Distributed MIMO Interference Cancellation for Interfering Wireless Networks: Protocol and Initial Simulation

Luis Miguel Cortés-Peña, and Douglas M. Blough

***Abstract*—In this report, the problem of interference in dense wireless network deployments is addressed. Two example scenarios are: 1) overlapping basic service sets (OBSSes) in wireless LAN deployments, and 2) interference among close-by femtocells. The proposed approach is to exploit the interference cancellation and spatial multiplexing capabilities of multiple-input multiple-output (MIMO) links to mitigate interference and improve the performance of such networks. Both semi-distributed and fully distributed protocols for 802.11-based wireless networks standard are presented and evaluated. The philosophy of the approach is to minimize modifications to existing protocols, particularly within client-side devices. Thus, modifications are primarily made at the access points (APs). The semi-distributed protocol was fully implemented within the 802.11 package of ns-3 to evaluate the approach. Simulation results with two APs, and with either one client per AP or two clients per AP, show that within 5 seconds of network operation, our protocol increases the goodput on the downlink by about 50%, as compared against a standard 802.11n implementation.**

***Index Terms*—MIMO, multiple antennas, beamforming, combining, overlapping basic service set, dense wireless networks**

I. Introduction

WIRELESS local area networks have exploded in popularity over the past decade. Access points (APs) are being deployed in many stores and in many homes. In dense areas, such as apartment complexes and shopping malls, basic service sets (BSSes) can overlap, reducing the performance of each network. In this paper, we tackle the problem of interference within densely deployed wireless networks. Although this problem can be solved by using different orthogonal channels on each access point (AP), the problem becomes unavoidable when the number of APs exceeds the number of available orthogonal channels. In this report, we focus primarily on integrating our approach within 802.11-based wireless networks. The problem of interfering 802.11 networks is sometimes referred to as the overlapping basic service set (OBSS) problem.

In the literature, proposed solutions to the OBSS problem look for ways to share the medium by careful scheduling [1], by modifications to the network allocation vector (NAV) [2], by modifications to the contention windows [3], or by quality of service enhancements [4]. We propose to tackle the OBSS problem by equipping nodes with multiple antennas, which can be used to perform interference cancellation.

When the transmitter and receiver of a link are equipped with multiple antennas, they form a multiple-input multiple-output (MIMO) link. Two capabilities of MIMO links that allow for increased performance of the overall network include [5]: spatial multiplexing, in which a link supports multiple streams in parallel; and interference cancellation, in which interfering transmitter-receiver pairs cancel interference between one another. With the "n" extension to the 802.11 protocol, wireless nodes are equipped with multiple antennas and exploit the spatial-multiplexing capabilities of MIMO links, supporting throughputs of up to 600 Mbps [6]. Although the 802.11n protocol improves the performance of a single link, overlapping BSSes must still take turns to coexist. In this paper, we propose to use the interference-cancellation capability of MIMO links so that multiple OBSSes can be active simultaneously. To the best of our knowledge this is the first work that exploits the interference cancellation technique of MIMO links to tackle the OBSS problem.

In this paper, we propose both semi-distributed and distributed protocols that enable the use of MIMO interference cancellation techniques. Our design philosophy is to make most of the changes at the access points so as to reduce the burden on the clients as much as possible. Our proposed protocols can use any algorithm for computing the MIMO beamforming and combining weights that cancel interference and support multiple streams on each link. However, in our simulations, we make use of the well-known Maximum Weighted Sum Rate (MWSR) algorithm from [7] for weight calculation. The proposed protocols contain periods for performing channel-state information (CSI) measurements, MIMO-weight computation, data transmission, and acknowledgement (ACK) transmission. Because computing MIMO weights is expensive in terms of overhead, our protocol is biased toward re-using link sets in which MIMO weights are already available. We have implemented the distributed version of our protocol within the 802.11 package of the ns-3 network simulator [8] for two APs. The simulation results for this case show that our approach improves the throughput by about 50% compared to the case where APs have to alternate their transmissions.

This report is organized as follows. In Section II, we state our network model and assumptions. In Section III, we present our semi-distributed protocol. In Section IV, we briefly present our distributed protocol. In Section V, we present simulations results for two OBSSes. Finally, in Section VI, we provide our conclusions and discuss our future work.

II. Network Model and Assumptions

We assume that all nodes operate on the same wireless channel. Also, we assume that a transmission interferes with

all unintended receivers, but the amount of interference is dependent on the MIMO channel and the distance between the transmitter and the unintended receivers. We say that two nodes are within communication range of each other if one of the nodes, taking the role of a transmitter, can send a single stream of data at the lowest data rate with no beamforming and the other node can, with high probability, receive and decode this data in the absence of interference. Additionally, we assume that a unicast packet is retransmitted by the MAC layer until it is successful (indicated by the transmitter receiving an acknowledgement from the receiver), and that all unicast packets will eventually succeed when the transmitter and receiver are within communication range.

We assume that a wireless node can act as either a transmitter or a receiver, but not both, at any given time. In this report, we target scenarios where the number of OBSSes is fairly small, say between two and five. Investigating scalability to larger numbers of OBSSes is future work. Additionally, we assume that the only method in which APs can communicate with each other is through the wireless medium and that APs can communicate with each other either directly or through other APs, but not through other clients. We assume that the set of APs is fixed throughout execution of our protocol; however, we can easily handle the cases where APs join and leave the protocol by re-initializing the protocol states.

We assume that the only method for measuring the CSI from a given node is by receiving a packet containing training symbols, called a *sounding packet*, from that node. Therefore, the only interference that can be considered for interference cancellation at a node is that in which the interferer is within communication range. We assume that the channels are symmetric so that CSI measured at a node can be used as an estimate for the reverse channel; however, the case of asymmetric channels can be handled by executing an extra synchronization step that allows for the channel to be treated as symmetric [9]. With the channels being symmetric, a given node that receives a sounding packet can estimate the channel to the node that transmitted the sounding packet, and can use this information for both the case where the given node takes the role of a transmitter and the case where it takes the role of a receiver.

We also assume that channels do not change rapidly, so that MIMO weights that are calculated at one time can be reused for some period of time before they must be recalculated. In our scenarios, the 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. In these environments, channels are much less dynamic than, for example, in the outdoor cellular environment. Thus, our assumption should hold within these environments. Detecting when channel states have changed sufficiently to necessitate a new round of measurements is beyond the scope of the report. Herein, we simply assume that this is done periodically and that measurements remain valid in between these measurement times.

In this document, we denote $\boldsymbol{A}$ with boldface uppercase letters as an AP labeled "A" and we denote $\boldsymbol{a}_k$ with boldface lowercase letters as the k^{th} client associated to $\boldsymbol{A}$. Figure 1 depicts an example topology of five OBSSes, where the solid lines between two APs represent that these APs are within communication range, and the dashed lines represent an example link set. Given this example link set, we have assumed that all links in the link set operate on the downlink (from AP to client) or on the uplink (from client to AP) so that the dotted lines in Figure 1 represent an example set of the interferences that can be cancelled (from an interferer that is within communication range).

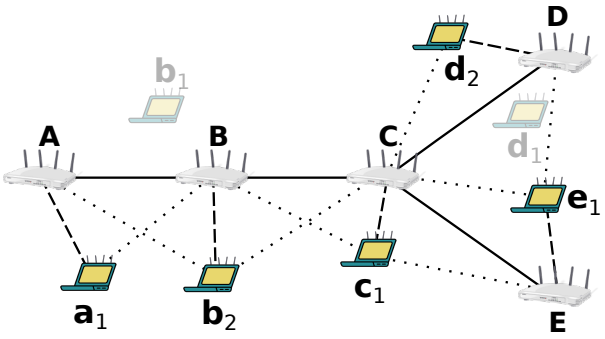

Fig. 1. A five AP example highlighting the AP topology (solid lines) and an example link set (dashed lines) with the interference that can be cancelled (dotted lines).

III. The Proposed Semi-Distributed Protocol

In this section, we describe our proposed semi-distributed protocol. The protocol has the following five phases: AP-discovery phase; CSI-measuring and MIMO-weight-computation phase; link-set-advertisement and synchronization phase; data-transmission phase; and acknowledgement phase. The only phase that is not fully distributed is the CSI-measuring and MIMO-weight-computation phase, where CSI is sent to a designated node to carry out weight computation. This is done, because of the costly nature of weight computation and the large number of messages that would have to be exchanged to perform it in a distributed fashion. Because of our policy of favoring known link sets with previously computed MIMO weights, this phase is not executed in steady state operation unless new clients join the network. Hence, the protocol operates in a fully distributed fashion most of the time. Next, we provide a high-level overview of the semi-distributed protocol before we describe the details of each phase in Subsection III-B to Subsection III-F.

A. High-Level Overview of Protocol

To take advantage of the interference-cancellation and spatial-multiplexing capabilities of MIMO links, participating nodes must set their MIMO weights appropriately. These MIMO weights are dependent on the CSI between every pair of interfering nodes that are active. Because CSI is different between every pair of nodes, different link selections produce different MIMO weights. To prevent the overhead of having to measure CSI and compute the MIMO weights for every link set, our proposed protocol reuses, whenever possible, link sets for which MIMO weights have been previously computed.

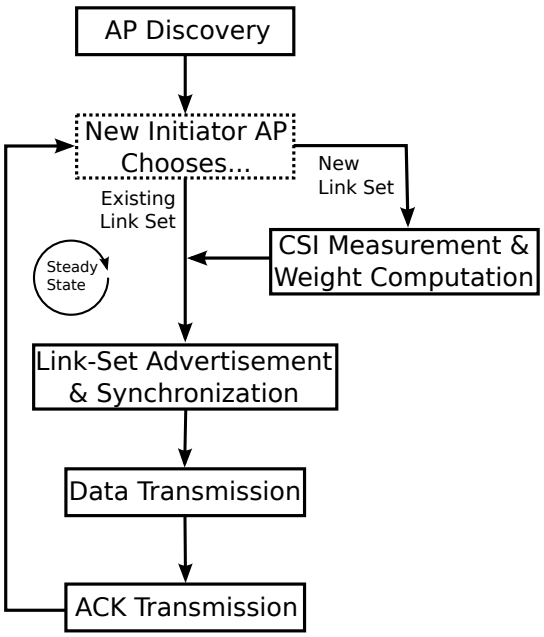

Fig. 2. High level flowchart of proposed protocol. Phases are shown with solid borders.

A high-level flowchart of the various phases in our proposed protocol is illustrated in Figure 2. In our protocol, the first phase is the AP-discovery phase and is executed only once. During this phase, APs construct a tree structure for collecting CSI and distributing MIMO weights, and agree on an AP round-robin order. After the AP-discovery phase ends, the APs begin taking turns, in choosing link sets for all APs, according to the round-robin order. Let us define the *initiator AP* as the AP whose turn it is to choose the link set in a given round. At the beginning of the initiator's turn, the initiator has two choices for link sets: it can suggest a link set for which MIMO weights have previously been computed, or it can suggest the creation of a new link set for which MIMO weights have not been computed. If the initiator AP chooses an existing link set, the initiator AP skips to the link-set-advertisement and synchronization phase.

If, however, a desired link is not within an existing link set, the initiator AP transitions to the CSI-measuring and MIMO-weight-computation phase. During this phase, the initiator AP begins by advertising to other APs that a new link selection is about to take place. Then, each AP chooses its desired client, measures CSI from every participating client possible (those clients that are within the communication range), and forwards this information to its parent in the tree structure until the CSI reaches the root node. Then, the root node computes the MIMO weights and distributes them to its child nodes. The MIMO weights are propagated down the tree until they reach the corresponding APs and clients. Once nodes know their MIMO weights for the current link set, nodes transition to the link-set-advertisement and synchronization phase.

At the beginning of the link-set-advertisement and synchronization phase, the initiator AP sends a synchronization beacon for synchronizing the data transmissions. In the case that the initiator AP chose an existing link set, the beacon also contains the chosen link set. Once nodes are synchronized, the data-transmission phase begins. During this phase the transmitting nodes transmit several packets within a fixed duration. After this duration, the receivers simultaneously acknowledge their received packets through a BlockAck. Once the acknowledgement phase is finished, the next AP in the round-robin order suggests the link set to use in the new round.

During the steady state (see Figure 2), APs have MIMO weights available for their desired clients, and so each AP simply reuses existing link sets. In this case, the only sources of overhead are announcing the link set and synchronizing the data transmission.

Our semi-distributed protocol requires very few changes to the clients. Namely, our protocol requires that the clients be able to perform the following:

- apply a given set of MIMO weights during the data-transmission phase and the acknowledgement-transmission phase;
- store the MIMO weights received for future use; and
- compute MIMO weights for the reverse channel (which are easily derived from the forward channel, explained in Section III-F) when receiving or transmitting the acknowledgment packets.

As we will explain in Section IV, the list of requirements on clients increases for the distributed protocol. Next, we discuss the details of each phase of our semi-distributed protocol.

B. AP Discovery

Initially, and only during the first round of our proposed protocol, the APs go through the AP-discovery phase. During this phase, APs accomplish the following three things:

- Choose a worker AP – APs choose a *worker AP*, which is the AP that receives all CSI measured by the APs and compute the MIMO weights for all nodes participating in the link set. The worker AP can be chosen deterministically. For example, APs can choose the AP with the smallest MAC address.
- Form a tree structure – APs form a tree structure, with the worker AP at the root of the tree. This structure facilitates the collection and distribution of the CSI and MIMO weights during the CSI-measurement and MIMO-weight-distribution phase.
- Create a round-robin order – The worker AP creates an order used by the APs to take turns becoming the initiator AP.

Selecting the worker AP is equivalent to selecting a leader during a leader election protocol. However, leader election protocols for wireless systems typically have very high overhead, assume that nodes have knowledge of neighboring nodes, and/or assume that messages are received reliably [10–14]. Although any leader election protocol can be used in our protocol, we describe a simple strategy in which reliability is achieved by a combination of transmitting unicast packets that require acknowledgements and by overhearing packets.

Our strategy for choosing the worker AP is to have each AP maintain a list of AP-MAC addresses and have them share their lists with neighboring APs. In Figure 3, we

```
1: if (Received AP-MAC list packet) then
2:     Merge received MAC list with local list
3:     Store merged MAC list as new local list
4:     if (Merged list different from received list) then
5:         Queue reply with unicast
6:         Stop the no-reply timer
7:     else if (Merged list different from old local list) then
8:         Delete all broadcast packets with MAC lists
9:         Queue new broadcast packet
10:        Stop the no-reply timer
11:    end if
12:    if (Packet destination is self) then
13:        Send ACK
14:    end if
15: end if
16: if (Channel is idle and have MAC list packet) then
17:    if (Unicast packet is available) then
18:        Dequeue unicast packet
19:    else
20:        Dequeue broadcast packet
21:        Start the no-reply timer
22:    end if
23:    Send packet with most updated MAC list
24: end if
25: if (No-reply timer expires) then
26:    Choose worker AP
27: end if
```

Fig. 3. Pseudocode for handling different events when choosing the worker AP while in the AP-discovery phase.

provide pseudocode for handling the different events that occur while selecting the worker AP during the AP-discovery phase. Initially, an AP's list only contains its own MAC address. Suppose that an AP wishes to form a link with one of its clients. This AP will acquire the channel to broadcast a packet containing its MAC address list. An AP that receives a packet with a MAC list merges its local list with the one in the received packet. If the merged list produces a list that is different from its local list, but is the same as the one received (the receiving AP has no new information for the AP that transmitted the list), then the AP acquires the channel to broadcast this information. However, if the merged local list and the one received are different (the receiving AP has new information for the AP that transmitted the list), it replies back with its new list using a unicast packet but transmitted at the lowest data rate. If an AP has both a unicast and broadcast packet to send in its queue, the unicast packet has priority over the broadcast packet. Also, when queueing a broadcast packet with an AP list, old broadcast packets are removed. Neighboring APs overhearing a unicast packet update their local AP list and possibly unicast or broadcast the new list. The destination AP of a unicast packet replies with an ACK, then updates its local list, and, if necessary, acquires the channel to reply or to broadcast its new local list. Whenever an AP that is itself the AP with lowest MAC address in its local list senses that none of its neighbors have replied with a new list for a given waiting time, it broadcasts its list one final time. If no new response is generated, it declares itself as the worker AP.

The worker AP must now inform all nodes that it has elected itself. The protocol uses this opportunity to set up the tree structure for communicating the CSI, and the round-robin order, which APs use to take turns when selecting the link sets. The worker AP is the root node in this tree structure. To set up the tree structure and round-robin order, the worker AP initializes a count to zero, and gives an order to each AP in the list, assigning itself as the first in the order. The count is used by each of the APs to choose the AP that becomes its parent in the tree structure. The worker AP sends this count and the round-robin order to each one of its neighboring APs using unicast packets that require ACKs. Each AP that receives such a packet increments the count and forwards the information to each of its other neighbor APs using unicast packets. Each AP chooses the neighboring AP with the smallest count as its parent node in the tree structure, and notifies its parent of its selection. APs that do not receive a parent notification label themselves as *leaf APs*.

As an example, consider the network of Figure 1. Suppose that $\boldsymbol{B}$ has the smallest of all AP MAC addresses. Also, suppose that the first AP to acquire the channel is AP $\boldsymbol{E}$ and that all APs wish to join the transmission. In this scenario, $\boldsymbol{E}$ broadcasts an AP-discovery packet containing only its MAC address. AP $\boldsymbol{C}$ receives this packet and adds its own MAC address to the list and replies to $\boldsymbol{E}$ with its new information using a unicast packet. APs $\boldsymbol{B}$, $\boldsymbol{D}$, and $\boldsymbol{E}$ update their local MAC address list with the MAC address of $\boldsymbol{C}$ and $\boldsymbol{E}$ and contend for channel access. Now, suppose that $\boldsymbol{B}$ acquires the channel to reply to $\boldsymbol{C}$ with its own AP-list packet. Then APs $\boldsymbol{A}$ and $\boldsymbol{C}$ receive the packet containing the MAC addresses of $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{E}$. Because the information stored at $\boldsymbol{C}$ is a subset of this new information, $\boldsymbol{C}$ does not reply to $\boldsymbol{B}$; however, it still contends for channel access to broadcast the new information. Lets suppose that at this time, $\boldsymbol{A}$ and $\boldsymbol{D}$ acquire the channel and transmit their AP list using unicast packets to $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively. Then $\boldsymbol{B}$ learns of $\boldsymbol{A}$, while $\boldsymbol{C}$ learns of $\boldsymbol{D}$. Suppose that $\boldsymbol{C}$ acquires the channel and transmits to $\boldsymbol{D}$ (because $\boldsymbol{C}$ has knowledge of $\boldsymbol{B}$ that $\boldsymbol{D}$ is missing) with the MAC information of $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{E}$. At this point, $\boldsymbol{B}$ and $\boldsymbol{E}$ learn of $\boldsymbol{D}$, while $\boldsymbol{D}$ learns of $\boldsymbol{B}$. Because the local lists stored at $\boldsymbol{D}$ and $\boldsymbol{E}$ are a subset of the information received from $\boldsymbol{C}$, then these nodes contend for channel access to broadcast their new AP-list received. AP $\boldsymbol{B}$, however, has knowledge of $\boldsymbol{A}$ and therefore contends for channel access to send this information to $\boldsymbol{C}$. Suppose that $\boldsymbol{E}$ acquires the channel and broadcasts the same information that it overhead from $\boldsymbol{C}$. In this case, $\boldsymbol{C}$ refrains from re-transmitting its AP list. Suppose that $\boldsymbol{B}$ acquires the channel to reply to $\boldsymbol{C}$, then $\boldsymbol{B}$ sends the MAC addresses of APs $\boldsymbol{A}$ through $\boldsymbol{E}$ in the network. Suppose that $\boldsymbol{A}$ and $\boldsymbol{C}$ broadcast their newly acquired information. Then APs $\boldsymbol{D}$ and $\boldsymbol{E}$ learn of $\boldsymbol{A}$ and contend for channel access to broadcast the final AP-list packet. At this point no more updates occur and $\boldsymbol{B}$ becomes the worker AP. AP $\boldsymbol{B}$ must then send to each of its neighbors a count set to zero and the order of all APs. Receiving APs choose their parents and inform their parents

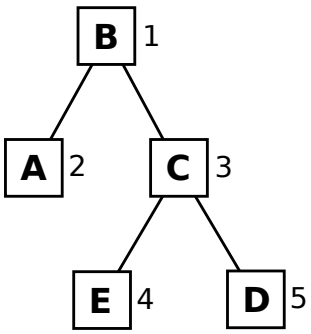

Fig. 4. Tree structure of five AP example showing an example round-robin order as chosen by the worker AP.

that they are children nodes. Figure 4 shows the final tree structure constructed using this procedure and an example order chosen by the worker AP.

Because the worker AP places itself first in the round-robin order, the worker AP becomes the initiator AP during the first round.

C. CSI Measurement and MIMO Weight Computation

The CSI-measurement and MIMO-weight-computation phase is executed in the case that the initiator AP wishes to create a new link set. As we describe in the following, the initiator AP informs other APs of its decision through a special beacon that reduces the interference while measuring CSI.

1) CSI Measurements: At the beginning of the CSI-measurement phase, the initiator AP sends a special beacon containing a contention-free period (CFP) initiation that prevents nodes other than the APs participating in the protocol from acquiring the channel. This beacon contains a flag that informs other APs that it is creating a new link set. Upon receiving this beacon, APs contend for channel access to send a beacon with a CFP duration that equals the duration of the received beacon minus any delay incurred for transmitting its own beacon.

This procedure initiates a CFP so that APs can measure the CSI with low or no interference. Because the MIMO channels are different between every pair of nodes, collecting all CSI requires large amounts of information. To reduce the number of channels required for computing the MIMO weights, all APs schedule the transmissions in the same direction as the initiator's link so that all transmissions are either on the downlink (from APs to clients) or on the uplink (from clients to APs). This approach removes the burden of clients having to measure CSI from other clients to cancel interference since all clients are scheduled as either transmitters or receivers.

The initiator AP, after determining that all nodes have entered the CFP by sensing the channel idle for a certain time, requests that its desired client sends a sounding packet. After receiving the sounding packet, the initiator AP sends a token to one of its neighboring APs and waits for the neighboring AP to acknowledge it. The token contains information as to whether the link selection is for the downlink, or the uplink. Whenever an AP receives this token, it first acknowledges the reception of the token by sending an ACK packet. If the AP receives the token for a second time, it returns the token back to the AP that sent it. If the AP has not already held the token, it chooses a client to form a link with (in the correct direction), and requests that it transmit a sounding packet. Then, the AP transfers the token to one of its neighboring APs. Once all neighboring APs have had a chance to hold the token, the current AP returns the token to the AP from which the token was initially received.

Using the network of Figure 1 again, an example of the token-passing mechanism is as follows. If node $\boldsymbol{B}$ becomes the initiator AP at a particular round, it initiates the CSI phase by broadcasting a CFP beacon. Once all APs get a chance to broadcast the CFP beacon, the initiator AP $\boldsymbol{B}$ senses the channel idle for a short time and requests that its desired client, say $\boldsymbol{b}_2$, transmit a sounding packet. APs $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ collect CSI from $\boldsymbol{b}_2$ since they are within the communication range. The initiator AP then sends the token to a neighbor AP, say $\boldsymbol{A}$, which requests that its desired client, say $\boldsymbol{a}_1$, send a sounding packet. AP $\boldsymbol{A}$, having no other neighbor to forward the token to, returns it to $\boldsymbol{B}$. AP $\boldsymbol{B}$ then forwards the token to $\boldsymbol{C}$ which chooses a client, say $\boldsymbol{c}_1$, and requests a sounding packet from it. Suppose $\boldsymbol{C}$ sends the token to $\boldsymbol{E}$, and it selects $\boldsymbol{e}_1$ as its desired client. AP $\boldsymbol{E}$ then requests that its client transmit a sounding packet. Because $\boldsymbol{E}$ does not have other neighboring APs, it sends the token back to $\boldsymbol{C}$. AP $\boldsymbol{C}$ sends the token to $\boldsymbol{D}$ which selects a client, say $\boldsymbol{d}_2$, and requests that it send a sounding packet. AP $\boldsymbol{D}$ then sends this token back to $\boldsymbol{C}$, which returns the token to the initiator AP. Figure 1 shows, using dashed lines, the link selection for this example.

After the token-passing part of this phase, APs have collected all necessary CSI for the clients selected and APs must forward all CSI to the worker AP. To initiate the forwarding of all CSI, the initiator AP broadcasts a contention-free end (CF-END) packet. Receiving APs contend for the channel and broadcast their own CF-END packets. After transmitting a CF-END packet, a leaf AP (i.e. $\boldsymbol{A}$, $\boldsymbol{D}$, and $\boldsymbol{E}$ in our ongoing example) transmits all the CSI measured to its parent in the tree structure. The transmissions use unicast packets and require acknowledgements from the parent APs. APs that are not leaf APs wait for all information to arrive from their children before aggregating the information and transmitting it to their parent. Once all information is received at the worker AP, it begins computing the MIMO weights.

2) MIMO Weight Computation and Distribution: During the MIMO-weight-computation phase, the worker node determines the MIMO weights to use for the current link selection. In our simulations, we use the Maximum Weighted Sum Rate (MWSR) algorithm from [7], although any MIMO weight algorithm with high performance can be used (for example [15, 16]).

To distribute the computed MIMO weights to their respective APs and clients, the worker AP transmits the MIMO weights to its own client, then aggregates the MIMO weights for each branch of the tree structure, and forwards them to each corresponding children. Each packet sent during the MIMO weight distribution contains the link set, and must be acknowledged by the receiver. An AP receiving an aggregated

MIMO weight packet deaggregates its MIMO link's weights, sends the corresponding MIMO weights to its client, and aggregates and transmits the remaining MIMO weights to each respective next branch in the tree.

In our example, the worker AP $\boldsymbol{B}$ computes the MIMO weights for each node, then sends the corresponding MIMO weights to its client. After this, the worker AP sends AP $\boldsymbol{A}$ the MIMO weights for $\boldsymbol{A}$ and its client. Finally, the worker AP sends $\boldsymbol{C}$ the MIMO weights of $\boldsymbol{C}$, $\boldsymbol{D}$, $\boldsymbol{E}$, and their respective clients. The receiving APs deaggregate the MIMO weights, then share the weights with their own clients before forwarding the remaining weights to their respective next-hop APs.

After sending the MIMO weights to its client, the initiator AP waits for a timer to expire before beginning the next phase. The timer value is based on an estimate of how long it takes for the rest of the MIMO weights to propagate through the network. In the small-scale scenarios targeted in this report, such a value can be easily determined.

D. Link-Set Advertisement and Synchronization

To synchronize the nodes for the data-transmission and acknowledgment phase, the initiator AP broadcasts a CFP beacon. In the case that MIMO weights were not computed during the current round (because the initiator AP chose an existing link set), then the initiator AP sets a flag to inform other APs that a link set is being reused and includes the desired link set within the beacon. However, if MIMO weights were computed during the current round (because the initiator AP chose a new link set), then the link set is redundant and it is not included with the beacon. By default, this beacon contains the duration of the CFP. However, to facilitate the synchronization, the beacon also includes a *beacon-propagation duration*, which is an estimate of how long the beacon takes to propagate through the network by the participating APs in the link set. A node receiving the beacon sets a timer for the beacon-propagation duration that it received and then rebroadcasts the beacon. However, before rebroadcasting the beacon, the node subtracts the delay it has incurred between receiving the beacon and retransmitting it from the beacon-propagation duration and from the CFP duration fields within the beacon.

The CFP initiated by the initiator AP protects the data and the acknowledgement transmissions during the next phases. Therefore, this CFP must have a duration that is large enough to account for the beacon propagation duration plus the transmission duration plus the acknowledgement duration.

Once the timer for the beacon-propagation duration expires, nodes listen to the channel, waiting for the channel to become busy before transitioning to the data-transmission phase.

E. Data Transmission

During the data transmission phase, nodes use the MIMO weights computed for the current link set to transmit and receive their data. To begin this phase, the initiator AP or its desired client, begins transmitting a data packet. Transmitter nodes begin their data transmissions immediately after they sense the channel busy.

The data-transmission phase has a fixed duration T_{data}. During this duration, transmitting nodes aggregate as many data packets as possible so as to occupy the entire transmit duration.

F. Acknowledgement Transmission

Once the T_{data} duration ends, the receivers sense the channel waiting for the channel to be idle for some time before acknowledging their packets by transmitting a BlockAck. To avoid additional overhead, all acknowledgements are sent simultaneously. To prevent having to compute MIMO weights for the reverse channel, nodes reuse the MIMO weights from the data-transmission phase to send/receive a single stream. Specifically, nodes that transmitted data use the first column of their beamforming weights as their combining weights for receiving the BlockAck, and nodes that received data use the first column of their combining weights, normalized to maximize the transmit power, as their beamforming weights for transmitting the BloackAck. This technique of reversing the roles but reusing the MIMO weights is commonly used to compute the transmitter weights [17–21]. Although this technique is suboptimal, we expect it to be sufficient since ACK packets are sent using modulation techniques that have lower SINR requirements than data packets.

After receiving or transmitting a BlockAck and sensing that the channel has been idle for a short duration, each AP transitions to the next round.

G. Fairness

Our proposed protocol does not enforce any particular fairness criteria. Instead, each AP chooses its fairness criteria within its BSS and tries to achieve that criteria by requesting the appropriate link sets during its turns.

H. Possible Short Comings

Possible problems include the case where two APs can communicate only through one or several clients. In this case, APs are not be able to communicate to share information or to synchronize, and the only solution is for clients to be modified further to have a more active role in the protocol. We will study this scenario in our future work.

IV. The Proposed Distributed Protocol

The distributed version of our protocol is very similar to the semi-distributed version except that there is no notion of a worker AP or a tree structure since the CSI-measurement and MIMO-weight-computation phase is completely distributed. To achieve this, each node must measure the CSI from all interfering nodes, compute its MIMO weights, and share them with neighboring nodes. This requires that clients be able to perform the following tasks:

- listen to sounding packets sent by interfering APs,
- compute their own MIMO weights, and
- participate during the MIMO-weight-computation phase by collecting MIMO weights for neighboring interfering APs, and transmitting their new MIMO weights.

The other phases, such as the link-selection-advertisement and synchronization phase, data-transmission phase, and acknowledgement phase are identical to the semi-distributed protocol.

V. Initial Simulation Results on the Steady-State Operation

In this section, we present initial simulation results for the steady-state operation of our protocols. We expect that the distributed and the semi-distributed versions of our protocol achieve similar performances because, during the steady state, all phases of the algorithm between the two versions are identical. In the following subsection, we use a partial implementation of the distributed version in the ns-3 simulator [8] to evaluate the steady-state operation of our algorithms.

A. ns-3 Simulation Results

To implement our protocol into ns-3, we have modified the Yans Physical Layer to support the matrix based MIMO physical-layer model. We have also modified the 802.11 protocol in ns-3 to support 802.11n-like capabilities, such as support for the 802.11n Greenfield preamble, support for sounding packets, and support for multiple parallel streams with independent data rates. In the simulation, packets are decoded on a stream-by-stream basis and a packet is received successfully only if all streams are decoded successfully. In our simulations, data rates are assigned by the APs according to the expected SINR of each stream using a modified version of the IdealWifiManager class. For a given SINR, the IdealWifiManager class chooses the highest data rate for which the bit-error rate (BER) is less than 1×10^{-5}. Table I shows the data rates and their corresponding SINRs as reported by ns-3.

TABLE I
Data rates and their SINR threshold for a BER of 1×10^{-5}

Data Rate	SINR
6Mbps	2.46851
9Mbps	4.80368
12Mbps	4.93702
18Mbps	9.60737
24Mbps	22.2137
36Mbps	45.4008
48Mbps	135.384
54Mbps	181.051

We have also implemented a mechanism for sending aggregate packets and acknowledging each packet individually using a BlockAck similar to the Aggregate MAC Protocol Data Unit (A-MPDU) mechanism of 802.11n [6]. In our implementation, an aggregate packet is composed of multiple individual packets that are transmitted sequentially without gaps. Each packet within an aggregate packet contains an index, which is used by the receiver to construct a BlockAck. Because each packet is independent, each packet contains repeated header fields, and so we expect extra overhead using our aggregate packet technique. Nevertheless, our simulations show increased performance when using our protocol.

We assume that all radios operate using a 20 MHz bandwidth at 5 GHz carrier frequency. Also, we assume flat fading across the whole bandwidth so that a single MIMO channel describes all OFDM subcarriers for that channel.

As mentioned earlier, we simulate the distributed version of our protocol and so all nodes compute MIMO weights and take turns transmitting them. In our simulations, each weight packet is spaced using a short-interframe space (SIFS). Also, a complex element in a MIMO weight matrix contains two 64 bit double-precision binary floating-point numbers: one for the real part, and one for the imaginary part. A receiver with four streams and four antenna elements needs to send a weight packet, containing the combining weights and other covariance matrices, that has a payload of 514 bytes. A transmitter with four streams and four antenna elements needs to send a weight packet, containing only the beamforming weights, that has a payload of 258 bytes.

When iteratively computing the MIMO weights, we say that a node reaches a convergence threshold of ϵ if the maximum absolute difference between the old MIMO weights and the new MIMO weights is below this threshold ϵ. In our simulations, when the MIMO weights reach a convergence threshold of $\epsilon_1 = 0.01$, we remove any stream that does not meet the minimum SINR listed in Table I and reallocate the power at each subsequent MIMO weight computation. We compute new MIMO weights and remove streams with low SINR until the weights converge within $\epsilon_2 = 0.0001$ at which point the node stops updating its weights.

During the data transmission phase, an AP sends as many packets as can fit within the $T_{data} = 10$ ms allocated time. We assume that APs send only User Datagram Protocol (UDP) data packets to their clients with a payload of 1048 bytes and that APs always have data to send to their clients.[1] After the $T_{data} = 10$ ms data-transmission duration, nodes compute the MIMO weights for the reverse channel, and clients wait for an SIFS before transmitting their BlockAck packets.

Using our implementation in ns-3, we simulate two different scenarios of the two OBSS problem. We consider the case where each AP has a single client, and the case where each AP has two clients. Our implementation in ns-3 lacks the round-robin order that is created during the AP-discovery phase. However, since we simulate only two OBSSes, the first AP that acquires the channel automatically becomes the initiator AP, and APs change roles after every ACK-transmission phase. Consequently, after receiving the BlockAck, the AP that was an initiator AP becomes a non-initiator AP and the AP that was a non-initiator AP becomes the initiator AP.

In the following subsections, we present our results for both the one-client-per-AP case and the two-clients-per-AP case. In both cases, every node in the network has four antenna elements.

1) Case One Client per AP: In the first scenario, illustrated in Figure 5, each AP has a single associated client. In this scenario, each AP chooses its client (forming a link set). Then the nodes compute the MIMO weights in a distributed fashion and store the final weights for use in all subsequent data

[1] Only downlinks are considered in these initial simulations.

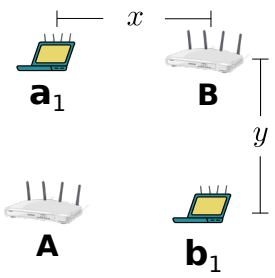

Fig. 5. Topology of two OBSSes where each BSS has one client.

transmissions.

We compare against a CSMA/CA strategy using a modified version of the 802.11 protocol on ns-3 that support the spatial multiplexing capability of MIMO links using the optimal singular-value-decomposition (SVD) weights with power allocated using the waterfilling algorithm [22]. Any stream that does not meet the minimum SINR is removed by setting its gain to zero within the waterfilling algorithm so that the algorithm allocates zero power on those streams. Additionally, we enable the Aggregate MAC Service Data Unit (A-MSDU) support within ns-3, which enables aggregate packets of up to 7935 bytes [6]. This CSMA/CA strategy represents the typical scenario in which two OBSSes must take turns to share the medium, each transmitting data at a maximum rate.

In Figure 6, we show the sum goodput as a function of x for $y = 50$ meters for the case of a single client per AP. The results shown are averaged over 10 random Rayleigh channels. We set the simulation time to five seconds. The results show that at low interference ($x = 100$ meters) and at high interference ($x = 20$ meters), our proposed protocol achieves a sum goodput that is 36% and 54% better, respectively, than the sum goodput achieved by the typical CSMA/CA mechanism with spatial multiplexing only.

2) Case Two Clients per AP: In the second scenario, illustrated in Figure 7, each AP has two associated clients. For this scenario, each AP selects a client, forming the first link set, and nodes compute the MIMO weights for this link set. Then, whenever one AP chooses its other client (for which MIMO weights have not yet been computed), the other AP also chooses its other client (for which MIMO weights have also not yet been computed). After the MIMO weights for this second link set are computed, the initiator AP always reuses either link set, depending on which client it selects. For this scenario, the protocol runs the MIMO-weight-computation phases two times, one for each link set.

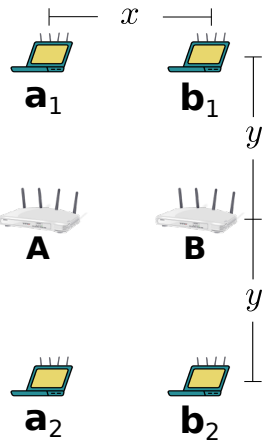

Fig. 7. Topology of two OBSS where each BSS has two clients.

Again, we compare against the CSMA/CA strategy with spatial multiplexing only. In Figure 8, we present results for the sum goodput as a function of x for $y = 50$ meters for the case of two clients per AP. The results show that the sum goodput of our protocol can be 40% and 49% better that of the CSMA/CA case for the cases where the two OBSSes have high interference ($x = 20$ meters) and low interference ($x = 100$ meters), respectively.

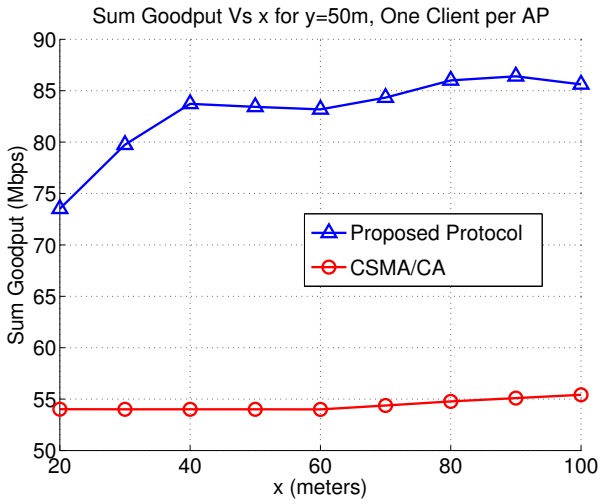

Fig. 6. Sum goodput as a function of x for $y = 50$ meters for the topology shown in Figure 5, where each of the two APs have a single client.

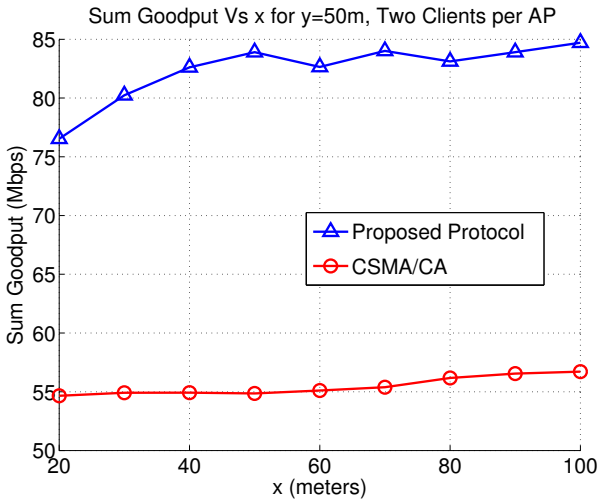

Fig. 8. Sum goodput as a function of x for $y = 50$ meters for the topology shown in Figure 7, where each of the two APs have two clients.

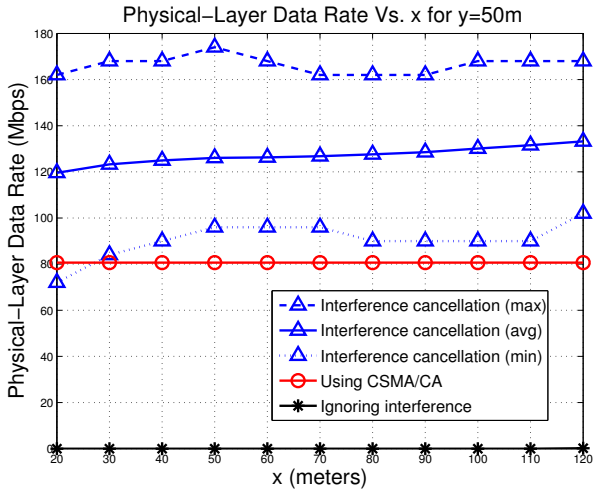

Fig. 9. Data rate for two OBSS with a single client in each BSS.

B. MATLAB Simulation Results

To exclude any protocol overhead, we also compare the achievable physical-layer data rates. We use MATLAB to simulate a similar single client per AP scenario shown in Figure 5. We fix $y = 50$ meters and vary x. We assume a path-loss exponent of 3. In Figure 9, we show the physical-layer data rate as a function of the distance between an AP and its unintended receiver. The results are averaged over 1000 random Rayleigh channels. In Figure 9, we show results for the case where both links are active in parallel (using spatial multiplexing and interference cancellation, labeled as interference cancellation), the case where links take turns (using spatial multiplexing only, labeled as CSMA/CA), and the case where the links transmit simultaneously, but do not perform interference cancellation (labeled as ignoring interference). Figure 9 shows that, as compared against the CSMA/CA strategy in terms of physical-layer data rate, the average performance improvement with interference cancellation at high interference ($x = 20$ meters) was 48%, at medium interference ($x = 60$ meters) was 57%, and low interference ($x = 120$ meters) was 65%. The maximum improvement ranged between 100% to 116% across all the values of x tested. Also, the results in Figure 9 confirm that the improvement observed is due to the interference cancellation capability of MIMO links since the performance drops significantly if interference is ignored and both links are active simultaneously.

The results obtained with this experiment show slightly higher improvements when performing interference cancellation than those improvements obtained using our protocol within ns-3 in Section V-A1 and Section V-A2. There are several causes for this decrease in performance improvement in our ns-3 simulations. One is the overhead of computing MIMO weights, which increases as the number of link sets increases. Another cause is the overhead of advertising the link set and synchronizing the data transmissions. Additionally, the aggregation mechanism used by our protocol is inefficient since various header fields are unnecessarily repeated for every packet.

Before we present our conclusions, we should note that we expect larger performance gains as the number of APs is increased beyond the two APs assumed since the total number of possible active streams increases [5, 23].

VI. Conclusions and Future Work

We have proposed semi-distributed and distributed protocols that take advantage of the interference cancellation and spatial multiplexing capabilities of MIMO links to improve the performance of OBSSes. Our design philosophy was to modify the clients as little as possible while exploiting the MIMO capabilities. In both strategies, the APs select an initiator AP whose job is to synchronize the transmissions. In the semi-distributed strategy, a worker AP collects all CSI, and computes and distributes the MIMO weights. To avoid having to compute beamforming and combining weights when sending acknowledgements, we reuse the weights used for the forward channel, but use only a single stream. We showed simulation results for a partial implementation of our distributed protocol using ns-3 for the case of two OBSSes. These simulations show that on average for two OBSSes, the goodput improvements are 54% and 49%, as compared to the goodput achieved by the typical CSMA/CA with spatial multiplexing only, for the scenarios where the APs have one and two clients each, respectively. MATLAB simulations show that the average performance improvement, in terms of physical-layer data rate with no overheads, of using spatial multiplexing and interference cancellation ranges from 48% up to 65% as compared to the case of spatial multiplexing only. Hence, our achieved results are within the general range of predicted benefits.

As part of our future work, we will simulate more than two APs. We will fully implement both the distributed and the semi-distributed protocols. Additionally, we will improve our protocols so that they can handle the cases where APs can communicate with other APs only through clients. Also, we will determine appropriate values for several parameters of our protocols, such as T_{data}, under different network conditions.

References

[1] L. Zheng and D. Hoang, "Applying graph coloring in resource coordination for a high-density wireless environment," in *IEEE CIT*. IEEE, 2008, pp. 664–669.

[2] Y. Fang, D. Gu, A. McDonald, and J. Zhang, "A two-level carrier sensing mechanism for overlapping BSS problem in WLAN," in *IEEE LANMAN*, Sep. 2005, pp. 1–6.

[3] B. Han, L. Ji, S. Lee, R. Miller, and B. Bhattacharjee, "Channel access throttling for overlapping BSS management," in *Proc. IEEE ICC*, Jun. 2009, pp. 1–6.

[4] S. Lo, G. Lee, and W. Chen, "An efficient multipolling mechanism for IEEE 802.11 wireless LANs," *IEEE Trans. Comput.*, vol. 52, no. 6, pp. 764–778, Jun. 2003.

[5] R. Srinivasan, D. Blough, and P. Santi, "Optimal one-shot stream scheduling for MIMO links in a single collision domain," in *Proc. IEEE Secon*, Jun. 2009, pp. 1–9.

[6] "IEEE std. 802.11n-2009: Enhancements for higher throughput," Oct. 2009, http://www.ieee802.org/.

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

[8] "The ns-3 network simulator," software available at http://www.nsnam.org.

[9] M. Guillaud, D. Slock, and R. Knopp, "A practical method for wireless channel reciprocity exploitation through relative calibration," *Proc. IEEE CSPA*, 2005.

[10] D. Peleg, "Time-optimal leader election in general networks," *J. of Parallel and Distributed Computing*, vol. 8, no. 1, pp. 96–99, 1990.

[11] N. Malpani, J. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proc. Int. Workshop on Discrete Algorithms and Methods for Mobile Comput. Commun.*, 2000, pp. 96–103.

[12] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks," in *Proc. Int. Conf. Netw. Protocols*. IEEE, 2004, pp. 350–360.

[13] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley, "Leader election algorithms for wireless ad hoc networks," in *Proc. DARPA Info. Survivability Conf. and Exposition*, vol. 1, Apr. 2003, pp. 261–272.

[14] A. Boukerche and K. Abrougui, "An efficient leader election protocol for mobile networks," 2006, pp. 1129–1134.

[15] S. Ye and R. Blum, "Optimized signaling for MIMO interference systems with feedback," *IEEE Trans. Signal Process.*, vol. 51, no. 11, pp. 2839–2848, Nov. 2003.

[16] M. Razaviyayn, M. Sanjabi, and Z. Luo, "Linear transceiver design for interference alignment: Complexity and computation," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, May 2012.

[17] F. Rashid-Farrokhi, K. Liu, and L. Tassiulas, "Transmit beamforming and power control for cellular wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 8, pp. 1437–1450, Oct. 1998.

[18] K. Gomadam, V. Cadambe, and S. Jafar, "Approaching the capacity of wireless networks through distributed interference alignment," in *Proc. IEEE Globecom*, Dec. 2008, pp. 1–6.

[19] ——, "A distributed numerical approach to interference alignment and applications to wireless interference networks," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3309–3322, Jun. 2011.

[20] L. Cortés-Peña, J. Barry, and D. Blough, "The Performance Loss of Unilateral Interference Cancellation," in *Proc. IEEE ICC*, Jun. 2012, pp. 4181–4186.

[21] S. Peters and R. Heath, "Cooperative algorithms for MIMO interference channels," *IEEE Trans. Veh. Technol.*, vol. 60, no. 1, pp. 206–218, Jan. 2011.

[22] E. Telatar, "Capacity of multi-antenna gaussian channels," *European Trans. Telecommun.*, vol. 10, no. 6, pp. 585–595, Dec. 1999.

[23] V. Cadambe and S. Jafar, "Interference alignment and degrees of freedom of the k user interference channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.