing in μ (Lemma 4), with $C(1) \geq D(\pi)$ (Lemma 2) , and $C(0) \leq D(\pi)$ (Lemma 3), and $D(\pi)$ is constant over μ by Lemma 1 for fixed π, then $C(\mu)$ must intersect $D(\pi)$, and this intersection occurs at μ^* for some $0 \leq \mu^* \leq 1$. □ □

Theorem 1 claims the existence of a threshold μ^*, such that for $\mu \leq \mu^*$, distributed scheduling performs at least as well as the optimal centralized scheduler. The value of μ^* depends on the topology, and in general, this threshold is difficult to compute. In the following, we characterize the value of the threshold for various network topologies.

4 Tree Topologies

In this section, we characterize the expected throughput over networks with tree topologies. The acyclic nature of these graphs makes them amenable to analysis. We focus on rooted trees, such that one node is the root and every other node has a *depth* equal to the distance from the root. Furthermore, for any node v, the nodes that are connected to v but have depth greater than v are referred to as *children* of v. If u is a child of v, then v is the *parent* of u. This familial nomenclature is standard in the graph-theoretic literature [16], and simplifies description of the algorithms over tree networks. A *complete* k-ary tree of depth n is a tree such that each node of depth less than n has k children, and the nodes at depth n are *leaf* nodes, i.e., they have no children. This section focuses on symmetric channel models such that $p = q$ to simplify the analysis, but the results are easily extended to asymmetric channels as well.

4.1 Distributed Scheduling on Tree Networks

Consider applying the distributed scheduling algorithm over a complete k-ary tree of depth n, where transmission priorities are assigned in order of node depth (lower depth has higher priority). The root node first makes a decision for its neighboring links. Then, the children of the root attempt to activate one of their child links, if this activation does not conflict with their parent's decision. Consequently, the average sum rate can be written recursively. Let D_n^k be the average sum rate of the distributed algorithm over a complete k-ary tree of depth n.

Proposition 1. *Let D_n be the average sum rate of the distributed algorithm over a complete k-ary tree of depth n. The average sum-rate is computed recursively as*

$$D_n^k = \left(\frac{1}{2}\right)^k \cdot kD_{n-1}^k + \left(1 - \left(\frac{1}{2}\right)^k\right)\left(1 + (k-1)D_{n-1}^k + kD_{n-2}^k\right). \tag{13}$$

Proposition 1 follows by conditioning on the state of adjacent links to the root of the tree. If all adjacent links to the root are OFF, then the throughput is given by the throughput over the k subtrees of depth $n-1$, illustrated in Fig. 5a. If at least one adjacent link is ON, as in Fig. 5b, then that link is scheduled, and neighboring links cannot be scheduled.

A closed-form expression is obtained by solving the above recursion.

$$D_n^k = \frac{\left(2^k - 1\right)}{\left(k2^k + 2^k - 1\right)(k-1)\left(2^{k+1} - 1\right)}\Bigg(1 - 2^k(k+1) + k^{n+1}(2^{k+1} - 1) + (1+k)(2^k - 1)\left[- \left(1 - (\frac{1}{2})^k\right)\right]^n\Bigg). \tag{14}$$

To determine the asymptotic per-link throughput, we divide Eq. (14) by the number of links in a k-ary tree, $\frac{k^{n+1}-1}{k-1} - 1$. Taking a limit as n grows large,

$$\lim_{n\to\infty} \frac{D_n^k}{\frac{k^{n+1}-1}{k-1} - 1} = \lim_{n\to\infty} \frac{(k-1)D_n^k}{k^{n+1} - k} = \frac{2^k - 1}{2^k - 1 + k \cdot 2^k}. \tag{15}$$

Since for large k, $2^k >> 1$, this limit is approximately equal to $\frac{1}{k+1}$. Intuitively, each node can only activate one neighboring link, and each node has $k+1$ neighbors.

4.2 Centralized Scheduling on Tree Topologies

Throughout this section, we assume the controller is located at the root of the tree. The optimal centralized policy schedules a maximum weight matching over the network, where the weight of each link is the belief given the delayed CSI. For tree networks, the maximum-weight matching is the solution to a dynamic programming (DP) problem. Consider a node $v \in \mathcal{N}$. Let $g_1(v)$ be the maximum weight matching on the subtree rooted at v, assuming that v activates one of its child links. Let $g_2(v)$ be the maximum weight matching on the subtree rooted at v assuming that v cannot activate a child link, due to interference from the parent of v. Let $r \in \mathcal{N}$ be the controller (root of the tree), and $d_r(v)$ be the distance of node v from r. Let $\mathrm{child}(v)$ be the set of children to node v. Let the delayed CSI at the controller for link (u, v) be $s(u, v)$. The DP formulation for the weight of the optimal max-weight matching $g^*(v)$ is given by

$$g^*(v) = \max(g_1(v), g_2(v)) \tag{16}$$

$$g_2(v) = \sum_{u \in \mathrm{child}(v)} g^*(u) \tag{17}$$

4. Fig. 4b presents throughput as a decreasing function of p, where in this theorem we have an increasing function of μ

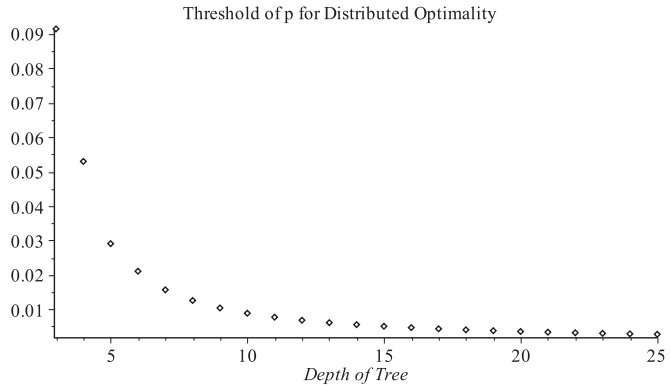


Fig. 6. Threshold value of $p^*(n)$ such that for $p > p^*(n)$, distributed scheduling outperforms centralized scheduling on n-level, binary tree.

$$g_1(v) = \max_{u\in\text{child}(v)} \left(p_{s(u,v),1}^{d_r(v)} + g_2(u) + \sum_{n\in\text{child}(v)\setminus u} g^*(n) \right) \tag{18}$$

$$= \max_{u\in\text{child}(v)} \left(p_{s(u,v),1}^{d_r(v)} + g_2(u) + g_2(v) - g^*(u) \right)$$

$$= g_2(v) + \max_{u\in\text{child}(v)} (p_{s(u,v),1}^{d_r(v)} + g_2(u) - g^*(u)), \tag{19}$$

where Eq. (18) follows from Eq. (17). While Eqs. (16), (17), (18), (19) give the optimal centralized schedule for a specific observation of delayed CSI, computing the average sum rate requires taking an expectation over the delayed CSI. For larger trees, this analysis becomes difficult; however, a recursive strategy can be used to bound the expected solution to the DP.

Let C_n^k be the average sum rate of the centralized algorithm over a complete k-ary tree of depth n, when the root node is chosen to be the controller. The following result analytically bounds the expected throughput of the optimal centralized scheduler recursively.

Proposition 2. *For a complete k-ary tree of depth n, the expected sum-rate throughput of the optimal centralized controller is bounded recursively as:*

$$C_n^k \le k\left(1-\left(\frac{1}{2}\right)^k\right)(1-2p)C_{n-1}^k + k^2\left(\frac{1}{2}\right)^k(1-2p)^2C_{n-2}^k$$
$$+ \left(1-\left(\frac{1}{2}\right)^k\right) + k\left(1-\left(\frac{1}{2}\right)^k\right)p\left(\frac{k^n-k}{k^2-1}+1\right) \tag{20}$$
$$+ k^2\left(\frac{1}{2}\right)^k 2p(1-p)\left(\frac{k^n-k}{k^2-1}+1\right).$$

Proposition 2 is proven by bounding the effect of delay on centralized scheduling, and writing the expression for expected throughput by conditioning on the possible state of the links adjacent to the root, for which the optimal decision is computed via the DP in Eqs. (16), (17), (18), (19). Solving the recursion in Eq. (20) yields a closed-form upper bound on the expected sum-rate throughput achievable by a centralized scheduler. The complete proof is given in the Appendix.

Consider the case of a binary tree. The limiting ratio of the centralized throughput to the number of links in the tree (for $p > 0$) is given by

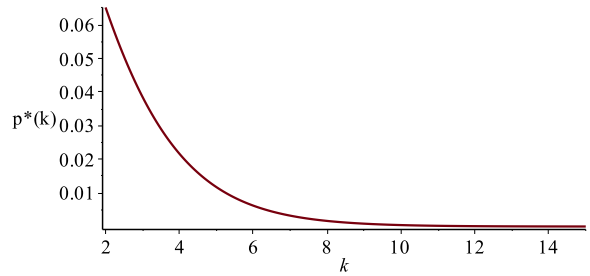


Fig. 7. Threshold value of $p^*(k)$ such that for $p > p^*(k)$, distributed scheduling outperforms centralized scheduling on 2-level, k-ary tree.

$$\lim_{n\to\infty} \frac{C_n^2}{2^{n+1}-2} = \frac{1}{6}. \tag{21}$$

This follows from observing that one third of these links are scheduled (size of a maximum cardinality matching) and they will be in the ON state with probability $\frac{1}{2}$. Hence, the limiting per-link throughput is $\frac{1}{6}$. Note that this value is independent of p, since as n grows large, infinitely many nodes are sufficiently far from the root such that the controller has no knowledge of their current state. Under distributed scheduling Eq. (15), the achievable throughput is $\frac{3}{11}$. Therefore, as the network grows large, distributed outperforms centralized scheduling.

The threshold p^* beyond which distributed scheduling is optimal, is computed by combining Eqs. (20) and (13). Fig. 6 plots p^* as a function of n. Note that as n gets large, this threshold approaches zero, implying that distributed is always better than centralized in large networks.

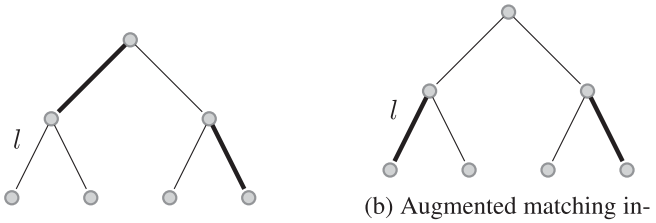
In Fig. 7, we plot the value of p^* for a k-ary tree of depth 2, as a function of k. As k grows large, p^* approaches zero, implying that distributed once again outperforms the optimal centralized approach. Unlike the results in Fig. 6, those in Fig. 7 represent a scenario where the diameter of the network is not increasing. In this case, the CSI is not becoming more delayed, but rather the distributed approach is improving, as there are fewer local sub-optimalities in a wide tree.

4.3 On Distributed Optimality

In the above analysis of a distributed priority scheduler over tree networks, nodes attempt transmission in increasing order of depth. Interestingly, for tree networks, there exists an ordering of transmission priorities such that the distributed policy is optimal, i.e., returns a schedule of maximum weight, and therefore always performs at least as well as the optimal centralized scheduler.

Theorem 2. *There exists an optimal distributed algorithm on tree networks that obtains an expected sum-rate equal to that of a centralized scheduler with perfect information.*

Proof. Consider the policy that gives priority to the leaves of the network. If a link adjacent to a leaf is ON, without loss of generality, there exists a maximum matching containing that link. Assume the optimal matching did *not* include this ON link. A new matching is constructed by adding the leaf link, and removing the link which interferes with it, as illustrated in Fig. 8. Since the new link is adjacent to a leaf, at most one interferer exists in the matching. Thus, the augmented matching is also optimal. Therefore, it is always optimal to include an ON leaf link



(a) Maximum matching 1. (b) Augmented matching including l.

Fig. 8. Example Matchings. If link l is required to be in the matching, there exists a new maximal matching including l.

in the optimal matching. The links interfering with that leaf cannot be activated, and the algorithm recurses. In conclusion, assigning priorities in order of highest depth to lowest depth results in a maximum matching. □

While Theorem 2 shows that there exists an optimal priority assignment, it does not hold for general topologies. Thus, we use the results in Section 4.1 to compare the cost of missed scheduling opportunities to the cost of scheduling with delayed information.

5 Clique Topologies

Next, we consider fully-connected mesh networks (i.e., clique topologies), in which each pair of nodes is connected. Compared to the tree networks of Section 4, mesh networks have a much smaller diameter, resulting in the centralized approach having access to fresher CSI.

5.1 Centralized Scheduling

Consider a fully-connected network where the channel state at each link is independent and identically distributed according to the Markov chain in Fig. 1. In this network, an arbitrary node is chosen as the controller; the choice of controller does not affect throughput due to the network symmetry. In an N-node mesh, the controller is connected to each other node, such that the controller has full information on $N-1$ links, and one-hop delayed information for the other $\frac{(n-1)(n-2)}{2}$ links.

The average sum-rate attainable by a centralized controller is upper bounded by that of a maximum cardinality matching consisting of ON links (links with belief greater than the steady state probability). The probability of this event increases with the size of the network; consequently, this bound becomes tight as the network size increases. If the controller finds such a matching, the expected sum-rate is given by

$$C_n^{UB} = 1 + \left\lfloor \frac{n-2}{2} \right\rfloor (1-q), \tag{22}$$

where q is the transition probability from ON to OFF, and $\lfloor \frac{n-2}{2} \rfloor$ is the size of the maximum cardinality matching in the graph that remains after a link emanating from the controller has been included in the matching.

5.2 Distributed Scheduling

Next, we apply the distributed scheme to a clique topology. The distributed algorithm operates as follows: a node transmits over a randomly chosen ON neighboring link, if one exists, and otherwise does not transmit. Then the next node repeats this process, only considering ON links which do not interfere with any previously scheduled links.

The average achievable sum-rate of this algorithm is computed recursively as follows. The first node to transmit has a probability $1-(1-\pi)^{n-1}$ of having an adjacent link in the ON state, where π is the steady state probability defined in Eq. (5). If there exists an ON link, the two nodes adjacent to that link cannot activate any other links, so the next node schedules over an $n-2$ node clique. On the other hand, if no neighboring links are ON, then no links are activated, and the next node schedules over an $n-1$ node clique. The sum-rate is lower bounded by assuming that the next node to transmit *always* schedules over an $n-2$ node clique, regardless of whether or not an ON link was found. This technique restricts the space of potential matchings which can be activated, and thus results in a lower bound on expected throughput.

$$D_n = (1-(1-\pi)^{n-1})(1+D_{n-2}) + (1-\pi)^{n-1} D_{n-1} \tag{23}$$

$$\geq (1-(1-\pi)^{n-1}) + D_{n-2}. \tag{24}$$

Eq (24) yields a recursion which is solved to lower bound the average sum-rate of the distributed priority scheduler.

$$\begin{aligned} D_n \geq{}& \frac{1}{2}\pi(-1)^n + \frac{\pi}{2} + \frac{(1-\pi)^{n+1}}{\pi(2-\pi)} - \frac{\pi(3-2\pi)(-1)^n}{8-4\pi} \\ &+ \frac{1}{2}(n+1) - \frac{2\pi^2+\pi+2}{4\pi}. \end{aligned} \tag{25}$$

In the case where $p=q$ ($\pi=\frac{1}{2}$), this equation simplifies to

$$D_n \geq \frac{1}{12}(-1)^n - \frac{3}{4} + \frac{2}{3}\left(\frac{1}{2}\right)^n + \frac{n}{2}. \tag{26}$$

As n increases, the expected fraction of nodes with an ON neighboring link tends to 1, implying that this bound is also asymptotically tight.

5.3 Comparison

The bounds in Eqs. (26) and (22) combine to give a bound on p^*, the value of the transition probability (for a symmetric channel) after which there exists a distributed policy that performs at least as well as the optimal centralized policy. For n even, the bound is given by

$$p^* \leq \frac{\frac{4}{3}(1-(\frac{1}{2})^n)}{n-2}. \tag{27}$$

Similarly, for odd values of n, combining Eqs. (22) and (26) yields

$$p^* \leq \frac{\frac{4}{3}(\frac{1}{2}-(\frac{1}{2})^n)}{n-3}. \tag{28}$$

Clearly, as n grows large, the distributed algorithm outperforms the optimal centralized scheduler for a wider range of channel transition probabilities p, since the upper bound goes to 0.

6 Simulation Results

In this section, the performance of the distributed policy is compared to the performance of a centralized controller through simulation. For distributed scheduling, node

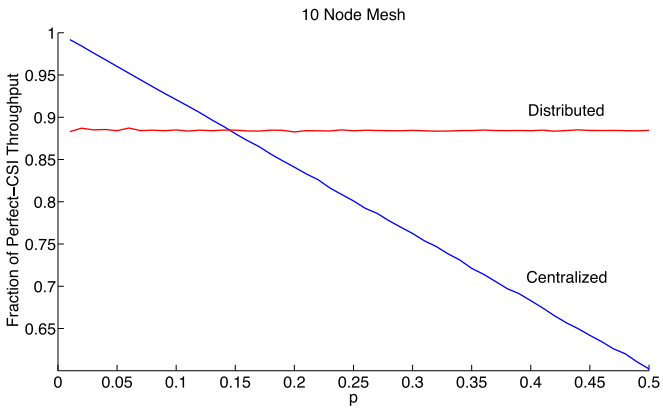


Fig. 9. The fraction of the perfect-CSI throughput obtained as a function of p for 10-node clique topology, over a horizon of 100,000 time slots.

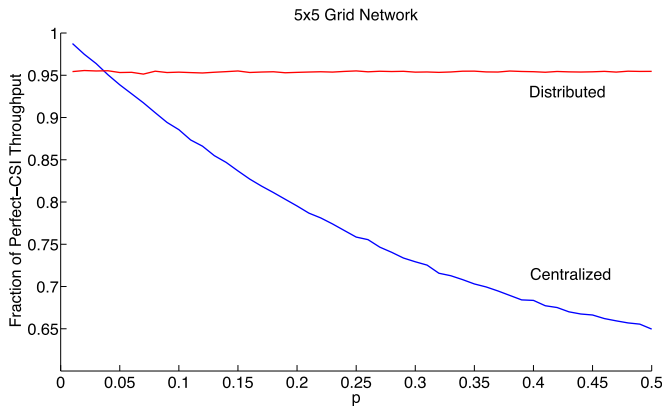


Fig. 10. The fraction of the perfect-CSI throughput obtained as a function of p for a 5 x 5 grid network, over a horizon of 100,000 time slots.

priorities are assigned in reverse order of degree. For each network, we simulate decisions over 100,000 time slots. To begin, consider a 10-node, fully-connected network. The average sum-rate throughput as a fraction of the perfect-CSI throughput[5] is plotted as a function of the channel state transition probability p in Fig. 9. In Fig. 10, the simulation is applied to a five-by-five grid network, where the centralized controller is located at the central-most node. These results show that for p small, i.e., channels with high degrees of memory, a purely centralized controller is optimal. As p increases, eventually distributed scheduling outperforms the optimal centralized scheme.

In Fig. 9, we see that at $p \geq .16$, the distributed algorithm outperforms the centralized algorithm. Recall the bound on p^* found in Eq. (27) for cliques shows that the theoretical bound on p^* is $p^* \leq .1665$. In this case, the theoretical bound agrees closely with the observed simulation results. Additionally, comparing the results for the 5x5 grid in Fig. 10 with the clique in Fig. 9, it is evident that the threshold p^* is higher in the mesh network. This is because the information available to the centralized scheduler is less-delayed in the clique than in the grid, where the diameter is larger. This illustrates the effect of the topology on the resulting performance of each scheduling approach.

7 Partially Distributed Scheduling

Up to this point, this paper has focused on comparing the performance of a distributed scheduler with the performance of an optimal centralized scheduler using delayed CSI. For large networks, the delayed CSI causes a reduction in the throughput of the centralized scheduler, as the links far from the controller have channel states largely independent of the available CSI at the controller. A distributed scheme is shown to outperform the centralized scheme in these scenarios; however, distributed policies suffer from the inability to compute a globally optimal schedule. An alternate to fully distributed scheduling is a partially distributed scheme, in which multiple controllers are used to schedule the links in local neighborhoods. In this section, we consider applying a partially distributed scheduling scheme to a binary tree topology, and show that this scheme outperforms both the fully centralized or distributed approaches.

5. This is the throughput attainable by a centralized scheduler with perfect CSI.

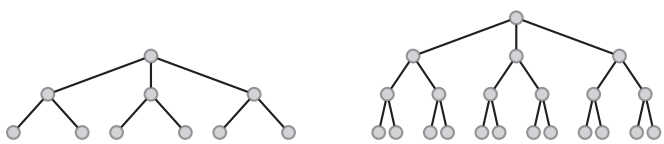
Consider an infinitely-deep binary tree. A single centralized controller has no information pertaining to the majority of the network, and at most attains an average per-link throughput of $\frac{1}{6}$, as shown in Eq. (21). We have shown that a distributed scheme outperforms the centralized scheme in this scenario. Now, we consider a partially distributed scheme to retain some of the benefits of centralized scheduling.

The full binary tree is partitioned into subtrees of depth k, such that each non-leaf node in the subtree has degree 3. Subtrees of depth 2 and 3 are shown in Fig. 11, and an example partitioning is shown in Fig. 12. Observe that there exists a partitioning with subtrees of any depth. Each node in the original binary tree either belongs to one subtree (inner node), or three subtrees (border node). Define a border node to be a node which belongs to three subtrees, as illustrated by the nodes labeled B in Fig. 12.

After the binary tree is partitioned, a controller is placed such that the resulting rooted subtree has the desired depth. Each controller computes a schedule for its partition, using delayed CSI pertaining to the links in the subtree. In order to eliminate inter-subtree interference, multiple controllers cannot activate links adjacent to the same border node simultaneously. Consider an algorithm which disables a set of links, such that a disabled link cannot be activated. We propose a link disabling algorithm with the result that different control regions cannot interfere with one another. Note that this link-disabling scheme is inspired by the work in [17].

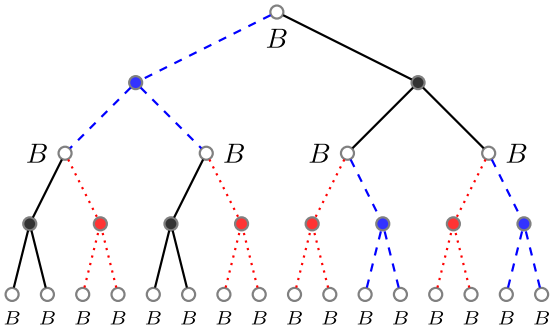
Theorem 3. *Under a primary interference constraint, it is sufficient to disable one link per subtree to completely eliminate inter-subtree interference.*

Proof. To begin, note that inter-subtree interference only occurs at border nodes. Furthermore, each border node has degree three, and each link adjacent to the border node belongs to a different subtree. Based on the visualization of the tree in Fig. 13, the three adjacent links at each border node are labeled as either U, L, or R, denoting whether the

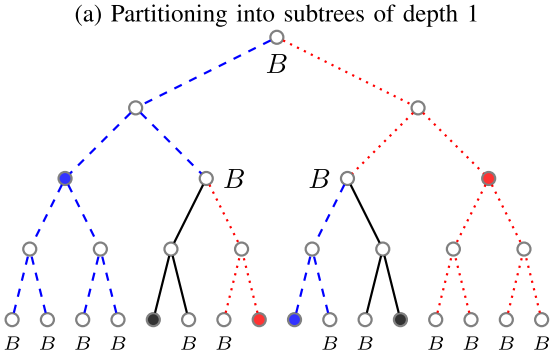


(a) Subtree of depth 2 (b) Subtree of depth 3

Fig. 11. Example subtrees from tree-partitioning algorithm.



(a) Partitioning into subtrees of depth 1



(b) Partitioning into subtrees of depth 2.

Fig. 12. Example partitioning of infinite tree (only first four level's shown). Dashed links, dotted links, and solid links each belong to different subtrees. The solid nodes represent controllers, which are located at the root of each subtree. Nodes labeled with B are border nodes.

link is the upmost link, the left link, or the right link incident to the node. In each subtree, all leaves are border nodes, and a subtree of depth k will have $3 \cdot 2^{k-1}$ leaves. Based on the partitioning scheme, one of the leaf links in each subtree is an L link or an R link, and the remainder of the leaf links will be U links, as illustrated in Fig. 13.

Consider the policy which disables all links labeled L or R. Each border node now has only one adjacent enabled link (the link labeled U), and thus interference is removed between subtrees. Furthermore, since each subtree only has one L or one R leaf link, only one link is disabled per subtree.[6] □

The above scheme for inter-subtree contention resolution disables one link per subtree, leading to a loss in throughput. As the size of the subtree grows, this loss becomes negligible. Fig. 14 shows the per-link throughput as a function of the state transition probability p for various subtree depths. For small values of p, using subtrees of a larger depth yields higher throughput, as the delayed CSI is useful. As p increases and delayed CSI becomes less valuable, it becomes optimal to use less information and add more controllers. Note, a partitioning with subtrees of depth 1 is fully distributed in the sense of this paper, as controllers use only local information with which to make scheduling decisions. Consequently, this plot illustrates a region in which partially distributed scheduling outperforms both fully centralized and fully distributed solutions.

6. While a policy disabling a fraction of the network links is inherently unfair, a practical implementation would enable alternating which links are disabled to meet a fairness requirement.

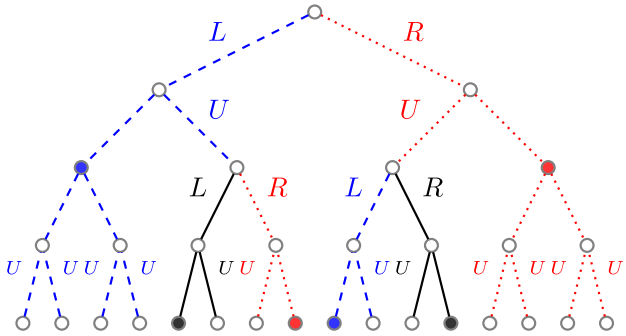


Fig. 13. Illustration of border link labeling scheme.

8 Conclusion

In this paper, we studied the effect of using delayed CSI on the throughput of wireless scheduling. We showed that a centralized scheduling approach suffers from having delayed CSI, and that distributed scheduling can outperform centralized scheduling, despite making suboptimal scheduling decisions. Moreover, approaches combining centralized and distributed scheduling can be used to further improve throughput. Since centralized policies are constrained to using delayed CSI, the location of the controller has an effect on the throughput performance of the scheduling algorithm. Characterizing the effect of controller location on throughput, as well as computing the optimal controller location, are interesting extensions to this work, and are examined in [18].

Appendix

A.1 Proof of Lemma 3

Lemma 3. *There exists a distributed policy with sum rate greater than or equal to the sum rate of the optimal centralized policy when $\mu = 0$.*

Proof. If $\mu = 0$, then the channel transition probabilities p and q satisfy $p = 1 - q$. In this scenario, there is no memory in the channel state process, and thus delayed CSI is useless in predicting the current channel state. To see this, consider the conditional probability of a channel state given the previous channel state.

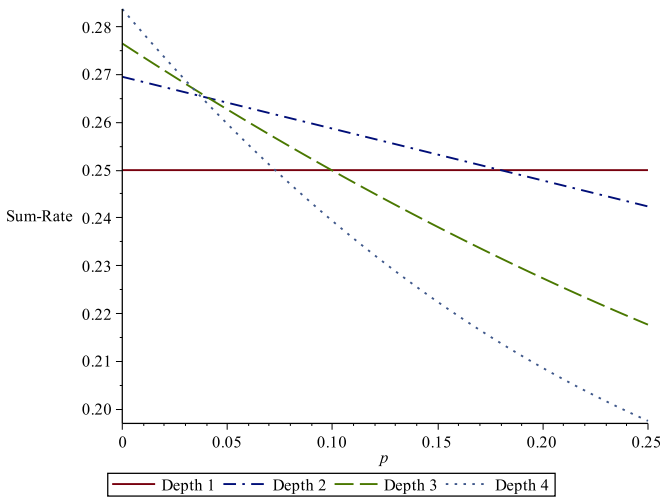


Fig. 14. Per-link throughput of the tree partitioning scheme, plotted as a function of transition probability p for various subtree depths.

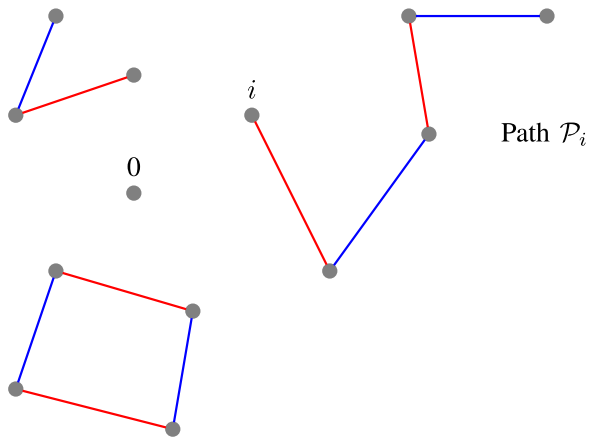


Fig. 15. Example of combining matchings to generate components. Red links and blue links correspond to maximum cardinality matchings M_0 and M_i. The component containing node i is referred to as path $\mathcal{P}_i$.

$$\begin{aligned}&\mathbf{P}(S(t+1)=1|S(t)=0)=p\\&=1-q=\mathbf{P}(S(t+1)=1|S(t)=1)\end{aligned} \tag{29}$$

$$\begin{aligned}&\mathbf{P}(S(t+1)=0|S(t)=0)=1-p\\&=q=\mathbf{P}(S(t+1)=0|S(t)=1).\end{aligned} \tag{30}$$

Thus, when $\mu=0$, the channel state process is IID over time.

Let G be the graph representing the topology of the network with the controller labeled as node 0. Let $\mathcal{N}_0$ be the set of neighbors of node 0, and Δ be the degree of node 0, i.e., $\Delta=|\mathcal{N}_0|$. Let $G_0\subset G$ be the graph obtained by removing the links adjacent to the controller from the network. Similarly, let $G_i\subset G_0$ be the graph obtained by removing all of the links adjacent to node i from G_0. Recall, a matching M of a graph G is any subset of the edges of G such that no two edges share a node. Let M_0 be a maximum (cardinality) matching over G_0, and M_i be a maximum cardinality matching over G_i.

Due to the IID channel process, each adjacent link either has belief 0 or 1, and each non-adjacent link has belief π. Thus, the optimal centralized scheduler operates as follows. The controller observes the state of its adjacent links and chooses a maximum throughput link activation. There are 2^Δ possible state combinations observed by the controller; however, due to the fact that the controller can only activate one adjacent link, the optimal centralized schedule is one of at most $\Delta+1$ matchings. Without loss of generality, when the controller does not activate an adjacent link, it activates matching M_0, and if the controller activates link $(0,i)$ for $i\in\mathcal{N}_0$, then it also activates matching M_i.

Lemma 3 is proved by constructing a distributed policy which activates the same links as the optimal centralized schedule. The $\Delta+1$ potential activations can be computed off-line,[7] and we assume each node knows the set of possible activations. Each node must determine which activation to use in a distributed manner. To accomplish this, node 0 activates the same adjacent link as in the centralized scheme, which is feasible since the centralized controller uses only local CSI in the IID channel state case. Each other node n activates links according to the matching M_0, unless that activation interferes with a neighboring activation. If a conflict occurs, then node 0 must have transmitted according to some other M_i for $i\in\mathcal{N}_0$, and node n detects this conflict, and activates links according to the appropriate M_i. The remainder of the proof explains the details of this distributed algorithm.

Consider the graph composed of the nodes in G and the edges in both M_0 and M_i, as done in [7], labeling edges in M_0 as red and edges in M_i as blue. An example is shown in Fig. 15. The resulting graph consists of multiple connected components, where each component is either a path or a cycle alternating between red and blue links. Note that every component not containing node i has the same number of red and blue links, since both matchings have maximum cardinality. If there exists a connected component in the union of red and blue links not containing node i, and this component has a different number of red and blue links (for example, one more red link than blue link), then the blue matching could be increased in size by switching the links in that component to the red links.

Consider the component including node i, which must be a path since no blue links can be adjacent to node i. Denote this path as $\mathcal{P}_i$. If node 0 schedules link $(0,i)$, then nodes in path $\mathcal{P}_i$ must schedule blue links instead of red links. Since each node detects neighboring transmissions, this can be accomplished in a distributed manner. In all other components, either red links or blue links can be scheduled to obtain maximum throughput, because each component has equal red and blue links, and switching between red and blue links will not affect any other components.

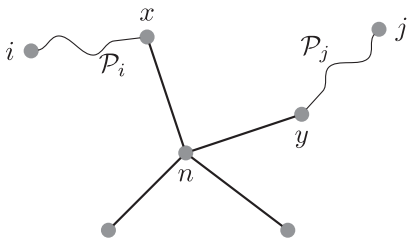
The remaining detail concerns the decision of which of the Δ alternate matchings to use if M_0 conflicts with a neighboring transmission. As explained above, node n is informed of the switch to matching M_i by blue links being activated on path $\mathcal{P}_i$, propagating from node i. If node n does not lie on any path P_i for $i\in\mathcal{N}_0$, then activating links according to matching M_0 never conflicts with any other transmissions. If node n lies on a single path P_i, then upon detecting a conflicting transmission, node n switches to matching M_i. If there is are $i,j\in\mathcal{N}_0$, such that $n\in\mathcal{P}_i$ and $n\in\mathcal{P}_j$, then node n decides between M_i and M_j based on the direction (neighbor) from which the conflicting transmission is detected, as illustrated in Fig. 16a. If $\mathcal{P}_i$ and $\mathcal{P}_j$ are such that the conflicting link at node n is detected from the same neighbor, as in Fig. 16b, then either M_i and M_j can be used. □

A.2 Proof of Lemma 4

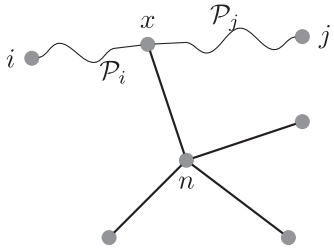
Lemma 4. *Let $C(p,q)$ be the sum-rate of the optimal centralized algorithm as a function of the channel transition probabilities p and q. For a fixed value of π, $C(p,q)$ is monotonically increasing in $\mu=1-p-q$.*

Proof. Let Φ represent the set of feasible schedules (matchings), and $\boldsymbol{\phi}\in\Phi$ be a binary vector, such that ϕ_l indicates whether link l is activated in the schedule. Consider two channel-state distributions, one with transition probabilities p_1 and q_1, and the other with probabilities p_2 and q_2,

7. To compute the set of potential activations, consider the case where only one link adjacent to the controller is ON, as well as the case where all adjacent links are OFF.



(a) Scenario 1. A conflict detected from neighbor x corresponds to matching M_i, and a conflict detected from neighbor y corresponds to matching M_j.



(b) Scenario 2. Node n can activate either according to M_i or M_j if a conflict is detected at neighbor x.

Fig. 16. Abstract representation of a node n's position on multiple conflicting paths.

satisfying $\pi_1 = \pi_2 = \pi$. Furthermore, assume that $\mu_1 \geq \mu_2$. Let $a_{s,1}^k$ ($b_{s,1}^k$) represent the k-step transition probability from s to 1 when the one-step transition probabilities are p_1 and q_1 (p_2 and q_2). Lastly, let $d_r(l)$ be the distance of link l from controller r, and let $\mathbf{S}(t-d_r) = \left[S_l(t-d_r(l))\right]_{l\in\mathcal{L}}$ be the delayed CSI vector, where the lth element is the delayed CSI of link l with delay equal to $d_r(l)$.

Let $\boldsymbol{\phi}^1(\mathbf{s})$ and $\boldsymbol{\phi}^2(\mathbf{s})$ be binary vectors representing the optimal schedules for state $\mathbf{s}$, when the state transition probability is (p_1, q_1) and (p_2, q_2) respectively, with an arbitrary rule for breaking ties, i.e.,

$$\boldsymbol{\phi}^1(\mathbf{s}) = \underset{\boldsymbol{\phi}\in\Phi}{\operatorname{argmax}} \sum_{l\in\mathcal{L}} \phi_l a_{s_l,1}^{d_r(l)} \tag{31}$$

$$\boldsymbol{\phi}^2(\mathbf{s}) = \underset{\boldsymbol{\phi}\in\Phi}{\operatorname{argmax}} \sum_{l\in\mathcal{L}} \phi_l b_{s_l,1}^{d_r(l)}. \tag{32}$$

The expected sum-rate of the centralized scheme is expressed as

$$C(p_1,q_1) = \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^1(\mathbf{s}) a_{s_l,1}^{d_r(l)} \tag{33}$$

$$C(p_2,q_2) = \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^2(\mathbf{s}) b_{s_l,1}^{d_r(l)}. \tag{34}$$

To prove the monotonicity of $C(p,q)$, we show that for all p_1, q_1, p_2, q_2 satisfying $\pi_1 = \pi_2$ and $\mu_1 \geq \mu_2$,

$$C(p_1,q_1) - C(p_2,q_2) \geq 0. \tag{35}$$

The above difference is bounded as follows.

$$\begin{aligned} &C(p_1,q_1) - C(p_2,q_2) \\ &= \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^1(\mathbf{s}) a_{s_l,1}^{d_r(l)} \\ &\quad - \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^2(\mathbf{s}) b_{s_l,1}^{d_r(l)} \end{aligned} \tag{36}$$

$$\geq \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^2(\mathbf{s}) \left(a_{s_l,1}^{d_r(l)} - b_{s_l,1}^{d_r(l)} \right). \tag{37}$$

where the inequality follows from the fact that $\boldsymbol{\phi}^2$ is the maximizing schedule for channel 2, and not channel 1. The proof follows by partitioning the state space into states which result in a specific schedule. Let $S_{\boldsymbol{\phi}} \subset \mathcal{S}$ be the set of states such that $\boldsymbol{\phi}$ is the optimal schedule, i.e.,

$$S_{\boldsymbol{\phi}} = \left\{ \mathbf{s} \in \mathcal{S} | \boldsymbol{\phi}^2(\mathbf{s}) = \boldsymbol{\phi} \right\}. \tag{38}$$

Due to the arbitrary tie-breaking rule in the optimization of $\boldsymbol{\phi}^2(\mathbf{s})$ in (32), each $\mathbf{s}$ belongs to exactly one $S_{\boldsymbol{\phi}}$. In other words, the sets $\{S_{\boldsymbol{\phi}_i}\}_i$ are disjoint, and $\bigcup_{\boldsymbol{\phi}\in\Phi} S_{\boldsymbol{\phi}} = \mathcal{S}$. Therefore, (37) can be rewritten as

$$\begin{aligned} &C(p_1,q_1) - C(p_2,q_2) \\ &\geq \sum_{\boldsymbol{\phi}\in\Phi} \sum_{\mathbf{s}\in\mathcal{S}_{\boldsymbol{\phi}}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l \left(a_{s_l,1}^{d_r(l)} - b_{s_l,1}^{d_r(l)} \right). \end{aligned} \tag{39}$$

The quantity $a_{s_l,1}^{d_r(l)} - b_{s_l,1}^{d_r(l)}$ simplifies using $\mu_i = 1 - p_i - q_i$, and the definition of the k-step transition probability.

$$a_{s_l,1}^{d_r(l)} - b_{s_l,1}^{d_r(l)} = \pi + (s_l - \pi)\mu_1^{d_r(l)} - \pi - (s_l - \pi)\mu_2^{d_r(l)} \tag{40}$$

$$= (s_l - \pi)\left[\mu_1^{d_r(l)} - \mu_2^{d_r(l)}\right]. \tag{41}$$

Combining Eqs. (39) and (41) yields

$$\begin{aligned} C(p_1,q_1) - C(p_2,q_2) &\geq \sum_{\boldsymbol{\phi}\in\Phi} \sum_{\mathbf{s}\in\mathcal{S}_{\boldsymbol{\phi}}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \\ &\quad \cdot \sum_{l\in\mathcal{L}} \phi_l (s_l - \pi)\left[\mu_1^{d_r(l)} - \mu_2^{d_r(l)}\right] \end{aligned} \tag{42}$$

$$\begin{aligned} &= \sum_{\boldsymbol{\phi}\in\Phi} \sum_{\mathbf{s}\in\mathcal{S}_{\boldsymbol{\phi}}} \sum_{l\in\mathcal{L}} \phi_l (s_l - \pi)\left[\mu_1^{d_r(l)} - \mu_2^{d_r(l)}\right] \\ &\quad \cdot \left(\prod_{j\in\mathcal{L}} \mathbf{P}(S_j(t-d_r(j)) = s_j) \right) \end{aligned} \tag{43}$$

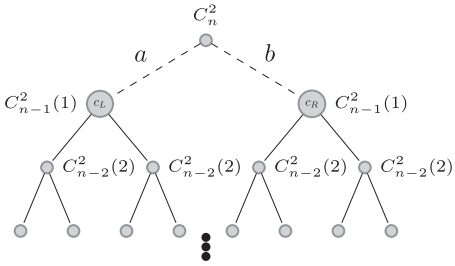
$$\begin{aligned} &= \sum_{\boldsymbol{\phi}\in\Phi} \sum_{l\in\mathcal{L}} \sum_{\mathbf{s}\in\mathcal{S}_{\boldsymbol{\phi}}} \phi_l \pi(1-\pi)(-1)^{1-s_l}\left[\mu_1^{d_r(l)} - \mu_2^{d_r(l)}\right] \\ &\quad \cdot \left(\prod_{j\in\mathcal{L}\setminus l} \mathbf{P}(S_j(t-d_r(j)) = s_j) \right), \end{aligned} \tag{44}$$

where Eq. (43) follows from the independence of the channel state process across links, and Eq. (44) follows from:

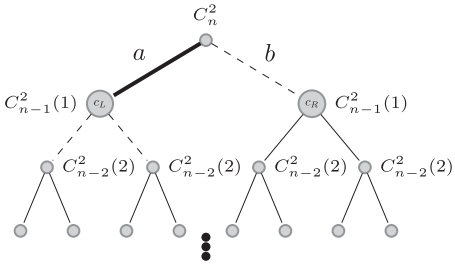
$$\begin{aligned} &\mathbf{P}(S_l(t-d_r(l)) = s_l)(s_l - \pi) \\ &= (\pi s_l + (1-\pi)(1-s_l))(s_l - \pi) \end{aligned} \tag{45}$$

$$= \pi s_l (s_l - \pi) + (1-\pi)(1-s_l)(s_l - \pi) \tag{46}$$

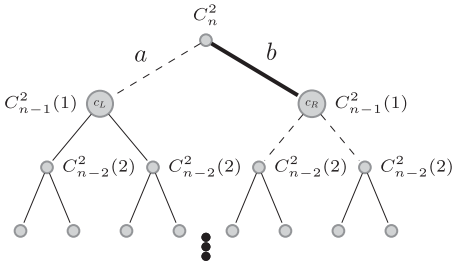
$$= (-1)^{1-s_l}\pi(1-\pi). \tag{47}$$



(a) Link a and link b are not activated. The expected throughput is computed by the maximum expected matching over the solid links, $2C_{n-1}^2$.



(b) If link a is scheduled, the dashed links cannot be scheduled, and the solid links can.



(c) When link b is scheduled, the dashed links cannot be schedules but the solid links can.

Fig. 17. Possible scheduling scenarios for centralized scheduler.

We prove that for any schedule $\phi \in \Phi$ and link $l \in \mathcal{L}$,

$$\sum_{\mathbf{s}\in\mathcal{S}_\phi} \phi_l \pi(1-\pi)(-1)^{1-s_l} \cdot \left[\mu_1^{d_r(l)} - \mu_2^{d_r(l)}\right] \left(\prod_{j\in\mathcal{L}\setminus l} \mathbf{P}(S_j(t-d_r(j)) = s_j)\right) \geq 0. \tag{48}$$

Fix a schedule $\phi \in \Phi$ and link $l \in \mathcal{L}$. The summand in Eq. (48) is non-zero only if $\phi_l = 1$, i.e., the link l is in the schedule ϕ. The summand is negative if and only if $s_l = 0$. Consider a delayed CSI vector $\mathbf{s} \in S_\phi$ such that $s_l = 0$, and the delayed CSI vector $\overline{\mathbf{s}}$ obtained from changing the lth element of $\mathbf{s}$ to 1, i.e., $\overline{s}_j = s_j \; \forall j \neq l$, $\overline{s}_l = 1$. Since $s \in S_\phi$, it follows that $\overline{\mathbf{s}} \in S_\phi$. This is because link l is scheduled under ϕ, and the throughput obtained by scheduling link l is strictly increased in moving from $\mathbf{s}$ to $\overline{\mathbf{s}}$, so the same schedule must remain optimal. Therefore, for every element $\mathbf{s} \in S_\phi$ contributing a negative term to the summation in Eq. (48), there exists another state $\overline{\mathbf{s}} \in S_\phi$ contributing a positive term of equal magnitude, implying that the entire summation must be non-negative. □

A.3 Proof of Proposition 2

Proposition 2 *For a complete k-ary tree of depth n, the expected sum-rate throughput of the optimal centralized controller is bounded recursively as:*

$$\begin{aligned} C_n^k \leq\; & k\left(1-\left(\frac{1}{2}\right)^k\right)(1-2p)C_{n-1}^k + k^2\left(\frac{1}{2}\right)^k(1-2p)^2C_{n-2}^k \\ & + \left(1-\left(\frac{1}{2}\right)^k\right) + k\left(1-\left(\frac{1}{2}\right)^k\right)p\left(\frac{k^n-k}{k^2-1}+1\right) \\ & + k^2\left(\frac{1}{2}\right)^k 2p(1-p)\left(\frac{k^n-k}{k^2-1}+1\right). \end{aligned} \tag{49}$$

Proof. Let $C_n^k(\delta)$ be the expected sum rate of a complete, k-ary subtree of depth n, where the root of that subtree is a distance of δ hops from the controller. Thus, the CSI of a link at depth h in the subtree is delayed by $\delta + h - 1$ time slots.

First, consider the case of $k = 2$. For a binary tree rooted at node v, let c_L and c_R be the left and right children of v respectively. The expected sum-rate is bounded by enumerating the possible states of the links incident to the controller. Label the links adjacent to the root as a and b. If both links a and b are OFF, as in Fig. 17a, then the root schedules neither link, and instead schedules links over the two $n-1$ depth subtrees. If only link a (link b) is ON, then link a (b) will be scheduled (this is optimal following Eq. (4)), and the links adjacent to that link cannot be scheduled, as in Fig. 17b, (17c). If both a and b are ON, then the controller chooses the maximum between the scenarios in Fig. 17b and 17c. Combining these cases leads to an expression for centralized throughput.

$$\begin{aligned} C_n^2 =\; & \frac{1}{4}\cdot 2C_{n-1}^2(1) + 2\cdot\frac{1}{4}(1 + C_{n-1}^2(1) + 2C_{n-2}^2(2)) \\ & + \frac{1}{4}\left(1 + \mathbb{E}\left[\max\big(g_1(c_L)+g_2(c_R), g_2(c_L)+g_1(c_R)\big)\right]\right) \end{aligned} \tag{50}$$

$$\leq \frac{3}{4} + C_{n-1}^2(1) + C_{n-2}^2(2) + \frac{1}{4}\mathbb{E}\left[g_1(c_L)+g_1(c_R)\right] \tag{51}$$

$$= \frac{3}{4} + \frac{3}{2}C_{n-1}^2(1) + C_{n-2}^2(2), \tag{52}$$

where $g_1(\cdot)$ and $g_2(\cdot)$ are defined in Eqs. (17) and (19). The bound in Eq. (51) follows from the fact that $g_1(u) \geq g_2(u)$ for any node $u \in \mathcal{N}$. The same technique can be applied to k-ary trees to bound C_n^k.

$$\begin{aligned} C_n^k \leq\; & k\left(\frac{1}{2}\right)^k(1 + kC_{n-2}^k(2) + (k-1)C_{n-1}^k(1)) \\ & + \left(\frac{1}{2}\right)^k kC_{n-1}^1(1) + \sum_{n=2}^{k}\binom{k}{n}\left(\frac{1}{2}\right)^k\left(1 + kC_{n-1}^k(1)\right) \end{aligned} \tag{53}$$

$$= 1 - \left(\frac{1}{2}\right)^k + k\left(1-\left(\frac{1}{2}\right)^k\right)C_{n-1}^k(1) + k^2\left(\frac{1}{2}\right)^k C_{n-2}^k(2). \tag{54}$$

In order to get a recursive expression for C_k^2, we also need to bound $C_n^k(\delta)$ in terms of $C_n^k(0)$. Note that a complete k-ary tree of depth n has $\frac{k^{n+1}-k}{k-1}$ links. Let $\phi_l(\mathbf{s})$ be an indicator variable equal to 1 if and only if link l is activated in the optimal schedule when the delayed CSI of the network is given by $\mathbf{s}$. Hence,

$$\phi = \operatorname*{argmax}_{\theta\in\Phi} \sum_{s\in\mathcal{S}} \mathbf{P}(S(t-d_r) = \mathbf{s}) \sum_{l\in\mathcal{L}} \theta_l(\mathbf{s}) p_{s_l,1}^{d_r(l)}. \tag{55}$$

Similarly, let $\phi_l^\delta(\mathbf{s})$ be an indicator variable equal to 1 if and only if link l is activated in the optimal schedule

when the CSI is further delayed by δ slots. Applying Eq. (55), the centralized sum rates are expressed as

$$\begin{aligned} C_n^k(\delta) &= \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^\delta(\mathbf{s}) p_{s_l,1}^{d_r(l)+\delta} \\ &= (1-2p)^\delta \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^\delta(\mathbf{s}) p_{s_l,1}^{d_r(l)} \\ &\quad + \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^\delta(\mathbf{s}) p_{0,1}^\delta \end{aligned} \tag{56}$$

$$\begin{aligned} &\le (1-2p)^\delta \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l(\mathbf{s}) p_{s_l,1}^{d_r(l)} \\ &\quad + \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{P}(\mathbf{S}(t-d_r)=\mathbf{s}) \sum_{l\in\mathcal{L}} \phi_l^\delta(\mathbf{s}) p_{0,1}^\delta \end{aligned} \tag{57}$$

$$= (1-2p)^\delta C_n^k(0) + p_{0,1}^\delta \mathbb{E}[\#\text{ of Links Activated}] \tag{58}$$

$$\le (1-2p)^\delta C_n^k(0) + p_{0,1}^\delta \left\lceil \frac{1}{k+1} \mathbb{E}[\#\text{ of Links}] \right\rceil \tag{59}$$

$$\le (1-2p)^\delta C_n^k(0) + p_{0,1}^\delta \left(\frac{1}{k+1} \mathbb{E}[\#\text{ of Links}] + 1 \right) \tag{60}$$

$$= (1-2p)^\delta C_n^k(0) + p_{0,1}^\delta \left(\frac{k^{n+1}-k}{k^2-1} + 1 \right). \tag{61}$$

In the above, Eq. (57) follows from Eq. (55). Combining the bound in Eq. (54) with that in Eq. (61), yields

$$\begin{aligned} C_n^k &\le k\left(1-\left(\frac{1}{2}\right)^k\right)(1-2p)C_{n-1}^k + k^2\left(\frac{1}{2}\right)^k (1-2p)^2 C_{n-2}^k \\ &\quad + \left(1-\left(\frac{1}{2}\right)^k\right) + k\left(1-\left(\frac{1}{2}\right)^k\right) p\left(\frac{k^n-k}{k^2-1}+1\right) \\ &\quad + k^2\left(\frac{1}{2}\right)^k 2p(1-p)\left(\frac{k^n-k}{k^2-1}+1\right) \end{aligned} \tag{62}$$

Inequality Eq. (20) can be solved to yield a closed form upper bound on the centralized sum-rate for large trees. □

ACKNOWLEDGMENTS

This work was supported by NSF Grant CNS-1217048 and ONR Grant N00014-12-1-0064. A preliminary version of this work was presented at ISIT 2015.

REFERENCES

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

[2] M. 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, pp. 89–103, Jan. 2005.

[3] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[4] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2005, pp. 1804–1814.

[5] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Adv. Appl. Probability*, vol. 38, pp. 505–521, 2006.

[6] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1132–1145, Aug. 2009.

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

[8] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," *ACM SIGMETRICS Perform. Evaluation Rev.*, vol. 35, no. 1, pp. 313–324, 2007.

[9] A. Proutiere, Y. Yi, and M. Chiang, "Throughput of random access without message passing," in *Proc. 42nd Annu. Conf. Inf. Sci. Syst.*, 2008, pp. 509–514.

[10] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[11] K. Kar, X. Luo, and S. Sarkar, "Throughput-optimal scheduling in multichannel access point networks under infrequent channel measurements," *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, pp. 2619–2629, Jul. 2008.

[12] L. Ying and S. Shakkottai, "On throughput optimality with delayed network-state information," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5116–5132, Aug. 2011.

[13] M. Johnston and E. Modiano, "Controller placement in wireless networks with delayed CSI," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1775–1788, Jun. 2017.

[14] P. Karn, "MACA-A new channel access method for packet radio," in *Proc. ARRL/CRRL Amateur Radio 9th Comput. Netw. Conf.*, 1990, pp. 134–140.

[15] R. G. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge, U.K.: Cambridge Univ Press, 2013.

[16] J. L. Gross and J. Yellen, *Graph Theory and its Applications*. Boca Raton, FL, USA: CRC Press, 2005.

[17] S. Sarkar and S. Ray, "Arbitrary throughput versus complexity tradeoffs in wireless networks using graph partitioning," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2307–2323, Nov. 2008.

[18] M. Johnston and E. Modiano, "Controller placement for maximum throughput under delayed CSI," in *Proc. 13th Int. Symp. IEEE Model. Optim. Mobile Ad Hoc, Wireless Netw.*, 2015, pp. 521–528.

Matthew Johnston received the BS degree in electrical engineering and computer science from the University of California, Berkeley, in 2008 and the MS and PhD degrees, both in electrical engineering, from the Massachusetts Institute of Technology, in 2010 and 2015, respectively. Since 2015, he has been a research scientist with Boeing Research & Technology. His research is on communication networks and protocols with an emphasis on network design, network control, and network robustness.

Eytan Modiano received the BS degree in electrical engineering and computer science from the University of Connecticut at Storrs, in 1986 and the MS and PhD degrees, both in electrical engineering from the University of Maryland, College Park, Maryland, in 1989 and 1992, respectively. He was a Naval research laboratory fellow between 1987 and 1992 and a National Research Council post doctoral fellow during 1992-1993. Between 1993 and 1999, he was with MIT Lincoln Laboratory. Since 1999, he has been on the faculty with MIT, where he is a professor and an associate department head in the Department of Aeronautics and Astronautics, and an associate director of the Laboratory for Information and Decision Systems (LIDS). His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks. He is the co-recipient of the MobiHoc 2016 Best Paper Award, the Wiopt 2013 Best Paper Award, and the Sigmetrics 2006 Best Paper Award. He is the editor-in-chief for the *IEEE/ACM Transactions on Networking*, and served as an associate editor for the *IEEE Transactions on Information Theory* and the *IEEE/ACM Transactions on Networking*. He was the technical program co-chair for IEEE Wiopt 2006, IEEE Infocom 2007, ACM MobiHoc 2007, and DRCN 2015. He is a fellow of the IEEE and an associate fellow of the AIAA, and served on the IEEE Fellows Committee.