MIMO Link Scheduling for Interference Suppression in Dense Wireless Networks

Luis Miguel Cortés-Peña
Government Communications Systems Division
Harris Corporation
Melbourne, FL 32919
cortes@gatech.edu

Douglas M. Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0765
doug.blough@ece.gatech.edu

***Abstract*—This paper addresses the problem of fair scheduling of MIMO links in dense wireless network deployments where interference suppression is carried out via linear MIMO processing to optimize network performance. We formulate and solve an optimal MIMO link scheduling problem, where the goal is to maximize aggregate throughput while meeting a specified fairness criterion. Evaluations in a modified version of ns-3 show that our MIMO link scheduler more than doubles the performance of 802.11n, while achieving a fairness index of 90–95%.**

I. Introduction

Interference in dense wireless networks is a major problem. These environments include unplanned deployments, e.g. apartment buildings and commercial districts, as well as planned deployments, e.g. access points of an enterprise WLAN. Recently, the use of linear MIMO techniques for interference suppression has been considered to optimize performance in this setting [11]. In this work, we refer to linear MIMO techniques for interference suppression as *MIMO interference cancellation*, or more succintly MIMO IC.

We address the problem of high-performance and fair scheduling of MIMO links in dense wireless networks. We briefly describe methods that allow efficient collection of channel state information and calculation of beamforming weights at the transmitters and combining weights at the receivers (jointly referred to as *MIMO weights*). We then formulate a MIMO link scheduling optimization problem, where the objective is to maximize aggregate throughput while meeting a specified fairness criterion. We provide an approximately optimal solution to a single instance of the scheduling problem and we also provide an adaptive scheduler whose performance converges to the optimal value over time.

Performance evaluations carried out in a modified version of ns-3 show that our time-fair MIMO link scheduling approach can achieve a fairness index of 90–95% while simultaneously achieving more than double the performance of 802.11n, which performs spatial multiplexing but not MIMO IC. Results also show that our time-fair scheduler is within 10% of the performance of a greedy throughput-maximizing scheduler, which has very poor fairness.

II. Network Model

We consider dense deployments of overlapping single-hop networks assigned to a small number of orthogonal channels. MIMO IC is applied within each orthogonal channel independently. We say that two nodes are within communication range if one of the nodes can send a single stream of data at the lowest data rate and the other node can, with high probability, receive and decode this data in the absence of interference.

Channel state information (CSI) from a given node is measured by receiving a sounding packet from that node. A *sounding packet* is a packet containing training symbols in its preamble so that the receiver can estimate all channel dimensions. Therefore, the only interference that can be considered for MIMO IC at a node is that coming from interferers within its communication range. When evaluating performance in Section VI, however, we account for all interference.

We assume that channels do not change rapidly so that MIMO weights calculated at one time can be reused for some period of time before they must be recalculated. This is consistent with the scenarios we consider, in which the access points (APs) are in fixed locations and cover environments like an office, a home, or a coffee house, where users are mobile but often stay in one location for a moderate amount of time in between movements. Detecting when channel states have changed sufficiently to necessitate a new round of measurements is beyond the scope of this paper. Herein, we simply assume that this is done periodically and that measurements remain valid in between these measurement times.

We assume that APs covering neighboring areas can cooperate with each other to schedule transmissions and suppress interference. With products from companies such as Aerohive Networks, APs already cooperate to perform tasks such as channel selection and transmission power control [1]. These technologies could easily be extended to incorporate the type of cooperation proposed herein. Alternatively, many enterprise wireless configurations employ centralized network controllers, which connect to APs through a wired network and could be used to facilitate AP cooperation. Finally, we assume that APs are loosely synchronized. There are a variety of ways in which this can be achieved. For example, [11] had the APs run NTP for this purpose.

III. MIMO Basics

The capabilities of MIMO links on which we primarily focus are spatial multiplexing (SM) and MIMO interference cancellation (IC). With SM only, a single MIMO link in the absence of interference with n_r antennas at the receiver and n_t

antennas at the transmitter can support up to $n = \min(n_r, n_t)$ streams. With MIMO IC but no SM, the same MIMO link can support a single stream on its link and cancel interference from and to other interfering links. When both SM and MIMO IC are used, the performance improvement can be larger than when only one of these capabilities is used [15]. However, there exists a trade-off between these capabilities in that the number of spatially multiplexed streams plus the number of interfering streams cancelled cannot exceed the number of antenna elements on a node.

When considering MIMO IC, it is important to understand the relationship between MIMO weights and interference. Interference caused by a transmitter on a node attempting to receive from a different transmitter is a function of the beamforming weights at the interfering transmitter, the channel between the interfering transmitter and the interfered-with receiver, and the combining weights at the interfered-with receiver. Therefore, the MIMO weights are interdependent, complicating their calculation. Additionally, because the interference is a function of the corresponding channel, an algorithm that computes the MIMO weights to perform MIMO IC requires CSI between every pair of interfering nodes.

IV. MIMO Link Scheduling Framework

To provide some context for our MIMO link scheduling approach, we describe a framework within which MIMO IC can be carried out in dense wireless networks. This framework provides the basis for our ns-3 implementation, which we used to perform the evaluations described in Section VI.

A. MIMO Weight Computation

We use an iterative algorithm for computing beamforming and combining weights, but limit the weight computation overhead by stopping after 10 iterations. It was shown in [6] that iterative algorithms achieve 70% higher sum rate than non-iterative methods in high-interference scenarios for 8-link networks. An assigned AP, called the Worker AP, collects CSI from the links and computes MIMO weights. The functions of the Worker AP could instead be assigned to a centralized network controller if one exists. This approach works well for scenarios with two to about six APs operating on one channel, which covers many practical cases. To avoid performing the expensive MIMO weight computation procedure too frequently, we reuse link sets for which MIMO weights have already been computed, as much as possible.

B. CSI Measurement

To optimize performance for a given set of interfering MIMO links, the nodes must collect all CSI. This includes the channels between every transmitter and their corresponding receivers and between every transmitter and all other receivers with which they interfere. If any combination of links (one link per AP) is allowed to be selected at a given time, the channels between every pair of interfering nodes must be measured. If there are A APs and C clients per AP and every node is within the communication range of every other node, then there are $A + AC$ nodes and approximately $(A + AC)^2 = A^2 + 2AC + (AC)^2$ measurements to collect. With $A = 5$ APs and $C = 10$ clients per AP, more than 2,500 CSI measurements are needed.

We propose a novel approach, called *Consistent AP Orientation with Channel Symmetrization*, or CAPOCS, to reduce the number of channel measurements and CSI communication overhead. Note that the number of channel measurements is dominated by the $(AC)^2$ term. The $(AC)^2$ term corresponds to each client measuring its channel with every other client. Since channel measurements are only needed on actual links and between nodes that interfere, which are transmitter-receiver pairs, we propose to coordinate active APs so that they are either all transmitting (downlink direction) or all receiving (uplink direction) in one slot. This prevents one AP from activating a downlink and another AP from activating an uplink in the same time slot. In this situation, clients do not interfere with each other, because they are consistently receivers or transmitters at one time. As a result, clients do not need to measure channels with other clients and the dominant term in the channel measurements expression disappears. Additionally, APs do not need to measure channels with other APs. What is left is to measure channels for every AP-client link, which requires A^2C measurements. In the above example, this produces more than an order of magnitude reduction in channel measurements, from more than 2,500 down to 250.

C. AP Cooperation

We aggregate as many packets as can fit within a fixed duration τ_{data} and have the receivers simultaneously acknowledge the packets using a BlockAck. This differs from the aggregate packet mechanism in 802.11n in that the packets in 802.11n are aggregated up to a maximum size [3], whereas we aggregate packets up to a maximum duration. To avoid having to compute and distribute MIMO weights for communicating the BlockAcks, we have the destination nodes use the vector that achieves the highest gain from their combining weights for the current link set, normalized to maximize the transmit power, as their beamforming weights for sending the BlockAck. Similarly, we propose to have the source nodes use the vector that corresponds to the highest gain from their beamforming weights for the current link set as their combining weights for receiving the BlockAcks. This technique of reversing the roles, but reusing the MIMO weights, is commonly used to aid in beamforming weight computation ([6], [9], [13], [14]).

D. High-Level Operation of MIMO Framework

The operational flow of the framework is shown in Figure 1. First, the APs discover each other and choose a Worker AP. The Worker AP then requests CSI from all APs. During this step, each AP takes a turn sending sounding packets and recording CSI for each of its associated clients. After CSI is collected by all APs, they forward their measurements to the Worker AP. The Worker AP then constructs and distributes the first schedule, which has only a single link assigned to each time slot. Optimal MIMO weights for these individual links correspond to the singular value decomposition (SVD) of their channels and can be computed in a single computational round by the APs of the links. While the network begins using this first schedule, the Worker AP computes a better schedule with multiple-link sets and the corresponding MIMO weights based on all received CSI. These computations are more intensive than an SVD computation, which is why the single-link link sets are used while the higher-performing schedule is being

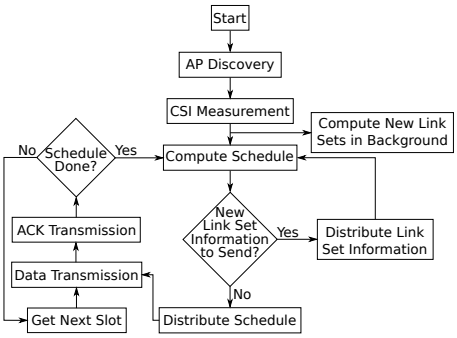

Fig. 1. High-level flow chart of framework.

computed. Details of the scheduling algorithm are given in Section V.

Once the nodes have the schedule, the transmitters of the first time slot transmit for a duration τ_{data}. Then, the receivers wait for a short time before sending their BlockAcks in the reverse channel. If the total time allocated for the BlockAck is τ_{ack}, then the duration of a time slot is $\tau_{\text{slot}} = \tau_{\text{data}} + \tau_{\text{ack}}$. This process repeats for every time slot in the schedule.

Upon completion of the last time slot, the Worker AP regains control and computes a new schedule by considering all link sets for which MIMO weights have been computed. The Worker AP then distributes the MIMO weights for all link sets that were selected in the schedule that have not already been distributed. Finally, the Worker AP distributes the desired schedule and all nodes follow the new schedule until completion, and the process repeats.

V. MIMO Link Scheduling

In this section, we present algorithms for selecting link sets and for generating a throughput-optimal MIMO link schedule, subject to a fairness criterion, from those link sets.

We focus on a set of K half-duplex links, composed of links formed by the APs and their associated clients. Since each AP can have multiple clients, it is possible that some of the links share a node. We assume that CAPOCS is used, so that all links are either downlinks or uplinks. The discussion in this section applies to each of these cases separately.

A. Generating Candidate Link Sets

We use a variation of the algorithm from [7] to compute candidate link sets and their corresponding MIMO weights. Compared to other MIMO weight algorithms, this algorithm has the advantage of jointly optimizing which subset of links to activate, how many streams are carried by each link, and the MIMO weights for all nodes. The algorithm in [7] assumes a set of links where no two links share a node. We make the following changes for our problem setting:

- Using the analysis of [5], we generalize the algorithm from [7] to also consider links that share a node. With this modification, all K links can serve as input to the algorithm. Partway through its execution, however, the algorithm selects only the best performing link among those links that share a node, so that all link sets contain at most one link for each AP.
- Using the analysis of [5], [12], we modify the algorithm to maximize a weighted sum rate. We denote υ_k as the *link weight* assigned to link k. We adjust these link weights to aid in generation of different link sets.

We propose two methods for adjusting the link weights. The first method, called the *At Least Once Method* (ALOM), generates link sets sequentially such that each new link set contains at least one new link and it stops when all links appear at least once. ALOM uses the following link weights as input to the MIMO weight algorithm:

$$\upsilon_k = \frac{1}{1+\delta_k} \text{ for all } k \in \{1, \ldots, K\}, \tag{1}$$

where δ_k is the number of times link k has been included in the link sets previously computed. To ensure that at least one new link appears in the next link set, we square the values of υ_k at each iteration of the MIMO weight algorithm if no new link has yet been included. ALOM achieves a minimal degree of fairness by ensuring that no link is completely starved.

To achieve more sophisticated fairness objectives, we propose a second method, called the *compensating method* (CM). CM first runs the ALOM method to completion to compute an initial set of link sets. Then, the CM procedure tries to aid the scheduler in achieving certain proportions of bandwidth by generating extra link sets that compensate for previously generated link sets. Let $\boldsymbol{b}$ be a vector of K positive elements, where the k^{th} element b_k represents the desired bandwidth portion to allocate to link k, so that $\sum_{k=1}^{K} b_k = 1$. We will see later that different choices of $\boldsymbol{b}$ can be used to achieve different fairness objectives. Assume that a total of N link sets have already been computed. Also, let $\boldsymbol{A}$ be a $K \times N$ matrix, where the element $a_{k,n} \geq 0$ at the k^{th} row and n^{th} column of $\boldsymbol{A}$ contains the data rate of link k in the n^{th} link set, which is set to zero if link k is not active in link set n. The CM procedure sets the link weights as follows:

$$\upsilon_k = \max\left(1 - \frac{\sum_{n=1}^{N} a_{k,n}}{b_k \sum_{i=1}^{K}\sum_{n=1}^{N} a_{i,n}}, 0\right) \tag{2}$$

According to (2), links that meet or exceed their bandwidth proportion from previously computed link sets will have $\upsilon_k = 0$ and thus will not be considered for the current link set.

B. Scheduling Problem Formulation

In this subsection, we present a throughput-optimal MIMO link scheduling problem with fairness constraint.

Let the number of candidate link sets be N. We wish to find an $N \times 1$ column vector $\boldsymbol{x} \geq \boldsymbol{0}$ that satisfies

$$\tfrac{1}{\alpha}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3}$$

where the n^{th} element x_n of vector $\boldsymbol{x}$ denotes the number of times to schedule the n^{th} link set, $\boldsymbol{x} \geq \boldsymbol{0}$ means that each element x_n of $\boldsymbol{x}$ satisfies the inequality $x_n \geq 0$, and $\alpha = \sum_{k=1}^{K}\sum_{n=1}^{N} a_{k,n}$. In (3), we scale $\boldsymbol{A}$ with $\frac{1}{\alpha}$ to maintain numerical stability since the elements of $\boldsymbol{A}$ are potentially large compared to the elements of $\boldsymbol{b}$. Note that the length of the schedule given by $\boldsymbol{x}$ is $\sum_{k=1}^{N} x_k$. We will attempt to minimize schedule length subject to the fairness criterion given by $\boldsymbol{b}$.

Note that (3) can have zero, one, or many solutions, depending on $\boldsymbol{A}$. However, because we initialize the available link sets with the single-link link sets, problem (3) is guaranteed to always have at least one solution. When multiple solutions exist for (3), we choose the solution that minimizes schedule length. The following linear programming problem formally defines the minimum scheduling length problem with fairness contraint:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \sum_{n=1}^{N} x_n \quad \text{subject to } \boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0} \tag{4}$$

Theorem 1 shows that the minimum schedule length problem is equivalent to maximizing aggregate throughput.

Theorem 1: If problem (3) has a solution, then the solution that minimizes the schedule length in (4) is also the solution that maximizes the sum of link rates.

Proof: Let $\boldsymbol{c} = \boldsymbol{Ax} = \alpha \boldsymbol{b}$ with elements c_k for $k \in \{1, \ldots, K\}$. The total rate of link set n is given by $r_n = \sum_{k=1}^{K} a_{k,n}$, and so the relative amount of data that can be sent throughout the schedule is $\tau_{\text{slot}} \sum_{n=1}^{N} r_n x_n$. Therefore, the sum rate is given by

$$\begin{aligned} \text{Sum Rate} &= \frac{\tau_{\text{slot}} \sum_{n=1}^{N} r_n x_n}{\tau_{\text{slot}} \sum_{n=1}^{N} x_n} = \frac{\sum_{k=1}^{K} c_k}{\sum_{n=1}^{N} x_n} \\ &= \frac{\alpha \sum_{k=1}^{K} b_k}{\sum_{n=1}^{N} x_n} = \frac{\alpha}{\sum_{n=1}^{N} x_n}, \end{aligned}$$

since $\sum_{k=1}^{K} b_k = 1$. Therefore, minimizing $\sum_{n=1}^{N} x_n$ also maximizes sum rate. ■

C. Scheduling Algorithm

In this subsection, we present our MIMO link scheduling algorithm, which maximizes aggregate throughput for a given set of candidate link sets, subject to the fairness constraint.

The vector $\boldsymbol{x}$ that solves (4) defines a relative number of times to schedule each link set, the values of which are potentially non-integers. Let $\boldsymbol{s}$ be a vector where the n^{th} element s_n contains the integer number of times that link set n is to be scheduled. To compute the schedule $\boldsymbol{s}$ using $\boldsymbol{x}$, we find a factor β such that

$$\boldsymbol{s} = \text{Round}(\beta \boldsymbol{x}), \tag{5}$$

$$\tau_{\text{slot}} \sum_{n=1}^{N} s_n \approx \tau_{\text{schedule}}, \tag{6}$$

where τ_{schedule} is the desired duration of the schedule.

Because the schedule duration is limited, the scaling function (5) can be non-ideal, causing the actual proportions to deviate from the desired proportions. To compensate for this, we define a history-aware version of (4) that accounts for the history of schedules when computing the new schedule.

Let $\boldsymbol{h}$ be a column vector where the n^{th} element h_n contains the number of times that link set n has been scheduled. The *history-aware version of* (4) can be obtained by replacing $\boldsymbol{b}$ in (4) with $\hat{\boldsymbol{b}} = \boldsymbol{b} - \frac{1}{\alpha}\boldsymbol{Ah}$ and setting α so as to both provide numerical stability and to ensure that all elements of $\hat{\boldsymbol{b}}$ are non-negative. In our simulations, we choose $\alpha = 2(\sum_{k=1}^{K}\sum_{n=1}^{N} a_{k,n})(\sum_{n=1}^{N} h_n + 1)/\min(\boldsymbol{b})$. Intuitively, $\hat{\boldsymbol{b}}$ represents the desired proportions minus those proportions that have already been satisfied. Theorem 1 also applies to the history-aware version of problem (4), and so the solution to this problem also maximizes the sum rate.

To compute the schedule $\boldsymbol{s}$ for the history-aware objective function, we set

$$\boldsymbol{s} = \text{Round}\Big((\boldsymbol{x} + \boldsymbol{h})\,\beta - \boldsymbol{h}\Big)_+, \tag{7}$$

where $(\cdot)_+$ is vector $(\cdot)$ with the negative entries replaced with zeros, and where β is the smallest factor that either satisfies (6) or satisfies a fairness constraint within some small value ϵ.

Figure 2 shows our MIMO link scheduling algorithm. Unless otherwise specified, we assume that the collection of link sets, $\boldsymbol{A}$, which is provided as input to the algorithm is generated by the CM method. To summarize the algorithm's performance, a single execution is approximately optimal due to inaccuracies introduced by scaling and rounding. The inaccuracies are compensated over time, so that performance converges to the optimal value. Optimality means achieving maximum aggregate throughput, for a given collection of link sets and subject to the fairness constraint imposed by $\boldsymbol{b}$. While there is no guarantee that the collection of link sets generated by CM is optimal, we increase the likelihood of getting a good collection by having CM produce a large number of high-performing link sets. In the simulations reported in the next section, we generated $2.5K$ link sets, where K is the number of links, so that, on average, each link appears in 2.5 different link sets. By the nature of the algorithm from [7], each chosen link set is a high-performing one.

Input: $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{h}, \tau_{\text{slot}}, \tau_{\text{schedule}})$
Output: $(\boldsymbol{s}, \boldsymbol{h})$

$\boldsymbol{\phi} = \boldsymbol{0}$; $\boldsymbol{\theta} = \boldsymbol{0}$;
if $\boldsymbol{h} \neq \boldsymbol{0}$ **then**
  Compute $\boldsymbol{x}$ by solving the history-aware version of (4) and compute $\boldsymbol{s}$ using (7);
  $\boldsymbol{\phi} = \boldsymbol{s}$; $\tau_{\text{schedule}} = \tau_{\text{schedule}} - \tau_{\text{slot}} \sum_{n=1}^{N} s_n$;
  Set $\boldsymbol{h} = \boldsymbol{0}$ if $1 - f(\boldsymbol{A}(\boldsymbol{s} + \boldsymbol{h}), \boldsymbol{b}) < \epsilon$;
end if
if $\boldsymbol{h} == \boldsymbol{0}$ **then**
  Compute $\boldsymbol{x}$ by solving (4) and compute $\boldsymbol{s}$ using (5);
  $\boldsymbol{\theta} = \boldsymbol{s}$;
end if
$\boldsymbol{s} = \boldsymbol{\phi} + \boldsymbol{\theta}$; $\boldsymbol{h} = \boldsymbol{h} + \boldsymbol{s}$;
Set $\boldsymbol{h} = \boldsymbol{0}$ if $1 - f(\boldsymbol{Ah}, \boldsymbol{b}) < \epsilon$;
return $(\boldsymbol{s}, \boldsymbol{h})$;

Fig. 2. Pseudocode for MIMO link scheduling algorithm.

D. Achieving Fairness

We show how to define $\boldsymbol{b}$ in order to achieve two fairness criteria for wireless networks.

1) Time-based fairness: Following the ideas of [4], we define the time-based proportions of links as the data rate proportions when each link is allocated an equal number of interference-free time slots. This is the standard notion of time-based fairness in wireless networks, except that it eliminates

interference-induced distortions on data rates. The time-based fairness criterion is achieved by solving problem (4) with $b_k = \rho_k / \sum_{j=1}^{K} \rho_j$ for all $k \in \{1, \ldots, K\}$, where ρ_k is the data rate for link k in the absence of interference when using the optimal singular-value decomposition (SVD) weights. We refer to the instance of our MIMO link scheduling algorithm that uses this definition of fairness as Algorithm *TimeFair*.

2) Rate-based fairness: With rate-based fairness, the goal is to achieve equal average rate across all links. This fairness criterion can be achieved by solving problem (4) with $b_k = 1/K$ for all $k \in \{1, \ldots, K\}$. We refer to the instance of our MIMO link scheduling algorithm that uses this definition of fairness as Algorithm *RateFair*.

VI. Simulation Results

In this section, we present numerical results on our MIMO link scheduling algorithm. For comparison, we also evaluate the performance of 802.11n, which performs MIMO spatial multiplexing (SM) only. We also evaluate a scheduling algorithm, referred to as *GreedyMaxRate*, that attempts to maximize throughput using MIMO IC but with only minimal fairness. This algorithm uses the ALOM candidate link sets, ensuring that every link is activated at least once per scheduling period. Finally, we include an algorithm that we refer to as *NoICuplink*, which performs MIMO IC on downlinks in the same manner as *GreedyMaxRate* but does not perform IC on uplinks (only SM), similar to the approach of [11].

To evaluate fairness, we use the index proposed in [4]:

$$f(\boldsymbol{u}, \boldsymbol{b}) = \frac{1}{\exp\left(\frac{1}{K} \sum_{k=1}^{K} \left| \ln \frac{u_k}{b_k \sum_{j=1}^{K} u_j} \right| \right)}, \qquad (8)$$

where $\boldsymbol{u}$ is actual bandwidth usage, $\boldsymbol{b}$ is desired bandwidth allocation, and (8) takes values between 0 and 1 with $f = 1$ indicating perfect fairness.

A. Simulation Setup

All algorithms were implemented in the ns-3 simulator [2]. We made several changes to ns-3 to perform these evaluations, including adding support for: the matrix-based physical-layer MIMO model ([7], [9], [13]), the Greenfield preamble, sounding packets, and MIMO SM. In the 802.11n simulations, we use the SVD weights, which are optimal for a single link with no interference and we enable the aggregate MAC service data unit (A-MSDU) support in ns-3, which sends aggregate packets of up to 7935 bytes and their corresponding BlockAcks [3].

We account for the overhead of computing the MIMO weights within our simulation by measuring the CPU time consumed by the MIMO weight computation function and scaling it so as to estimate the running time of an AP running at 3 GHz. For the simulation of channel measurements necessary for MIMO link scheduling, we consider the overhead of 30 OFDM subcarriers at all times. Additionally, we assume that CSI and MIMO weights are transmitted uncompressed when collecting CSI and when distributing the MIMO weights, respectively. These assumptions overestimate the overhead of our approach for exchanging CSI and MIMO weights.

We assume that $\tau_{\text{schedule}} = 500$ ms initially. Once the Worker AP finishes computing all link sets and their associated MIMO weights, we set $\tau_{\text{schedule}} = 3$ secs. Additionally, we set the number of link sets generated by the CM procedure to $2.5K$, where K is the number of links in the network. Unless otherwise stated, we set the duration in which packets can be aggregated to $\tau_{\text{data}} = 10$ ms and we set $\tau_{ack} = 210$ μs. We assume that all data packets are UDP packets of 1024 bytes. We also assume: a flat-fading Rayleigh MIMO channel [8], every wireless node has four antenna elements, and the path-loss exponent is three. Finally, we set the fraction of downlink traffic to the total traffic as $p_{\text{downlink}} = 0.6$.

B. Evaluation of CAPOCS

We compare CSI collection overhead when APs can arbitrarily become transmitters or receivers (requiring almost all CSI) to the overhead when CAPOCS is used. We simulate four APs in a line at 50 meter intervals, all operating on the same channel. Each AP has C associated clients that are uniformly distributed within a radius of 80 meters from the AP. These parameters were chosen so as to produce substantial interference among clients and APs across the network.

Figure 3 shows the time to collect CSI as a function of the number of clients per AP (C). This figure validates the analysis of Section IV-B, showing a linear time increase for CAPOCS and a superlinear increase when CAPOCS is not used. When considering the actual time values required for CSI collection, we see that, for up to 6 clients per AP, the collection time with CAPOCS is at most a few hundred milliseconds. Since in the limited-mobility environments we are considering, we expect scheduling periods of 10–30 seconds, this demonstrates that CAPOCS can keep CSI overheads low enough to make the proposed approach practical. On the other hand, without CAPOCS, CSI collection times grow to almost two seconds for 4 APs and 6 clients per AP, and this could have a substantial impact on the performance and practicality of the approach. We do not show a detailed aggregate throughput comparison of CAPOCS' link sets and arbitrary link sets due to space limitations, but we have conducted such tests and the results showed only about a 2% performance loss when limiting link orientations as prescribed by CAPOCS.

C. Evaluation of Scheduling Approaches

1) Under a Controlled Topology: Here, we consider the topology of Figure 4 with fixed distance APs and their associated clients (thereby fixing the signal-to-noise ratio). We vary interference by adjusting the distance between interfering APs.

Figure 5 shows sum goodput as a function of AP separation (x) for an client-to-AP distance (y) of 50 meters, averaged over 100 trials. The results show that the goodput is much greater for the strategies that perform both IC and SM than for 802.11n, which performs SM only. The GreedyMaxRate algorithm achieves a goodput that is approximately three times the goodput of 802.11n at $x = 70$. Figure 5 also shows that GreedyMaxRate achieves a goodput that is almost 35% better than NoICuplink, highlighting the importance of performing IC on both uplinks and downlinks. Despite being subject to significant fairness constraints, Algorithms TimeFair and RateFair have a goodput that is within 8% of Algorithm GreedyMaxRate, which achieves only minimal fairness. TimeFair and RateFair achieve similar throughputs because the time-fair and rate-fair proportions are almost equal due to the symmetry of the simulated topology.

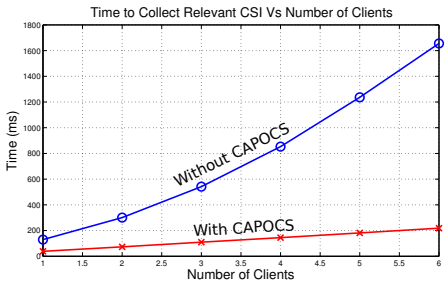

Fig. 3. CSI collection time with/without CAPOCS.

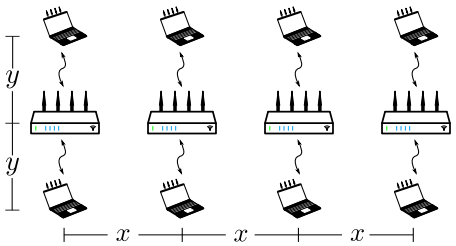

Fig. 4. Controlled topology with four APs.

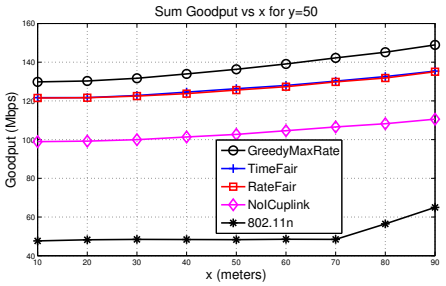

Fig. 5. Sum goodput for topology of Figure 4.

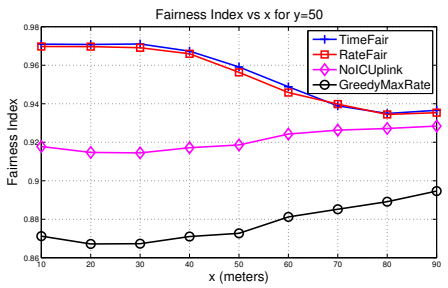

Fig. 6. Avg. fairness index for topology of Fig. 4.

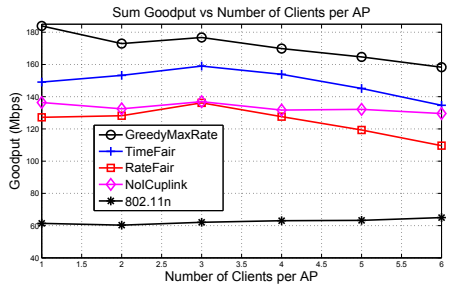

Fig. 7. Sum goodput for four APs in a line.

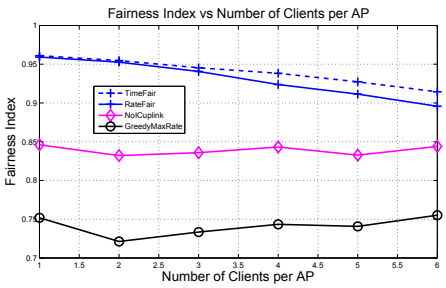

Fig. 8. Avg. fairness index for four APs in a line.

Figure 6 shows the average fairness index for the experiment of Figure 5. Algorithms TimeFair and RateFair achieve a fairness index that is very close to their goals. As expected, Algorithm GreedyMaxRate is the most unfair. Algorithm NoICuplink achieves a higher fairness index than GreedyMaxRate because the uplink, having only single-link link sets, achieves almost perfect time-based fairness.

2) With Random Clients: We now return to the scenario used in the CAPOCS evaluation to evaluate scheduling algorithm performance with randomized clients. As before, we simulate four APs in a line at 50 meter intervals on the same channel and we uniformly distribute C clients with a radius of 80 meters around each AP.

Figure 7 shows the sum goodput versus the number of clients per AP (C), averaged over 50 trials. Again, GreedyMaxRate performs best, achieving a goodput that is as high as 2.8 times that of 802.11n. Algorithm TimeFair has a goodput as much as 2.5 times that of 802.11n and it outperforms Algorithm RateFair by 20–30%. This is consistent with the well known fact that with rate-based fairness, low rate links dominate air time, which substantially reduces overall performance [10]. Figure 8 shows that Algorithm TimeFair achieves excellent fairness in the randomized scenario also.

VII. Conclusions

In this paper, we presented a MIMO link scheduling algorithm for dense wireless networks, which schedules links that use linear MIMO interference suppression to improve performance. The scheduling algorithm was shown to achieve throughput-optimal performance under a specified fairness constraint for a given set of candidate link sets. Simulation results showed that the algorithm can achieve more than double the performance of 802.11n, while simultaneously achieving a fairness index of 90–95%. Future work could consider improvements to the selection process for candidate link sets and consideration of non-linear processing techniques on links.

References

[1] HiveOS. http://www.aerohive.com/products/access-points/products/software-management/hiveos.

[2] The ns-3 network simulator. http://www.nsnam.org.

[3] IEEE std. 802.11-2012. Mar. 2012.

[4] D. Blough, G. Resta, and P. Santi. Interference-aware proportional fairness for multi-rate wireless networks. In *INFOCOM*, 2014. pp. 2733–2741.

[5] S. Christensen, R. Agarwal, E. Carvalho, and J. Cioffi. Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design. *IEEE Trans. Wireless Commun.*, 7(12):4792–4799, Dec. 2008.

[6] L. Cortés-Peña, J. Barry, and D. Blough. The performance loss of unilateral interference cancellation. In *ICC*, 2012. pp. 4181–4186.

[7] L. Cortés-Peña, J. Barry, and D. Blough. Joint optimization of stream allocation and beamforming and combining weights for the MIMO interference channel. In *GLOBECOM*, 2013. pp. 4012–4018.

[8] G. Foschini and M. Gans. On limits of wireless communications in a fading environment when using multiple antennas. *Wireless Personal Commun.*, 6(3):311–335, Mar. 1998.

[9] K. Gomadam, V. Cadambe, and S. Jafar. A distributed numerical approach to interference alignment and applications to wireless interference networks. *IEEE Trans. Inf. Theory*, 57(6):3309–3322, June 2011.

[10] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11 b. In *INFOCOM*, 2003. pp. 836–843.

[11] S. Kumar, D. Cifuentes, S. Gollakota, and D. Katabi. Bringing cross-layer MIMO to today's wireless LANs. In *SIGCOMM*, 2013. pp. 387–398.

[12] F. Negro, S. Shenoy, I. Ghauri, and D. Slock. On the MIMO interference channel. In *Inf. Theory Appl. Workshop*, pages 1–9, Feb. 2010.

[13] S. Peters and R. Heath. Cooperative algorithms for MIMO interference channels. *IEEE Trans. Veh. Technol.*, 60(1):206–218, Jan. 2011.

[14] F. Rashid-Farrokhi, K. Liu, and L. Tassiulas. Transmit beamforming and power control for cellular wireless systems. *IEEE J. Sel. Areas Commun.*, 16(8):1437–1450, Oct. 1998.

[15] R. Srinivasan, D. Blough, and P. Santi. Optimal one-shot stream scheduling for MIMO links in a single collision domain. In *Proc. IEEE SECON*, pages 1–9, June 2009.