Real-time Distributed MIMO Systems

Ezzeldin Hamed Hariharan Rahul Mohammed A. Abdelghany Dina Katabi
Massachusetts Institute of Technology

ABSTRACT

Recent years have seen a lot of work in moving distributed MIMO from theory to practice. While this prior work demonstrates the feasibility of synchronizing multiple transmitters in time, frequency, and phase, none of them deliver a full-fledged PHY capable of supporting distributed MIMO in real-time. Further, none of them can address dynamic environments or mobile clients. Addressing these challenges, requires new solutions for low-overhead and fast tracking of wireless channels, which are the key parameters of any distributed MIMO system. It also requires a software-hardware architecture that can deliver a distributed MIMO within a full-fledged 802.11 PHY, while still meeting the tight timing constraints of the 802.11 protocol. This architecture also needs to perform coordinated power control across distributed MIMO nodes, as opposed to simply letting each node perform power control as if it were operating alone. This paper describes the design and implementation of MegaMIMO 2.0, a system that achieves these goals and delivers the first real-time fully distributed 802.11 MIMO system.

CCS Concepts

•**Networks** → **Network protocols; Wireless access points, base stations and infrastructure;** •**Hardware** → **Digital signal processing;**

Keywords

Wireless Networks, Multi-user MIMO, Distributed MIMO

SIGCOMM '16, August 22 - 26, 2016, Florianopolis , Brazil

 ISBN 978-1-4503-4193-6/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2934872.2934905

1. Introduction

Distributed MIMO has long been studied in theory because of its ability to dramatically increase the throughput of wireless networks [3, 14, 16, 9]. Recent years have seen significant interest in moving distributed MIMO from theory to practice. Multiple papers [11, 4, 17] have demonstrated the ability to synchronize distributed transmitters to enable them to concurrently transmit to multiple independent receivers, without interference.

The primary focus of past work has been to synchronize time, frequency and phase across multiple transmitters. While this is an important first step, several additional critical challenges need to be addressed in order to deliver practical distributed MIMO. Specifically, for distributed MIMO to work in practical settings they need to operate in real-time while being able to sustain their gains. They also need to adapt to dynamic environments with users moving around and the possibility of mobile clients.

This paper describes the design and implementation of MegaMIMO 2.0, the first real-time fully distributed 802.11 MIMO system. MegaMIMO 2.0 delivers a full-fledged 802.11 PHY, while meeting the tight timing constraints of the 802.11 protocol. It also supports dynamic environments and mobile clients. To achieve its performance goals, MegaMIMO 2.0 has to address the following key challenges:

(a) Real-time channel updates: At the heart of all distributed MIMO designs, there is a core subsystem that measures the channels from all the transmitters to all the different end users and uses them to apply desired beamforming and nulling. For any real-time system, these measurements have to be collected and updated on the scale of tens of milliseconds. Even in today's point-to-point MIMO systems, the process of collecting channel measurement is known to be high overhead [10]. The problem becomes *quadratically more expensive* in a distributed MIMO scenario because all senders have to measure channels to all clients for all subcarriers.

To illustrate how significant a problem overhead is, we simulate a distributed system consisting of N access points and N clients. We use typical feedback parameters (8 bits magnitude and phase for all OFDM subcar-

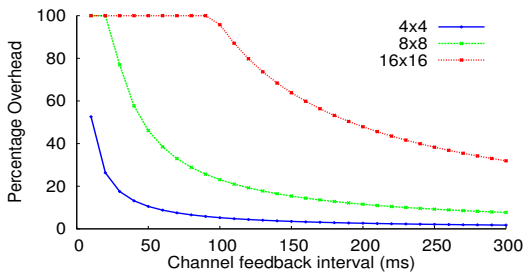


Figure 1: Channel feedback overhead for distributed MIMO system. The figure shows that at typical coherence times of about 100ms, the channel feedback overhead can significantly limit the gains of distributed MIMO systems, particularly as they scale to more nodes. For mobile clients with lower coherence times, the overhead is even higher.

riers, and QPSK with 1/2 rate for the channel feedback data with proper headers and DIFS), and evaluate overhead for various channel feedback intervals. Fig. 1 plots this overhead as a percentage of medium occupancy. As we can see, the overhead increases drastically with the number of users and can consume most of the wireless medium resources for large distributed MIMO systems. In fact, for a 16×16 system, feedback consumes most of the channel time at typical indoor coherence times of 100 ms. The overhead is even bigger for mobile clients with lower coherence times. It is therefore clear that for a real-time distributed MIMO system to be plausible, it cannot rely on explicit feedback and needs to devise other mechanisms to update channel information at low cost.

Addressing this problem in the context of distributed nodes is not easy. The natural approach for eliminating channel feedback would be to use channel reciprocity. Reciprocity refers to the property that the over-the-air channel from a node, say node A, to another node, say node B, is the same as the over-the-air channel from node B to node A. Point-to-point MIMO systems have leveraged reciprocity to enable a transmitter to infer the forward channel from its measurements of the reverse channel, without the need for any receiver feedback. They do this by performing a one-time computation of a constant calibration factor that compensates for the part of the channel introduced by the transmit and receive hardware, and correcting their estimate of the forward channel from A to B by applying this calibration factor to the reverse channel from B to A. In contrast, we demonstrate that in distributed MIMO systems, there is no such constant calibration factor that can be computed one-time, and applied to correct for reciprocity. We present a mathematical model that captures the variations in the calibration factor in a distributed system. We also introduce a protocol that computes these variations without additional transmission overhead, thereby extending the benefits of reciprocity to distributed MIMO systems.

(b) Power control: Practical wireless systems all use Automatic Gain Control (AGC), an analog module that dynamically adjusts the received signal to ensure it fills the range of the ADC. However, in a distributed MIMO system, the nodes must maintain a consistent view of the channels and other signaling information (e.g., their phase with respect to the lead access point). Furthermore, the measurements of the channels and signaling information have to be consistent across time. These requirements are at odd with today's AGCs, which operate independently from the AGCs on other nodes and have no memory across packets. Of course, one way to address this problem is to deactivate any individual control of AGC across the different devices.[1] However, doing this is not acceptable for any practical system, since the loss of data rates due to the inability to control power will translate to a large performance loss, defeating the very purpose of distributed MIMO.

MegaMIMO 2.0 therefore designs a system that infers the AGC parameters from the hardware on a per-packet basis, and incorporates these parameters into both distributed MIMO signaling and channel estimation.

(c) Rearchitecting the baseband and firmware: Distributed MIMO requires redesigning the firmware-hardware interface. Event timing in the existing Wi-Fi stack is local to each device. Thus, the firmware-hardware interface operates on event sequence, and timing is buried into the hardware. In contrast, in a distributed MIMO system, the hardware needs to react to interactions between devices, and perform coordinated actions across multiple devices, as opposed to purely local timing interaction like in traditional Wi-Fi. MegaMIMO 2.0 extends the interface between the PHY and the MAC to support such distributed coordination, and further enhances the real-time component of the MAC to enable it to effect this distributed coordination using local actions at each node.

We have built MegaMIMO 2.0 in a system-on-module comprised of an FPGA connected by a high-speed bus to an ARM core. Our implementation features a real-time full-fledged 802.11 PHY capable of distributed MIMO. We evaluated our system in an indoor deployment consisting of multiple 802.11 distributed-MIMO capable APs and unmodified 802.11 clients in an indoor testbed. Our results show the following:

- MegaMIMO 2.0 can deliver a real-time distributed MIMO system capable of adapting to mobile devices and dynamic environments with people walking around. In particular, a four-AP distributed MIMO system running MegaMIMO 2.0 delivers a median throughput of 120Mb/s and a maximum throughput of 194 Mb/s to four clients mounted on moving Roomba robots.

[1]This is typically the case in USRPs which have no support for AGC and on which prior systems have been demonstrated.

- MegaMIMO 2.0's reciprocity is both accurate and necessary for high throughput. Specifically, in a fully static environment, beamforming using reciprocity and beamforming using explicit feedback deliver the same gain. In contrast, in a mobile environment with four APs and four mobile clients, explicit feedback reduces the median throughput by 20% in comparison to reciprocity, due to feedback overhead. However, reducing the feedback rate can decrease the throughput by as much as 6x due to stale channel information. These results show the importance of using reciprocity even in a relatively small 4×4 distributed MIMO system. Since the feedback overhead increases quadratically with the size of the distributed MIMO system, we expect that reciprocity is even more essential for larger systems.
- MegaMIMO 2.0's ability to accommodate distributed power control is critical. In the absence of distributed gain control, the throughput of clients drops dramatically as the channels between some APs and clients become significantly weaker than channels between other APs and clients. Our experiments show a reduction of 5.1× in throughput when we deactivate MegaMIMO 2.0's distributed power control.

2. Related Work

There is a large body of theoretical work that analyzes the performance gains provided by distributed MIMO, and shows that it can scale wireless throughput with the size of the network [3, 14, 16, 9]. Motivated by these results, recent years have seen significant research effort in moving theory to practice [11, 4, 17, 2]. While these systems differ in details, they focus only on the problem of synchronizing the transmitters in time, phase and frequency, do not address power control and the overhead of learning and tracking the channels. Further, they demonstrate their results using one-shot channel measurements, and unlike MegaMIMO 2.0, do not design or show a full fledged physical layer or a real-time system capable of dealing with moving clients and dynamic environments.

There is a recent industry effort that targets building distributed MIMO systems [7, 1, 6]. However, existing systems are all based on CoMP(cooperative multi-point), which assumes a shared clock, distributed either via GPS or a wire, and a dedicated high throughput fiber backhaul infrastructure to deliver signals to all antennas with very high throughput and carefully controlled latencies. Examples of such systems are PCell [7, 1] and a demonstration by Ericsson [6]. In contrast, MegaMIMO 2.0 operates with fully distributed independent radios, and does not need a single clock. Further, it introduces a new technique for extending reciprocity to distributed MIMO systems and presents detailed evaluation results.

Also related to our work are papers studying the use of reciprocity for channel estimation [5, 8, 13]. The growth of massive MIMO systems has led to interest in scalable channel estimation techniques [12]. However, all these systems assume that all the antennas being calibrated for reciprocity share a single clock, and are on the same device. As a result, they do not extend to distributed scenarios where the different devices do not share a clock, and perform independent gain control. MegaMIMO 2.0 demonstrates how to extend reciprocity to these distributed scenarios, thereby enabling scalable channel estimation for distributed MIMO.

MegaMIMO 2.0 builds on the above related work but fills in an important gap by delivering the first fully operational 802.11 distributed MIMO PHY. This performance is enabled by novel techniques for extending reciprocity to distributed MIMO, coordinating power control, and providing a software-hardware architecture that can meet the strict timing constraints of distributed MIMO.

3. MegaMIMO 2.0 Overview

MegaMIMO 2.0 is a combined hardware-software system that performs distributed MIMO across multiple APs to multiple clients. The hardware implements a fully 802.11 a/g/n compatible PHY with enhancements to support distributed MIMO. The software running at each AP performs calibration of that AP for adapting uplink channel estimates and performing power control. Additionally, the software at each node performs distributed MIMO, tracking channels to each client, and coordinating between APs to perform distributed beamforming to the clients.

MegaMIMO 2.0 is designed to work across a distributed set of nodes without requiring a shared clock across the nodes. It builds upon a prior such system, MegaMIMO [11], and extends its design and implementation to support reciprocity, distributed power control, and a full-fledged real-time 802.11 PHY. At a high level, MegaMIMO works as follows. One AP acts as the lead AP (master AP), and all other APs act as slaves. Each slave AP maintains a reference channel from the lead AP. A joint transmission is initiated by the lead AP by transmitting a synchronization header, followed after a fixed time by the data. Each slave hears the synchronization header, compares it to the reference header to estimate its oscillator phase drift from the master, and corrects for this phase drift before jointly transmitting its data. We refer the reader to [11] for the full details of the system.

In the following sections, we describe each of the components of MegaMIMO 2.0.

4. Channel Update and Tracking

Knowing the channels is a core requirement for any multi-user MIMO system. In order for the AP to apply MIMO techniques like beamforming and nulling, it needs to know *a priori* the downlink channels to the clients. However, learning the downlink channels and tracking them as they change over time can cause excessive overhead for the system, as shown in Fig. 1. The

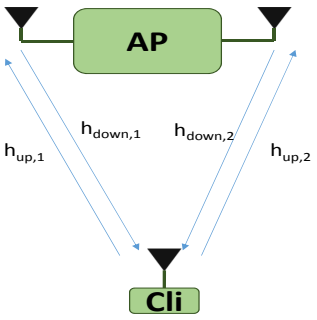


Figure 2: A 2-antenna AP transmitting to a 1-antenna client. With reciprocity based channel estimation, the AP would need to estimate the downlink channels $h_{down,1}$ and $h_{down,2}$ from the uplink channels $h_{up,1}$ and $h_{up,2}$.

overhead quickly increases for distributed MIMO as the number of participating APs and clients increases, and can eat up most of the gains even for relatively moderate sized distributed MIMO systems with, say, 8 or 16 APs. In this section, we describe a completely passive approach for learning the downlink channels and updating them in real-time. Our approach extends the normal reciprocity concept used in point-to-point MIMO to infer downlink channels from uplink channels. Below, we explain how reciprocity is used in today's MIMO, the challenges in extending the same concept to distributed MIMO, and finally our solution for addressing those challenges.

4.1 Reciprocity in Traditional MIMO

Let us consider the simple example in Fig. 2 where a two-antenna AP is communicating with a single-antenna client. As mentioned earlier, to perform MIMO techniques, the AP needs to know the downlink channels $h_{down,1}$ and $h_{down,2}$ to the client. The most straightforward approach would be to have the AP transmit on the downlink to the client from both antennas, and have the client measure the channels and transmit them back to the AP.[2] Alternatively, the AP can avoid the feedback overhead by leveraging the concept of reciprocity, which says that the forward channel on the air is the same as the reverse channel on the air. Thus the AP can leverage the client's transmissions to measure the uplink channels $h_{up,1}$ and $h_{up,2}$. It can then convert them to downlink channel estimates by multiplying them by a calibration factor, K_i, as follows:

$$\begin{aligned} h_{down,1} &= K_1 h_{up,1} \\ h_{down,2} &= K_2 h_{up,2} \end{aligned}$$

The calibration factor compensates for the fact that the measured channels include the hardware of the AP and the clients, as well as the air channels. Specifically, the downlink channels include the impact of the transmit chain on the AP and the receive chain on the client, while the uplink includes the transmit chain on the client and receive chain on the AP. Thus,

$$\begin{aligned} K_1 &= \frac{h_{tx,AP,1}}{h_{rx,AP,1}} / \frac{h_{tx,Cli,1}}{h_{rx,Cli,1}} \\ K_2 &= \frac{h_{tx,AP,2}}{h_{rx,AP,2}} / \frac{h_{tx,Cli,1}}{h_{rx,Cli,1}} \end{aligned}$$

From the above, it might seem that the calibration factor used at the AP are client dependent. However, this is not the case. Specifically, MIMO systems do not need the exact values of the channels but rather need the relative ratios of the channels from the different transmit antennas.[3] [15]. Therefore, instead of computing the above channels, we can divide all channels by K_1, and compute the following MIMO channels.

$$\begin{aligned} h'_{down,1} &= h_{up,1} \\ h'_{down,2} &= C_2 h_{up,2} \end{aligned}$$

where this new calibration factor:

$$C_i = \frac{h_{tx,AP,i}}{h_{rx,AP,i}} / \frac{h_{tx,AP,1}}{h_{rx,AP,1}}$$

is independent of the client.

Further, this calibration factor, C_i, is independent of time and can be computed once and used for all further transmissions from this AP.

4.2 Reciprocity in Distributed MIMO

Ideally, one would like to leverage the concept of reciprocity to learn the downlink channels without any client feedback, as is the case for traditional MIMO. Unfortunately, the traditional reciprocity formulation does not extend to distributed MIMO, *i.e.*, there is no such constant factors that can be computed once and used to infer the downlink channels from the uplink channels.

To understand why this is the case, let us go back to our previous example and assume that instead of two independent APs, we have two APs each with one antenna. In principle, a distributed MIMO system aims to emulate a traditional MIMO system with all the antennas on one humongous transmitter. Unfortunately, now each antenna is on a different AP, which has a separate oscillator and hence the two antennas would have carrier frequency offsets relative to each other. This simple fact means that the calibration factor C_i is no longer constant over time. Recall that the calibration factor is defined as:

$$C_i = \frac{h_{tx,AP,i}}{h_{rx,AP,i}} / \frac{h_{tx,AP,1}}{h_{rx,AP,1}}$$

When the two antennas are on the same device, they are connected to the same oscillator, and therefore their hardware chains do not change with respect to each

[2]The client would need to measure the channels for all subcarriers and send them back to the AP.

[3]In fact, all channels get eventually scaled by the transmit power and therefore, all measurements are up to a scaling factor.

other. However, when the two antennas are on independent APs, they are connected to different oscillators. Since the oscillator is part of the hardware chain, the differences between oscillators are part of the calibration factor. However, the differences between oscillators do not stay constant over time since their phases rotate relative to each other according to their CFO. In particular, say that the first oscillator has a carrier frequency ω_1 and the second oscillator has a carrier frequency $\omega_2 = \omega_1 + \Delta\omega_{21}$. Then, the calibration factor C_i changes over time as

$$C_i(t) = C_i(0)\exp(j2\Delta\omega_{21}t) \quad (1)$$

Two points are worth noting.

- First, one option to compute the calibration with respect to the lead AP is to compute the CFO with respect to the lead AP, and update the calibration factor according to Eq. 1. As mentioned earlier, and is widely known, this does not work in a distributed MIMO system since even small errors in computing the CFO accumulate over time leading to unacceptable errors in the estimate.
- The factor of two in Eq. 1 arises from the fact that MegaMIMO 2.0 needs to correct *uplink* channel estimates for the phase offset between master and slave and convert them to correct *downlink* channel estimates. In contrast, MegaMIMO directly corrects *downlink* channel estimates for the phase difference between the master and the slave.

In the following section, we describe a protocol that extends distributed MIMO to account correctly for this factor.

4.3 Distributed Reciprocity

We use the term *distributed reciprocity* to refer to the extension of the reciprocity context to distributed environments. Thus, the objective of distributed reciprocity is to compute the time dependent calibration parameter $C_i(t)$ which allows the distributed MIMO system to infer the downlink channels from the uplink channels.

Recall that, in distributed MIMO, there is a lead AP and multiple slave APs, and all the slave APs calibrate with respect to the lead AP. In the context of reciprocity, this means that the lead AP simply uses its uplink channel estimates as its downlink channel estimates without any correction, and all slave APs have to compute their downlink channel estimates as their uplink channel estimates corrected by their calibration factor with respect to the lead AP, *i.e.*, the $\Delta\omega$ in Eq. 1 is the CFO relative to the lead AP.

As mentioned earlier, simply estimating the phase offset using the CFO will lead to large errors. Thus, instead of computing the value of the calibration factor over time, we only compute the instantaneous value of the calibration factor exactly when the channel is measured, *i.e.*, we compute the difference between the oscillator phase on the master and the oscillator phase on the slave at the exact time as the uplink channel measurement.

MegaMIMO 2.0's calibration occurs in two steps: *initialization* and *update*. The *initialization* step occurs at reboot or when the AP joins the distributed MIMO system. It estimates both the magnitude and phase of the calibration parameter at the initialization time. The *update* step is invoked upon any reception from a client. It assumes the existence of some prior estimate of the calibration factor, and updates that prior estimate to account for change of phase relative to the lead AP.

We first describe the *initialization* step. The goal of this step is twofold. First, it estimates the magnitude and phase of the calibration parameter, as mentioned earlier. Second, it computes a reference channel from the lead AP to the slave AP, which is used during the *update* step, as described later.

The step consists of two back to back transmissions, the first from the lead AP to the slave, and the second from the slave AP to the lead. The slave measures the channel from the lead AP, $h_{AP1\to i}$, using the preamble of the first transmission and the lead measures the channel from the slave AP, $h_{APi\to 1}$, using the preamble of second transmission. By reciprocity, the forward air channel between the lead AP and the slave AP is the same as the reverse air channel. Hence, the slave can compute the initial calibration factor as:

$$C_i(0) = \frac{h_{APi\to 1}}{h_{AP1\to i}}$$

Additionally, the slave stores the channel from the lead AP, $h_{AP1\to i}$ as a reference channel, $h^{lead}(0)$.

We now explain the *update* step. For this step, we introduce the concept of a synchronization trailer. Similar to how a synchronization header synchronizes the transmission functions of slave APs during a joint transmission in distributed MIMO [11], we use a synchronization trailer here to synchronize the reception function on the slave APs. Specifically, when the client transmits its data, the lead AP follows the client transmission with a synchronization trailer. In fact, MegaMIMO 2.0 leverages the MAC layer ACK transmission from the lead AP which acknowledges the client's data to act as a synchronization trailer at all the slaves.

Each slave uses the preamble of the trailer to compute the channel from the lead, $h^{lead}(t)$ at that point in time. Now that the slave has an estimate of the lead channel both at time 0 and at time t, it can compute the total rotation, $\phi(t)$, of its oscillator relative to the lead AP as the difference between the two phases. Specifically,

$$\phi(t) = \Delta\omega t = angle(h^{lead}(t)) - angle(h^{lead}(0))$$

The slave then computes the updated calibration parameter at the current time t as

$$C_i(t) = C_i(0)exp(j2\phi(t))$$

and uses this updated calibration parameter to compute the downlink channel estimate.

However, these computed downlink channel estimates cannot directly be used for beamforming and joint transmission by the slaves. This is because the downlink channels for different clients are now estimated at different times (specifically, the times of their respective uplink transmissions). Recall that this is different from [11] where the APs jointly estimate downlink channels to all clients. As a result, during joint transmission, MegaMIMO 2.0 slaves cannot apply a single phase correction to the beamformed packet to account for the oscillator rotation between channel estimation and joint transmission to all clients, unlike in [11]. To account for this, MegaMIMO 2.0 instead performs an additional phase correction step during channel estimation. Specifically, each slave, after computing the instantaneous downlink channel estimate from the uplink client transmission as described above, then applies an additional phase rotation to infer the downlink channel estimate at an earlier time, specifically time 0. Note that the slaves can do this simply using the reference master-slave channel at time 0 ($h^{lead}(0)$), as well as the master-slave channel estimate at time t from the synchronization trailer. After this step, each slave then has an estimate for the downlink channel to each client as if it was measured at time 0, independent of the actual time of the uplink transmission.

These computed downlink channel estimates can be used for beamforming, nulling *etc.* in future joint transmissions as described in prior papers [11].

5. Power Control

Power control is a fundamental aspect of any wireless communication system. In particular, wireless receivers need to perform adaptive gain control to amplify their received signal and ensure that it maximally utilizes the range of the receiver ADC. Similarly, wireless transmitters need to scale their transmit power to fill the range of their DAC.

In point-to-point MIMO, each node performs power control locally. However since distributed MIMO involves the joint operation of multiple transmit and receive chains across multiple APs, one needs to ensure that power control across all these distributed chains is also coordinated. Below, we describe three key problems that occur due to the interaction between distributed MIMO and power control, and our corresponding solutions.

5.1 Coordinating AGC across time

The first step in a receive chain is a subsystem called Adaptive Gain Control (AGC), which constantly monitors the analog signal, and scales it up or down in the analog domain to make sure it fills the range of the ADC. For example, if your receiver has a 12 bit ADC, you would like your incoming signal to cover somewhere in the range of 10-12 bits. If the incoming signal is too large, the AGC will scale it down so it does not get clipped. If the incoming signal is too small, the AGC will scale it up so that it has enough bit resolution.

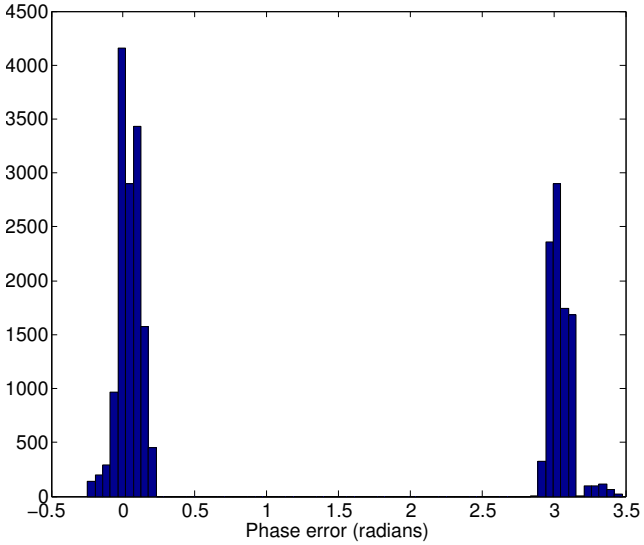


Figure 3: Histogram of the difference in phase between different AGC gains. The figure shows that there are very large differences in phase across the different gain settings, in fact as large as π radians. Different gain ranges involve activation of different elements of the analog RF front-end, and hence can introduce significantly different phase shifts.

The AGC has no memory across packets and makes a fresh gain decision for each packet. This unfortunately creates problems for distributed MIMO, which requires a predictable relationship between channel measurements across time.

Recall that distributed MIMO works by having each slave node maintain an estimate of its channel from the lead AP. Every time the slave hears from the lead, it recomputes this channel estimate. It assumes that any change in the phase of the channel estimate is due to oscillator drift between master and slave, and hence compensates for the change in channel phase. This process can interact adversely with the AGC function. Specifically, since the AGC has no memory between packets, it makes an independent scaling decision every time it hears a new signal from the lead. While, in principle, the decision should be similar since the signal is coming from the same source, in practice, due to noise in the medium, there is a level of uncertainty in the AGC decision. Different AGC decisions can introduce different scaling of the estimated channel, which the slave would incorrectly attribute to oscillator drift, leading to synchronization errors. Note that AGC scaling involves both magnitude and phase as different gains involve activating different elements of the analog chain. In fact, even a small variation of one step in the AGC can introduce very large variations in phase. For instance, in our step, changing the AGC gain by just 1 dB (for instance, from 34 to 35 dB) can introduce a phase change of π radians (since

this activates a different analog element - this is an inverting amplifier in our RF front-end). Not accounting for this would completely destroy the synchronization of the slave with the lead AP.

MegaMIMO 2.0 addresses this problem by inferring the phase introduced by each gain setting, and correcting for it on a per-packet basis so that it does not impact the synchronization of distributed MIMO. Estimating the phase introduced by the AGC, however, is not straightforward. The problem is that the hardware knows the AGC setting; however, it does not know the phase introduced by each specific setting. So, the device needs to calibrate the phase introduced by each gain value. Note that these phases are not the same across all radios from the same manufacturer; in fact, every individual device needs to do this calibration on its own to account for hardware variations.

MegaMIMO 2.0 performs this calibration as follows. For each antenna, the device transmits and receives on the same antenna measuring the loopback channel. It does this by operating the AGC in a mode where the AGC gain is set manually, and then stepping through the entire range of gains supported by the RF chain. For each gain setting, it measures the received channel. Of course, this received channel contains phase contributions from both the actual channel as well as the gain setting. However, note that as described earlier, MIMO only needs channel measurements relative to a reference. The same principle applies here, and hence MegaMIMO 2.0 simply computes the change in phase of the measured channel relative to the channel at a reference gain setting.

Note that simply doing this process naively by transmitting the same signal and simply changing the received gain setting will not work correctly. This is because the loopback channel is typically quite strong, and hence setting a high gain setting will cause the receiver to saturate and therefore report an incorrect channel. Hence, MegaMIMO 2.0 performs the process in two steps: It first estimates the ideal gain setting for the loopback channel by running the AGC in its regular mode where it is free to adapt the gain to the optimal setting. The hardware reports this gain setting to the calibration software. As the calibration software increases the receiver gain above this optimal AGC setting, it simultaneously digitally scales down the transmitted power by a corresponding amount. Specifically, for an increase in X dB in the receive gain setting, the calibration software applies a digital scaling factor of -X dB to the transmitted signal. This ensures that the signal swing at the ADC stays in the optimal range even as the gain setting is increased to larger values.

We calibrate all our boards using the technique described above. Fig. 3 plots a histogram of the relative phase between the different gains computed on different boards and across different subcarriers. The figure shows that it is essential to calibrate and account for the phase changes introduced by gains. In fact, some gain

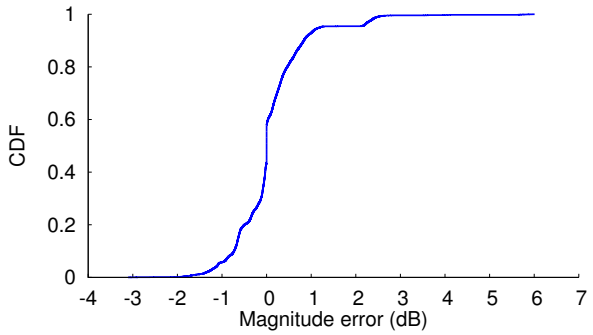


Figure 4: CDF of the difference between nominal and actual AGC gains. The figure shows that there are significant differences between nominal and actual gains, sometimes as large as 6-7 dB.

settings introduce a phase change as large as π radians. Not correcting for this phase change would completely destroy phase synchronization and beamforming in distributed MIMO. The same is true even of the smaller changes. There are differences on the order of 0.2-0.3 radians, which if not corrected for, would cap the maximum achievable SNR at any client at around 12-14 dB.

It is worth noting that the phase correction for AGC should be applied to all transmissions from the lead AP: the reference channel, the synchronization header transmissions for joint transmissions, as well as the synchronization trailer for client channel estimation transmissions described in §4.3.

5.2 Coordinating AGC across space

As described in §4.3, channel estimation in MegaMIMO 2.0 is performed independently by the different APs from a client's transmission. Since each AP applies gain control independently to its reception, the client's signal and hence the estimated channel from the client is scaled differently at different nodes. If these channels were simply communicated to the master without accounting for the AGC at each slave, they would each have an unknown scale component, and hence could not be used for joint precoding. Hence, each MegaMIMO 2.0 slave needs to compensate for the magnitude (and phase) change introduced by its AGC before communicating its estimated channels to the master.

Of course, the most straightforward way to do this would be for the receiver hardware to simply undo the effect of gain control. Specifically, if the receiver AGC applies a gain setting of X dB, it could simply scale down the measured channel magnitude by a corresponding amount. Since $Power(dB) = 20\log_{10}(Magnitude)$, we can compute the channel magnitude corresponding to an AGC gain of XdB as $10^{\frac{X}{20}}$. However, this does not work for two reasons. First, due to hardware variations, a gain setting of X dB does not actually provide an exact gain corresponding to that amount but has some errors around that number. Second, even if the gain is accu-

rate, it represents an average gain across all subcarriers. The actual gain in each subcarrier is different due to the presence of various receive filters.

MegaMIMO 2.0 addresses this issue by extending the calibration process described in the previous section. Specifically, in addition to the change in phase introduced by each gain, MegaMIMO 2.0 also computes the ratio of the channel magnitude in each subcarrier relative to the reference channel, during the calibration step. It then corrects each reported channel by the magnitude scaling factor in each subcarrier before using the channel for further beamforming computations.

The result of this calibration process can be used to see the effects of the deviation due to hardware variations and across subcarriers described above. Specifically, we convert the calibration factor for each gain computed as described above to the actual power gain, $\hat{X}dB$ (this is simply computed as $20\log_{10}(Calibration\ Factor)$. We then compute its difference from the nominal AGC gain, XdB expected for that AGC setting. We repeat this process for all gain settings and all subcarriers across all the boards in our system. Fig. 4 plots the CDF of all these values. As can be seen, the variations are significant, with the 90^{th} percentile going to 1 dB, and the maximums as large as 6 dB. To understand the impact of this error, consider a simple case where the difference between the nominal and actual AGC gain is 3 dB. Such an error can lead to an incorrect estimate of the channel magnitude by a factor of 1.4. Using a channel with this incorrect magnitude to null another signal of comparable power would lead to a residual noise of 0.4 times the magnitude of the channel, thereby capping the SNR of the system to $20\log 10(1/0.4) = 8dB$ independent of the actual SNR. This shows that calibrating AGC gain magnitude is fundamental to the correct functioning of distributed MIMO.

5.3 Coordinating transmit power

In distributed MIMO, each transmitter creates its transmitted signal by multiplying the user data with a precoding matrix. This precoding matrix ensures that the joint transmission satisfies the desired beamforming and nulling constraints. In principle, there are two ways to perform this multiplication. The first way is to perform the multiplication completely in the digital domain after which the signal is passed to the DAC and then to the power amplifier (PA). The problem with this approach is that if the multiplier significantly reduces the value of the signal such that it uses only a few bits of the DAC, the final signal will have very low resolution. Thus, a better approach is to split the multiplication between the analog and the digital domain. Specifically, the multiplication is split into two factors: the first is applied in the digital domain and ensures that the signal after multiplication still spans the range of the DAC, the second factor is then applied in the analog domain by controlling the attenuation of the PA. This ensures that the final signal has high resolution and therefore improves the overall SNR of the system.

Of course, changing the attenuation of the PA can cause phase offsets which one needs to precompensate for in the digital domain. This effect is similar to the AGC effect mentioned earlier and is calibrated using a similar technique.

6. Architecture

In the previous sections, we have described algorithmic modifications to the PHY layer in order to support efficient channel estimation and coordinated distributed power control. In this section, we describe how to modify the interface between the PHY and the MAC to support distributed MIMO, and the design of the time critical lower layer MAC subsystem to control the PHY.

A full-fledged distributed MIMO MAC has various functions, including updating channels from clients, determining which APs should jointly transmit to which clients at any time, computing the associated precoding matrices, and so on. Many of these functions occur at long timescales, corresponding to multiple packets, and we do not address these MAC functions in this paper. This paper focuses on the PHY and the real-time controls needed for the PHY.

The PHY interface to the MAC has two components: control of the PHY transmit subsystem by the MAC, and reporting from the PHY receiver subsystem to the MAC. The interface enables the PHY to be stateless across packets while still supporting distributed MIMO functionality. We first describe the enhancements to the interface, and then describe the enhancements to the time critical MAC subsystem to utilize these enhancements.

6.1 Transmitter PHY-MAC Interface

In the 802.11 standard, the interface between PHY and MAC for a packet transmission is called TXVECTOR. It provides the ability for the MAC to specify for each packet the associated payload, payload length, precoding matrix (if applicable), modulation and coding scheme (rate) to be used for the packet, and similar metadata. For distributed MIMO, the PHY needs to provide additional support for timing, phase, and frequency synchronization.

In particular, it supports the following additional functionalities:

Timing Synchronization: In addition to regular CSMA/CA transmission, the PHY provides the ability to transmit packets at specific time stamps defined relative to a system timer. This feature is to be used in triggered transmissions which is described later in this section.

Initial Phase Correction: For successful joint transmission, all slave APs are required to correct for any phase offset relative to the master AP at the instance of transmission. In order to do so the PHY transmit inter-

face provides an initial phase correction capability. This feature enables the MAC to define a slope and intercept to be applied on the OFDM subcarriers for the given packet. Using this initial phase correction all APs can be configured to start the joint transmission with no relative phase offset. However, due to the frequency offset between the APs the relative phase will keep changing during the packet.

Frequency Offset Correction: This feature provides the slave APs with the ability to correct for the relative frequency offset to the master AP during the given packet. It enables the MAC to define two different rates to correct for CFO and SFO. The first rate is the CFO correction and is provided as a phase change per sample, while the second rate is the SFO correction and is provided as a change in the phase slope (over the subcarriers) per symbol.

6.2 Receiver PHY-MAC interface

Similar to the TXVECTOR, the 802.11 standard defines the RXVECTOR to provide the interface between the PHY and MAC for a packet reception. For distributed MIMO, the PHY needs to provide additional support for MAC in order to adapt the transmitter metadata for future transmissions.

Specifically, for each packet, the receiver reports the following:

Frequency Offset Estimation: The receiver computes an estimate of the frequency offset with the transmitter for each packet and reports it along with the received data and other metadata to the MAC. The MAC maintains this information for every master transmitter, to be used in frequency corrections for future joint transmissions.

Channel: The receiver also reports the measured channels in each subcarrier to the MAC. Depending on the type of packet and its source, the MAC deals with the channels differently: either as a reference channel from the master, or as a channel corresponding to a synchronization header to be used for phase synchronization for joint transmission, or a channel measurement from a client that can be utilized for later beamforming.

6.3 Real-time MAC interface to the PHY

In this section, we describe the real-time components of the MAC that need to be implemented in hardware to ensure timing, phase, and frequency synchronization. Fig. 5 shows a schematic of the timing, frequency and phase synchronization subsystem.

Timing Synchronization Subsystem: In order to support timing synchronization, the real-time MAC component has the abstraction of triggered transmissions. A triggered transmission has two elements: a triggering condition, and an elapsed time after the triggering condition at which a packet is transmitted. A triggering condition comprises of either a transmission or reception of a packet with the MAC address of the master. This is a simple check and can be performed with low hardware complexity.

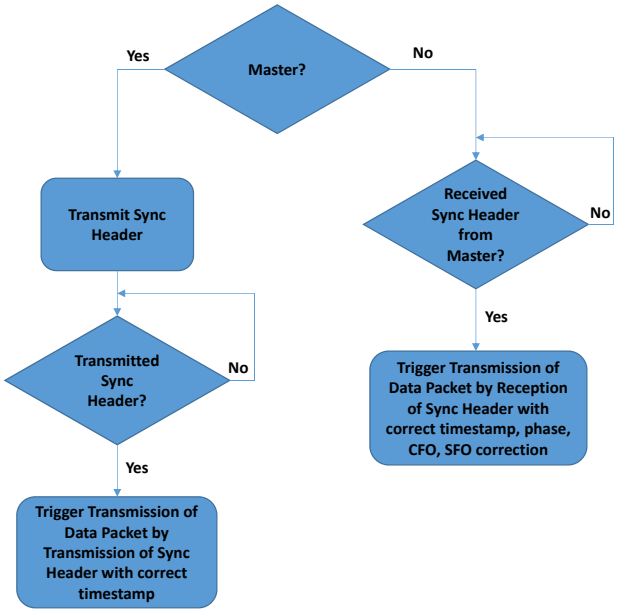


Figure 5: Timing, Frequency and Phase Synchronization Subsystem. This subsystem of the MAC operates in real-time. It interacts with the PHY and triggers transmission of packets based on transmission or reception of sync headers. It also applies the correct frequency and phase correction at the slave APs for the joint transmission.

At a high level, to initiate a joint transmission, the MAC at the master provides two packets to the physical layer. The first packet is the synchronization header, which is transmitted using the typical contention based medium access. The second packet is the joint transmission, whose transmission is triggered by the transmission of the first packet. The timestamp of this second transmission is a fixed time after the transmission of the first packet, say a SIFS.

At each slave, the MAC examines each received packet. If the received packet is a synchronization header, the MAC at each slave participating in the joint transmission then initiates a joint transmission triggered by this matching reception. The timestamp for this joint transmission is determined by the timestamp of reception of the synchronization header, and like in the master, is computed as a fixed time after the previous reception (specifically, it is the inter packet gap in the master less the receive-transmit turnaround time in the slave hardware).

Frequency and Phase Synchronization Subsystems: At each slave, this subsystem operates jointly with the timing synchronization subsystem to ensure correct joint transmission. Specifically, at each node, this subsystem examines every received packet. If the received packet is a synchronization header from a master, the MAC determines the associated CFO and SFO

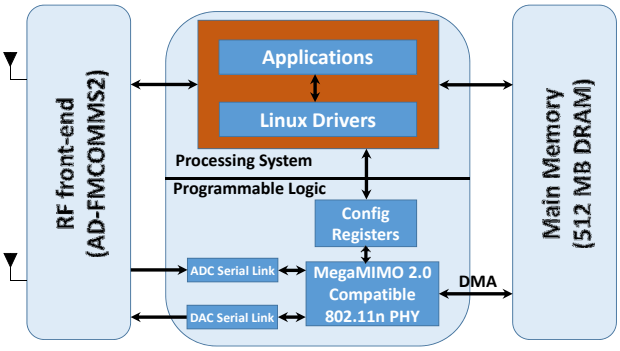


Figure 6: Platform Architecture The figure shows the software-hardware architecture of our platform. The PHY on the FPGA implements an 802.11n MIMO system as well as the real time synchronization facilities needed for distributed MIMO. The software on the ARM core configures the PHY and manages data transfer to and from the device.

Resource	Used	Utilization (%)
Slice Registers	49492	46.52
LUTs	43475	81.72
Block RAMs 36Kb	28	20
DSP48 (multipliers)	45	20.45

Table 1: FPGA Utilization on Xilinx Zynq Z7020. This table shows the utilization of different FPGA elements by our real-time PHY and MAC implementation.

of that master. Further, it uses the channel from the synchronization header, and compares it with the reference channel from that master to determine the initial phase correction to be applied to the joint transmission. It uses these parameters to apply the appropriate correction to the joint transmission packet at the slave. It is worth mentioning that this process needs to be performed in hardware in order to meet the short gap between synchronization header and the joint transmission, which is usually a SIFS.

7. Implementation

We implement MegaMIMO 2.0 and evaluate it in an indoor testbed.

Each node in our system consists of a Zedboard connected to an Analog Device FMCOMMS2 transceiver card. The Zedboard is equipped with a Xilinx Zynq Z-7020, which consists of an ARM dual-core Cortex A9 processing system connected to an Artix family FPGA via a high-speed AXI bus.

We implement our baseband system in Verilog on the FPGA. Our baseband consists of a full-fledged 802.11 a/g/n PHY layer that can operate in real-time and support all the 802.11 modulations and code rates on the FPGA. We enhance our PHY implementation to support distributed MIMO as described in the previous sections, and also implement various time critical MAC functionalities on the FPGA. Based on the size of the FPGA, our current implementation supports up to 4 distributed transmitters transmitting simultaneously to 4 independent clients. Table 1 shows the resource utilization of our real-time PHY and MAC implementation on our current FPGA platform.

We also implement the higher layer control system that triggers channel measurement, channel updates, precoding, and interfaces with user traffic in C on the ARM core. The FMCOMMS2 board acts as an RF front-end capable of transmitting and receiving signals in the 2.4 and 5 GHz frequency ranges. Each Zedboard is equipped with a Gigabit Ethernet interface through which it is connected to an Ethernet backhaul. Fig. 6 shows the architecture of our system.

8. Evaluation

We evaluate MegaMIMO 2.0 both through microbenchmarks of its individual components, and integrated system results of its overall performance.

(a) Testbed: We evaluate MegaMIMO 2.0 in an indoor testbed that emulates a typical conference room or lounge area. The APs are deployed high up on the walls near the ceiling as is typical in these environments. The clients are deployed at or near the floor level. The environment has furniture, pillars, protruding walls *etc.* that create rich multipath, and line of sight and non line of sight scenarios. We evaluate our system under both static and mobile conditions. All our experiments are conducted in the 2.4 GHz band, channel 10 (center frequency 2.457 GHz and 20 MHz bandwidth), using the 802.11n protocol.

(b) Compared Systems: We compare MegaMIMO 2.0's performance both with a traditional 802.11 system and distributed MIMO systems based on explicit channel feedback.

Note that in prior distributed MIMO systems, channel estimation happens in a sequential process, where at each time, the channel from each AP antenna to each client is measured jointly with one antenna of the lead AP. The reason for including one antenna from the lead AP in every measurement is to provide a reference to relate the measurements to each other even though they are performed at different times. These measurements are then corrected to account for phase rotation across time, which can be inferred from the phase changes in the reference channel measured from the antenna of the lead AP. The corrected measurements can then be used as the downlink channel estimates for beamforming. The details are described in [11].

For traditional 802.11, we assume the standard carrier sense based medium access protocol that allows one transmitter to transmit at any given time.

(c) Metrics: The metrics of interest that we compare are: the SNR after beamforming of the received packets at client, the total network throughput (in Mbps), and the individual throughput at each client (in Mbps). Depending on the experiment, we compare one or more of these metrics in different scenarios.

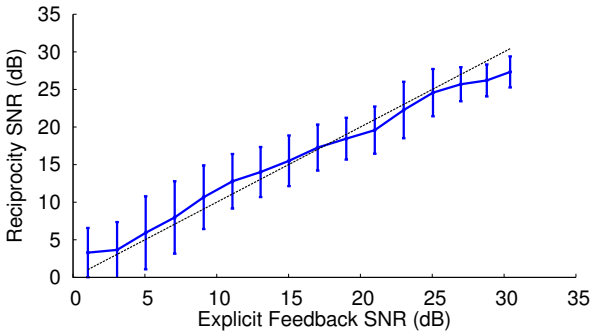


Figure 7: SNR with reciprocity (MegaMIMO 2.0) and explicit feedback (MegaMIMO) based distributed MIMO systems. The 45^o degree line is shown in dotted black. The figure shows that reciprocity based distributed MIMO can achieve the same SNR as explicit feedback across the range of SNRs.

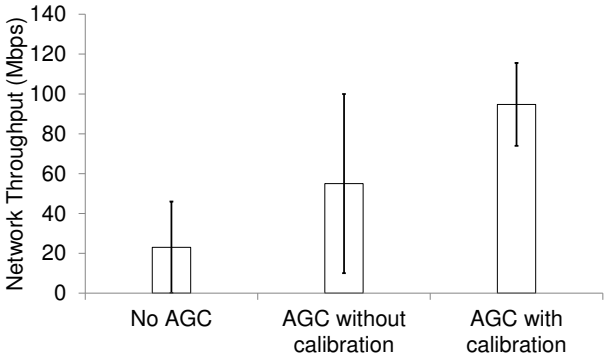


Figure 8: Throughput comparison of MegaMIMO 2.0 with full AGC calibration, MegaMIMO 2.0 using AGC without calibration, and MegaMIMO (fixed gain). The figure shows that distributed MIMO needs the use of AGC with full calibration in order to achieve high gains with low variance.

8.1 Accuracy of Reciprocity

Reciprocity eliminates the overhead of channel feedback, which tends to be excessive in distributed MIMO systems (see Fig. 1). However, would reciprocity lead to a degradation in MIMO gains in comparison to using channel feedback? In this section we answer this question by evaluating whether the channels inferred via reciprocity are as effective at delivering MIMO beamforming as the channels measured at the clients and explicitly sent to the access points –*i.e.*, explicit channel feedback.

Method: We evaluate a simple 2-transmitter, 2-receiver system in a static environment. The network has both downlink and uplink traffic with 90% of the traffic being on the downlink. We evaluate two scenarios: 1) The APs transmit packets on the downlink to the clients and receive explicit channel feedback from the clients, as in MegaMIMO. 2) The APs apply MegaMIMO 2.0's reciprocity protocol and use the clients' data transmissions to infer the downlink channels without any explicit feedback. We use both these explicit downlink channels, and estimated downlink channels to perform beamformed transmissions to the clients. We interleave the beamforming measurements using explicit feedback with those using reciprocity to ensure that the two compared methods experience similar channels. We then compare the SNR of these beamformed transmissions at the clients in the two scenarios. We repeat the experiment across a variety of locations, and the entire range of 802.11 SNRs, from 5-30 dB.

Results: Fig. 7 shows a plot of the SNR achieved with reciprocity as a function of the SNR achieved with explicit channel feedback for each topology. As the graph shows, MegaMIMO 2.0's reciprocity based channel estimation performs as well as explicit channel feedback through the entire range of SNRs. This means that distributed MIMO systems can safely use the reciprocity technique developed in this paper to avoid the excessive overhead of explicit channel feedback.

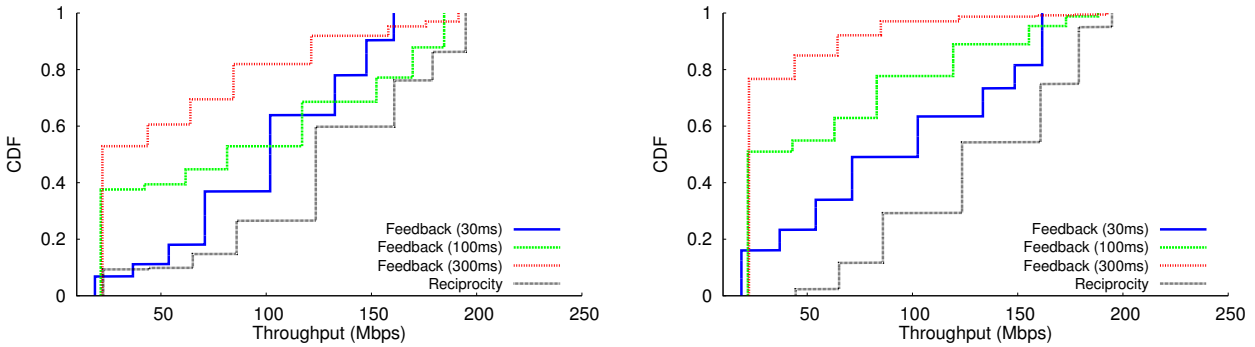
8.2 Need for AGC calibration

A key feature of MegaMIMO 2.0 is its ability to enable the use of AGC by calibrating for AGC phase and magnitude impact on a per-packet basis both in hardware and software. In this section, we evaluate the importance of this calibration.

Method: We evaluate a 4-transmitter, 4-receiver system in a static environment. The network performs distributed MIMO beamforming from all 4 transmitters to all 4 receivers using reciprocity based channel estimation. The network has both uplink and downlink traffic with uplink traffic accounting for 10% of the load. The first is full-fledged MegaMIMO 2.0 with AGC running on all nodes, and both hardware and software calibration. The second is MegaMIMO 2.0 with AGC running on all nodes, but with only magnitude calibration and no phase calibration. The final system is MegaMIMO 2.0 with a fixed manually chosen gain setting (this is similar to MegaMIMO in USRPs). We repeat the experiment 10 times and change the client locations across runs. We compare the network throughput obtained by the system in all three settings.

Results: Fig. 8 shows the network throughput in each of the three scenarios described above. The following points are worth noting.

- MegaMIMO 2.0 with an operating AGC, and both magnitude and phase calibration, achieves the highest throughput among all the systems.
- We compare MegaMIMO 2.0 with a system with an operating AGC where we turn off the phase calibration but still apply the magnitude calibration based on the rated AGC gain. That is, we assume that an AGC gain of X dB scales the signal magnitude by $10^{\frac{X}{20}}$ (as described in §5.2), and correct for it accordingly. We use this reference because ignoring magnitude calibration completely makes the beamforming extremely sensitive even to small changes in AGC



(a) Environmental Mobility (b) Client Mobility

Figure 9: CDF of user throughput of a 4x4 distributed MIMO in mobile scenarios using MegaMIMO 2.0's real-time PHY with reciprocity, and MegaMIMO 2.0's real-time PHY but with explicit feedback at different intervals. The figure shows that MegaMIMO 2.0's real-time PHY can react to changing environments and deliver a distributed MIMO system even in the presence of mobility. The figure also shows that reciprocity based distributed MIMO always outperforms explicit feedback based systems. At low feedback rates, the feedback based systems suffer from stale channel information. At high feedback rates, the systems have fresh channel information but have high overhead.

gain during channel estimation at any of the APs (the impact of gain errors is analyzed in §5.2 in a different context). Since this system does not correct for phase, it sometimes experiences extremely large errors and therefore has higher variance than the full-fledged MegaMIMO 2.0 system. Additionally, it loses performance because of differences between the actual hardware gain and the nominal hardware gain intended by the AGC.

- Finally, the system with lowest performance is the one with manually assigned gain control. This is because one cannot pick a single gain value for a large system that will allow all master-slave and all client-AP links to function at reasonable SNRs across a variety of topologies. As a result, this system too has low throughput and high variance.

8.3 Real-time Performance

In this section, we evaluate whether MegaMIMO 2.0 can deliver a distributed MIMO system capable of operating in real-time.

Method: We consider dynamic environments common in indoor settings. We introduce dynamism into the system in two ways. In the first case, all the nodes are static, but there is mobility in the environment due to moving people. In the second case, we introduce additional mobility by moving the nodes themselves. Specifically, the clients in the testbed are moved by either mounting them on Roomba robots, or by walking humans. The client mobility speeds change from one run to another and are in the range [0.2m/s, 1m/s].

We deploy 4 nodes acting as APs in our testbed, and 4 nodes acting as clients. We compare two schemes: distributed MIMO with explicit channel feedback, and distributed MIMO with reciprocity. Note that both schemes are running MegaMIMO 2.0's real-time PHY with all of its components (AGC, calibration *etc.*, necessary to deal with mobility) and differ only with the mechanism for tracking the channels. We compute the throughput of individual nodes under 4 scenarios: reciprocity based distributed MIMO where the clients send uplink traffic about 10% of the time (*i.e.* 90% of the traffic is downlink traffic), and explicit feedback at three different feedback intervals: 30 ms, 100 ms, and 300 ms. We run this experiment for several hours. We repeat it for various topologies and the entire range of 802.11 SNRs.

Result: Figs. 9(a) and (b) plot the CDF of the throughput obtained by each client in the various scenarios. A few of points are worth noting.

- First, MegaMIMO 2.0's real-time PHY can support dynamic environments, and adapt to both moving devices and people. In particular, a four-AP distributed MIMO system running MegaMIMO 2.0 delivers a median throughput of 120Mb/s and a maximum throughput of 194 Mb/s to four *mobile* clients.
- Second, as expected, reciprocity based distributed MIMO obtains the highest throughput in both scenarios: dynamic environment and dynamic clients. The median throughput gain of reciprocity based distributed MIMO over explicit feedback ranges from 20% to 6x in when the device is mobile, and 10% to 6x when device is static yet people are moving around. Further, explicit feedback systems with infrequent feedback (100-300 ms) get significantly lower throughput than the case of reciprocity, in spite of having significantly lower channel feedback overhead. This is because they suffer from stale channel information since mobility causes the actual channels to deviate from the reported channels faster than the feedback interval. In fact, explicit feedback with intervals of 100-300 ms suffers from extremely low throughput between 35-50% of the time because of channel stale-

ness. Explicit feedback at a high rate (30 ms) is also worse than reciprocity. In this case the APs have fresh channel information, but explicit feedback suffers a throughput loss due to the overhead of feedback.

- The performance of the explicit feedback system in Figs. 9(a) and (b) is worse than the simulation results of a 4×4 system in Fig. 1. This is because the simulation results do not account for the impact of stale channel information on the behavior of distributed MIMO systems.
- Overall the empirical results show the importance of using reciprocity even in a relatively small 4×4 distributed MIMO system. Since the feedback overhead increases quadratically with the size of distributed MIMO, we expect that reciprocity is even more essential for larger systems.

8.4 Performance in a Static Environment

Finally, we check MegaMIMO 2.0's performance in static settings to ensure that our implementation supports the gains expected from distributed MIMO in static environments.

Method: We deploy 4 nodes acting as APs in our testbed, and 4 nodes acting as clients. We compare MegaMIMO 2.0 with reciprocity to traditional 802.11. The network has both uplink and downlink traffic with uplink traffic creating ≈10% of the load. We perform this experiment for 15 different runs, and change the clients' locations from one run to another. We evaluate the throughput of each of these three schemes in three SNR ranges: low (6-12 dB), medium (12-18 dB), and high (18+ dB).

Result: Fig. 10 shows that MegaMIMO 2.0 with reciprocity achieves about 3.6× gain with 4 transmitters. This is compatible with the behavior expected from distributed MIMO since this system can deliver 4 packets to 4 clients concurrently. The figure also shows that this behavior is consistent across the whole range of 802.11 SNRs.

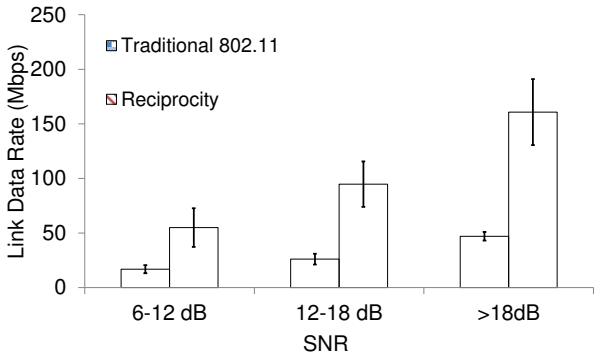


Figure 10: Comparison of throughput obtained with traditional 802.11, and MegaMIMO 2.0 with reciprocity. The figure shows that MegaMIMO 2.0's implementation can scale throughput linearly with the number of nodes. At all SNRs, the throughput of MegaMIMO 2.0 with 4 nodes is 3.6× the throughput of a single 802.11 link.

9. Future Work

In this paper, we discuss the architecture and implementation of a practical full-fledged real-time PHY and real-time MAC layer for distributed MIMO. This system can serve as a building block to address the next set of questions that need to be tackled for distributed MIMO.

Specifically, for a joint transmission, the distributed MIMO PHY receives the master, set of slaves, and the set of clients as an input. It is the responsibility of a higher (non real-time) MAC layer to pick this set of APs and clients based on the current traffic patterns, as well as the channels between different APs and clients, in order to optimize network fairness and throughput. Additionally, this MAC layer will also determine which among the APs will act as a master for any given transmission. Ideally, the master would be picked to be at the center of the transmission cluster so that it can be heard with good SNR by all other APs and clients. The MAC layer can utilize the measured channel information for doing so.

10. Conclusion

This paper presents MegaMIMO 2.0, the first full fledged real-time PHY capable of supporting distributed MIMO. MegaMIMO 2.0 is 802.11 compatible, and addresses various key practical issues required for a practical PHY layer that can operate across diverse SNRs and channel conditions. Further, it extends the 802.11 PHY interface to support a MAC layer capable of distributed MIMO, and can therefore serve as a building block for a full stack distributed MIMO system. We believe that MegaMIMO 2.0 represents a significant step forward in bringing distributed MIMO closer to practice.

Acknowledgments: We thank the NETMIT group, Arthur Berger, our reviewers and our shepherd, Deepak Ganesan, for their insightful comments. This work is funded by NSF. We thank members of the MIT Center for Wireless Networks and Mobile Computing: Amazon, Cisco, Google, Intel, Mediatek, Microsoft, ST Microelectronics and Telefonica for their interest and support.

11. References

[1] An Introduction to pCell. http://www.rearden.com/artemis/An-Introduction-to-pCell-White-Paper-150224.pdf. Artemis, February 2015.

[2] O. Abari, H. Rahul, and D. Katabi. AirShare: Distributed Coherent Transmission Made Seamless. In *IEEE INFOCOM 2015*, Hong Kong, China, April 2015.

[3] S. Aeron and V. Saligrama. Wireless Ad Hoc Networks: Strategies and Scaling Laws for the Fixed SNR Regime. *IEEE Transactions on Inf. Theor.*, 53(6), 2007.

[4] H. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire. AirSync: Enabling Distributed Multiuser MIMO With Full Spatial Multiplexing. *Networking,*

IEEE/ACM Transactions on, 21(6):1681–1695, Dec 2013.

[5] A. Bourdoux, B. Come, and N. Khaled. Non-reciprocal transceivers in OFDM/SDMA systems: impact and mitigation. In *Radio and Wireless Conference, 2003. RAWCON '03. Proceedings*, pages 183–186, Aug 2003.

[6] 5G live test: Multipoint Connectivity with Distributed MIMO. https://www.youtube.com/watch?v=jCO68dPoNwA. Ericsson Inc.

[7] A. Forenza, R. W. H. Jr., and S. G. Perlman. *System and Method For Distributed Input-Distributed Output Wireless Communications*. U.S. Patent Application number 20090067402.

[8] M. Guillaud, D. Slock, and R. Knopp. A practical method for wireless channel reciprocity exploitation through relative calibration. In *Signal Processing and Its Applications, 2005. Proceedings of the Eighth International Symposium on*, volume 1, pages 403–406, August 2005.

[9] A. Ozgur, O. Leveque, and D. Tse. Hierarchical Cooperation Achieves Optimal Capacity Scaling in Ad Hoc Networks. *IEEE Trans. on Info. Theor.*, 2007.

[10] E. Perahia and R. Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. Cambridge University Press, 2013.

[11] H. Rahul, S. Kumar, and D. Katabi. MegaMIMO: Scaling Wireless Capacity with User Demands. In *ACM SIGCOMM 2012*, Helsinki, Finland, August 2012.

[12] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong. Argos: Practical many-antenna base stations. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 53–64, New York, NY, USA, 2012. ACM.

[13] J. Shi, Q. Luo, and M. You. An efficient method for enhancing TDD over the air reciprocity calibration. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 339–344, March 2011.

[14] O. Simeone, O. Somekh, H. Poor, and S. Shamai. Distributed MIMO in multi-cell wireless systems via finite-capacity links. In *ISCCSP*, 2008.

[15] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[16] S. Venkatesan et al. A WiMAX-based implementation of network MIMO for indoor wireless. *EURASIP*, '09.

[17] V. Yenamandra and K. Srinivasan. Vidyut: Exploiting power line infrastructure for enterprise wireless networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 595–606, New York, NY, USA, 2014. ACM.