Practical Distributed MIMO for WiFi and LTE

by

Ezzeldin Omar Hussein Hamed

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2018

Certified by .....
Dina Katabi
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Practical Distributed MIMO for WiFi and LTE

by

Ezzeldin Omar Hussein Hamed

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Distributed MIMO has long been known theoretically to bring large throughput gains to wireless networks. Recent years have seen significant interest and progress in developing practical distributed MIMO systems. Multiple systems have developed coordination protocols to synchronize multiple nodes and enable them to transmit simultaneously without interference. However, there are still several challenges that need to be addressed in order to have a practical distributed MIMO system for WiFi and LTE.

In this thesis, we address the practical limitations in distributed MIMO systems. We present algorithms that enable practical distributed MIMO in the context of WiFi and LTE. Furthermore, we demonstrate these algorithms through Hardware/Software architectures which are implemented on an FPGA platform and run in real-time.

We present two different systems in this thesis. For WiFi, we present MegaMIMO 2.0, the first real-time 802.11 distributed MIMO system. MegaMIMO 2.0 builds on top of state of the art work in distributed MIMO, and adds support for dynamic environments and mobile clients while meeting the tight timing requirements of the 802.11 WiFi standard. We also introduce Chorus the first scalable distributed MIMO system for LTE. Chorus works with the different modes of LTE and is resilient to changes in network configuration. Both MegaMIMO 2.0 and Chorus require no change to the user devices, and are fully compatible with the 802.11 standard for WiFi and the 3GPP standard for LTE.

Thesis Supervisor: Dina Katabi
Title: Professor

Dedicated to my daughters Yasmine, Mariam & Hajar

Acknowledgments

All praise and thanks are due to Allah, Lord of the worlds. The one who blessed me with the opportunity to start my PhD at MIT, and surrounded me with all the support I needed to carry it through. I am grateful for many people who helped me through this long journey since the day I decided to join MIT, 6 years ago.

I was blessed to have Prof. Dina Katabi as my PhD Supervisor. Dina was a continuous source of motivation and support throughout my PhD. The amount of energy and excitement she had about our work was always an example to follow. I managed to get the best out of myself under Dina's supervision, she introduced me to the field of wireless networks, and in few years, helped me to expand my knowledge to be orders of magnitude bigger. I am proud to be one of Dina's students, and I hope to bring the lessons I learned from her into practice.

I have been fortunate to work with my colleague Rahul. Rahul has been deeply involved in the design and implementation of all the systems that I developed for my PhD research. Not only that, but thanks to Rahul I was able to learn so many things in short time. Rahul is like a walking encyclopedia, working with Rahul helped me strengthen my understanding about distributed systems, wireless communication, software design, and many other things inside work and outside.

Having the opportunity to work in the NETMIT group is like a dream come true. Each and every individual in the group is so special, and carefully selected. I am grateful for every converstation that I had with each one of my colleagues, the constructive feedback that I got on each project I worked on, and the friendly environment that helped me fit in very quickly and be a productive member of the group.

I am thankful for all my family and friends who has been a continuous support for me throughout my PhD. My mother Samira, who is always worried about me and doing everything she can to support me. My father Omar, who has never stopped believing in me and praying for my success. My sister Hind, who is always a reason for me to laugh and forget about all my worries. My mother in law Samia, who has

been the superhero that I summon whenever things get tough. And my two close friends Amr Suleiman (ZAZA), and Amr Hola who are always there for me whenever I needed them, advising me and cheering me up.

I am grateful to my wife Aisha for sacrificing everything she had just to be with me and take care of me and our kids during my tough PhD years. And I am blessed to have three little daughters who fill my life with love and beauty.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent years have witnessed fast growth in the use of mobile devices, and a significant increase in demand for wireless throughput. However, the wireless spectrum is limited. In the case of WiFi, devices fight over the unlicensed frequency bands and with the increased demand the network becomes easily congested and suffers from losses due to interference. In the case of cellular networks, it costs billions of dollars to obtain a spectrum license [24]. Therefore, researchers and network providers have been considering advanced MIMO (Multiple Input Multiple Output) techniques to increase throughput per unit spectrum and provide better connectivity.

Initial research and theoretical work on using MIMO for wireless communication showed that it could increase the throughput per unit spectrum linearly with the number of antennas [64]. However, when antennas are densely packed together in a small area, the channels between the different antenna pairs become correlated. And as a result, the capacity of such MIMO systems would not be proportional to the number of antennas, and gains would saturate after adding few antennas to the system.

Distributed MIMO has long been studied in theory because of its ability to eliminate the limitations in traditional MIMO and deliver dramatic increase in the wireless network throughput [3, 60, 69, 44, 73]. It does so by synchronizing the oscillators on independent nodes, allowing a network of transmitters to act as if they were one huge MIMO system. Recent years have seen significant advances in moving distributed

MIMO from theory to practice [49, 2, 5, 71, 55, 74]. Multiple systems have developed coordination protocols to synchronize distributed transmitters and 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 that was an important first step, there are still several challenges that need to be addressed in order to have a practical distributed MIMO system for WiFi or LTE. Specifically, in order for any proposed technique to be practical it must be able to fit within the current WiFi and LTE standards without major changes. Further, it should work with existing user devices. Techniques that require significant changes to the standards or even minor changes to the user devices, will need decades to be deployed and in many cases will end up not deployed at all.

In this thesis, we present new algorithms and system design principles that enable distributed MIMO under the existing standards for WiFi and LTE. We introduce Hardware/Software architectures to demonstrate the performance of our systems on an FPGA platform. We show that our systems provide practical, standard compliant, real-time implementations of distributed MIMO for WiFi and LTE.

While the basic idea of distributed MIMO synchronization is similar in WiFi and LTE, the rest of the system must differ significantly due to major differences between the two protocols. Specifically, the Medium Access Control (MAC) layer in WiFi is intrinsically different from LTE. In the case of WiFi, the MAC protocol uses a random access technique, called Carrier Sense Multiple Access (CSMA), which requires any transmitter to listen to the medium and wait until the medium is free before it can transmit. Since WiFi medium access is on-demand, when adding distributed MIMO to WiFi access points (AP), the synchronization process has to be performed on demand. In contrast, the LTE MAC protocol relies on a central unit responsible for scheduling the medium and assigning the different time/frequency slots to the transmitting devices. Given the schedule, LTE small cells continuously transmit their signals without idle time between frames. Since LTE small cells access the medium continuously, adding distributed MIMO to small-cell base stations requires continuous synchronization, and cannot be done on-demand. Below we elaborate on

these differences between the two protocols. In later chapters, we revisit the challenges and explain our solutions for WiFi and LTE separately.

1.1 Challenges for WiFi Distributed MIMO

The random access nature of the WiFi protocol and the on demand synchronization impose a set of challenges that need to be addressed in order to have a practical distributed MIMO system for WiFi. Those challenges can be summarized as follows:

(a) Channel Feedback Overhead: In order to adapt to changes in the environment and mobility of users, a distributed MIMO system will need to continuously track the changes in the wireless channels between all the APs and user devices. This problem is complicated by WiFi's on-demand random access MAC, where channel measurements have to be bootstrapped every time a node accesses the medium. State of the art solutions rely on channel feedback from the user devices to track the wireless channel. However, this introduces a large overhead on the wireless medium. For example, the overhead of channel feedback for an 8×8 system in typical indoor scenarios is more than 25%. We discuss the channel feedback overhead in more details later in Section 4.1.1.

(b) Distributed Gain Control: A WiFi receiver design typically have Automatic Gain Control (AGC) on the receiver path. The task of this AGC is to adjust the magnitude of the signal in order to prevent close by transmitters from saturating the receiver, and reduce quantization noise in the case of far away transmitters. Due to the random access nature of WiFi, a WiFi receiver can not know the source of a packet until it is fully decoded. Hence, the AGC would start estimating a new gain configuration for each packet it receives. And as a result, it introduces both magnitude and phase mismatch that affects the phase correction and ruins the synchronization. We discuss those properties in more detail in Section 4.3.

(c) Tight Timing Constraints of WiFi: The CSMA protocol, used in WiFi, requires fast turn around time in order to maintain access to the medium. For example, a WiFi receiver is expected to turn around within 10 microseconds from receiving the

last sample of the packet to transmitting the MAC layer Acknowledgement (Ack) packet. If the medium stays idle for more than that time, other nodes will be allowed to get access to the medium and start transmitting. This tight timing requirement, puts real-time constraints on the algorithms one may design for phase correction, channel estimation, AGC correction etc.

To address the above challenges we present MegaMIMO 2.0, the first real-time 802.11 distributed MIMO system. MegaMIMO 2.0 builds on top of state of the art work in distributed MIMO, and adds support for dynamic environments and mobile clients, without excessive channel feedback. It also deals with the AGC constraints and the tight timing requirements of the 802.11 WiFi standard.

1.2 Challenges for LTE Distributed MIMO

The LTE MAC protocol relies on a centralized scheduling architecture, and an LTE cell is continuously transmitting control information in order to define the time boundaries and declare the schedule for all users. Hence, LTE small cells are required to be continuously synchronized in order to have a practical distributed MIMO system.

Fortunately, the use of centralized scheduling and continuous synchronization alleviate and sometimes eliminate the challenges that exist for WiFi. For instance, continuous transmission and the ability to control the schedule means that the wireless channels can be tracked as opposed to having to bootstrap the channel estimates with every packet transmission. Further, the LTE protocol performs power control on the user devices in order to maintain the signals received from all users at a similar power level, which in turn eliminates the need for an AGC on the LTE small cell. Finally, due to centralized scheduling, the turn around time in LTE is more relaxed than in WiFi. Specifically, nodes have a few milliseconds to decode the data and turn around with an ACK. This gives more flexibility for the architecture of the transceiver design in the LTE case.

Although distributed MIMO for LTE small cells does not have the same challenges that exist in WiFi. It introduces its own set of challenges that need to be addressed.

Those challenges are mainly driven by the requirement of continuous synchronization, and the larger geographic span of LTE small cell networks.

Cellular operators expect massive deployments of small cells in 5G networks in order to meet capacity requirements, especially in dense urban settings such as Manhattan, downtown Tokyo *etc.* [12, 65, 25]. Such a distributed network naturally spans a large geographic scale, and experiences large amount of multipath fading and shadowing effects. To make the matter even worse, small cells are typically deployed on third-party premises with limited access and control for the operators of the network. Those properties of 5G small cells require LTE distributed MIMO systems to have a synchronization structure that is scalable, resilient and easy to manage.

Unfortunately, prior distributed MIMO systems for WiFi do not meet these synchronization requirements. Specifically, in WiFi distributed MIMO systems, the APs are typically organized around an architecture consisting of clusters, each with a single leader AP and multiple slave APs. All slaves listen to the leader signal and synchronize the phase of their signal to match that of the leader. Such an architecture is feasible in WiFi since WiFi networks do not have the same stringent scalability requirements of a cellular network, and are rarely deployed on third party premises. This architecture, however, is untenable for LTE small cells, and would prevent the network from achieving the key requirements of being scalable, resilient and easy to manage – it does not scale beyond the range of a single small cell, has a single point of failure, and requires frequent reconfiguration and full control over the location of the small cells.

In this thesis we introduce Chorus, a scalable distributed MIMO system for LTE small cells. In Chorus there are no special roles, *i.e.*, no leader and slaves. All Chorus nodes transmit a synchronization signal, and all synchronize their oscillator phases by listening to the synchronization signals transmitted by other nodes in their vicinity. There are also no special constraints on topology or connectivity. Thus, the coordination protocol is easy to manage and naturally scales to large networks with transmitters that are no longer in hearing range of each other. It also makes the system resilient to node failure, addition or removal. In order to achieve the above

properties and provide a practical distributed MIMO system for LTE, Chorus has to address the following challenges:

(a) Leaderless Distributed Synchronization: Each Chorus node transmits a synchronization signal and synchronizes with the composite synchronization signal that it hears. Naively applying this design leads to synchronization loops –e.g., a node may be synchronizing with a second node, that is synchronizing with a third node, which is synchronizing with the first node in the loop. Such loops are destabilizing, *i.e.*, they prevent the system from converging [42]. To prevent loops, Chorus has a distributed protocol to organize the nodes in the form of a tree (specifically a fat tree). In Section 5.3, we explain the protocol in detail and show how it is made resilient to failures including failures of the root of the synchronization tree.

(b) Robust Phase Tracking: In past systems, the phase difference is computed at the time of transmission and applied immediately for the duration of a single packet. In contrast, in LTE, the synchronization signal can only be sent sporadically due to the LTE frame structure to avoid large overhead and is required to maintain the synchronization continuously. This makes the synchronization task much harder. Furthermore, due to the leaderless architecture, the received synchronization signal is now a composition of synchronization signals sent from multiple nodes. It is likely for this combined signal to suffer from destructive interference at the receiver resulting in a low signal strength for synchronization. This in addition to the disturbance caused while nodes are joining and leaving the network. Chorus deals with these stringent synchronization requirements by combining techniques from signal processing and control theory, we discuss those techniques in Section 5.4.

(c) LTE Standard Compatibility: In order to provide a practical distributed MIMO for LTE small cells, all the techniques used by Chorus have to fit within the current LTE standard without introducing any changes to the LTE user devices. This puts extra constraints on the structure of the synchronization signal and the architecture of the system in general.

In this thesis, we address the practical limitations in distributed MIMO systems. We present algorithms and Hardware/Software architectures that enable practical distributed MIMO in the context of WiFi and LTE. Furthermore, we demonstrate real-time implementations of our designs on an FPGA platform.

Specifically, we present two different systems. For WiFi, we present MegaMIMO 2.0, the first real-time 802.11 distributed MIMO system. MegaMIMO 2.0 adds support for dynamic environments and mobile clients while meeting the tight timing requirements of the 802.11 WiFi standard. We also introduce Chorus the first scalable distributed MIMO system for LTE. Chorus works with the different modes of LTE and is resilient to changes in network configuration. Finally, MegaMIMO 2.0 and Chorus require no change to the user devices, and are fully compatible with the 802.11 standard for WiFi and the 3GPP standard for LTE.

The rest of the thesis is organized as follows, In Chapter 2 we give a brief background on distributed MIMO and discuss some of the main differences between the physical and medium access layers of WiFi and LTE. We discuss the related work in Chapter 3. In Chapter 4 we present the design, implementation and evaluation of MegaMIMO 2.0. Then similarly we introduce Chorus in Chapter 5. Finally, a conclusion is given in Chapter 6.

Chapter 2

Related Work

In this chapter we discuss the differences between the work presented in this thesis and the related work in literature. First we discuss some of the theoretical work done on MIMO in general and on Distributed MIMO in specific. After that we present the related work in literature for practical distributed MIMO in the context of WiFi and LTE. And finally, we show the difference between the synchronization required for distributed MIMO compared to the synchronization required by other distributed network protocols.

2.1 History of MIMO

Wireless communication technologies have been in continuous demand for higher data rates. Traditionally this demand has been achieved using methods such as providing more bandwidth and using higher modulation schemes. In the last few decades, multiple antenna systems (Multiple Input, Multiple Output - MIMO) were introduced and found to achieve a significant enhancement to data rate and channel capacity. In this section, we give some background on MIMO communication, and how it was developed over the past years.

Although some of the ideas concerning multi-channel digital transmission systems are sometimes traced back to the 1970s [35, 11, 68], it wasn't until the early 1990s when inventors started looking at the possibility of using multiple antenna systems

to transmit and receive different data streams on the same frequency at the same time [45].

In the late 1990s, researchers started looking at the channel capacity limits for MIMO systems and showing that the channel capacity for MIMO systems increases with the number of antennas, and is proportional to the smaller of the number of transmit antennas and receive antennas in the system. This was first established by the ground-breaking papers of Foschini [27, 28] and Telatar [64].

Briefly after that, many researchers were studying the design of channel codes for improving the data rate and reliability of wireless communication systems using MIMO. This resulted in a large body of theoretical work on MIMO systems, examples are Tarokh et al. [63, 62], Alamouti [4] and many others. After that many systems were introduced to show the gains of MIMO systems, as an example the Bell Labs Layered Space-Time (BLAST) architectures [27, 70, 54].

The gains from using MIMO are best achieved under the assumption of Independent and Identically Distributed (i.i.d.) channel fading model. However, when the scattering in the environment is insufficient, or when antennas are densely packed together in a small area, the channels between the different antenna pairs become correlated. And as a result, the capacity of such MIMO systems would not be proportional to the number of antennas, and gains would saturate after adding few antennas to the system.

The impact of correlated channel fading have been studied by a large body of research [59, 39, 8, 29, 38, 51, 23], this in addition to other research that studied the effect of the area and geometry of the antenna array on the degrees of freedom in a MIMO system [41, 47, 48, 31].

In order to deal with the issues related to correlated channel fading, and provide a MIMO system that could scale indefinitely with the number of antennas, researchers started studying the idea of Distributed MIMO [73]. Distributed MIMO is a promising technique which allows wireless systems to benefit from all the MIMO gains without having all the antennas on the same device. As a result it reduces the dependancies between the channels and allow for higher MIMO gains than what is achievable in

regular MIMO systems. We discuss the related work on distributed MIMO both in theory and practice in more details in the next sections.

2.2 Theoretical work on Distributed MIMO

There is some theoretical work [6, 66] that addresses distributed phase synchronization, but assumes frequency synchronous oscillators and only provides one-time phase offset calibration. Furthermore, The promise of distributed MIMO to improve the scalability of wireless networks has been explored in the theoretical community.

Jungnickel et al. [34, 33] used capacity estimates derived from channel measurements in an urban microcellular system, and demonstrated that distributed MIMO can provide a large gain in spectral efficiency through cochannel interference mitigation. Simeone et al. [60] studied the limitations imposed by capacity constraints on the performance gains provided by distributed MIMO.

In the context of ad hoc wireless networks, work by Aeron et al. [3] and Ozgur et al. [44] considered the throughput scaling laws for distributed MIMO in the case of dense networks. Aeron et al. [3] focused on networks with fixed Signal to Noise Ration (SNR), and introduced schemes to optimize the total network throughput. Ozgur et al. [44] showed that using distributed MIMO in such a setup can scale the total capacity of wireless ad hoc networks linearly with the number of nodes.

In the context of WiMAX networks, Venkatesan et al. [69] studied the use of distributed MIMO and quantified the spectral efficiency gains that can be obtained. In their work they considered realistic propagation models and operational conditions for a typical indoor deployment, and performed detailed simulations

Motivated by these theoretical results, recent years have seen significant research effort in moving distributed MIMO from theory to practice for both WiFi and LTE, and we discuss those system in the next two sections.

2.3 Distributed MIMO Schemes for Wi-Fi

There has been a significant research effort in providing practical distributed MIMO for WiFi [49, 5, 71, 2, 74, 71]. 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.

Also related to our work are papers studying the use of reciprocity for channel estimation [10, 30, 58]. The growth of massive MIMO systems has led to interest in scalable channel estimation techniques [57]. 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.

2.4 Distributed MIMO Schemes for LTE

The typical design of WiFi distributed MIMO systems assume a leader/coordinator that plays a special role in synchronizing the oscillators of the nodes. Such a design would not work at the scale and conditions of LTE as discussed earlier in Section 1.2. There are some previous schemes that have considered scaling beyond a single leader.

For example, NEMOx [74] considers multiple clusters each having its own leader

AP. The leader APs coordinate with each other to limit interference across clusters. However, such a system still is not resilient to the loss of leaders within a cluster, or changes in network topology that prevent adjacent leaders from coordinating, and is difficult to manage since it requires operators to determine the partitioning of the network into clusters and the corresponding leader assignment.

An extension to Airsync [52] proposes a hierarchical leader scheme that faces the same problems of scalability and resilience, and further, that system has only been proposed in theory and not evaluated empirically.

Vidyut [71] avoids the need for a single leader by allowing the Wi-Fi access points to synchronize their transmissions over the power lines, even if they are not within the same coverage area. Such a scheme applies to nodes within the same home or building, but does not scale to a larger network and does not apply to small cells where nodes are geographically dispersed and connected to different power systems. Furthermore, the reliance on power lines implies that the system is not resilient to power surges and noise on the power lines.

LTE has some mechanisms for loose GPS-based synchronization between devices, to enable joint transmission from multiple devices in nearby subcarriers, subcarrier suppression for inter cell interference cancellation *etc* [9]. Similarly, schemes have been proposed to achieve such inter-cell interference elimination in 802.11ac [72]. Unlike distributed MIMO, these schemes do not allow concurrent transmissions in the same frequencies at the same time. They only limit interference between adjacent frequencies. Hence their throughput gains are much lower than distributed MIMO. Other LTE systems based on DAS and CoMP has been widely used to provide better coverage and improve te network performance for in-door scenarios and around cell edges. However, current solutions based on DAS or CoMP fail to deliver the benefits of a truly distributed MIMO system.

A popular recent trend in LTE is massive MIMO [57, 56, 32, 40]. For instance, systems such as Argos [57] have demonstrated designs where a large number of MIMO antennas are packed densely on a single node. distributed MIMO is complementary to Massive MIMO. Specifically, massive MIMO is typically applicable to base stations

or macro cells, which can accommodate the size and power requirements of the large number of antennas and their corresponding digital and analog chains. In contrast, Chorus is targeted at small cells, which have a much smaller form factor and lower power budget.

2.5 Other Related Work

There is a significant literature on a variety of coordination, consistency, and consensus protocols in the distributed systems literature, such as Dynamo, Paxos, Spanner, and Raft [14, 43, 36, 37]. These systems are typically used to determine a leader among a set of servers in a distributed system, and agree in a distributed manner on states of a state machine. These systems are however fundamentally different in scope, and timescales, from distributed MIMO. Specifically, in these distributed protocols, the state of the system is determined only by interactions between the servers in the distributed system, and advances or rolls back based on messages exchanged between them. In contrast, the goal of distributed MIMO is to track an analog state, specifically, the phase of a reference oscillator, which is constantly changing independent of the interaction between the nodes in the system. As a result, the role of the protocol is to ensure that all nodes in the system track this reference oscillator accurately within tens of nanoseconds.

Chapter 3

Background

3.1 Distributed Multi-User MIMO Systems

Unlike wired communication the wireless medium is shared. If two different transmitters try to access the wireless medium at the same time and on the same frequency their signal will interfere and a receiver will not be able to get the right data. As a result wireless system tend to divide the resources between them in time or frequency or both.

Using MIMO in wireless communication introduces a new dimension to be used which is the spatial dimension. When a communication system has multiple transmit and multiple receive antennas, like the system shown in Figure 3-1, those antennas increase the degrees of freedom in the system and allow for spatial multiplexing.

Since each antenna on the receiver side is capable of hearing the antennas on the transmitter side multiplied by some channel h, under the condition that all the channels are known and form an independant set of equations, the receiver would be able to recover different data streams sent on the different antennas of the transmitter. So for the example shown in Figure 3-1, this system would be capable of sending three different data streams at the same time and on the same frequency. In such a system where all the transmit antennas are on one device and all the receive antennas are on another device, it is called point-to-point MIMO or Single User MIMO.

The most obvious limitation to single user MIMO is that the multiple streams of

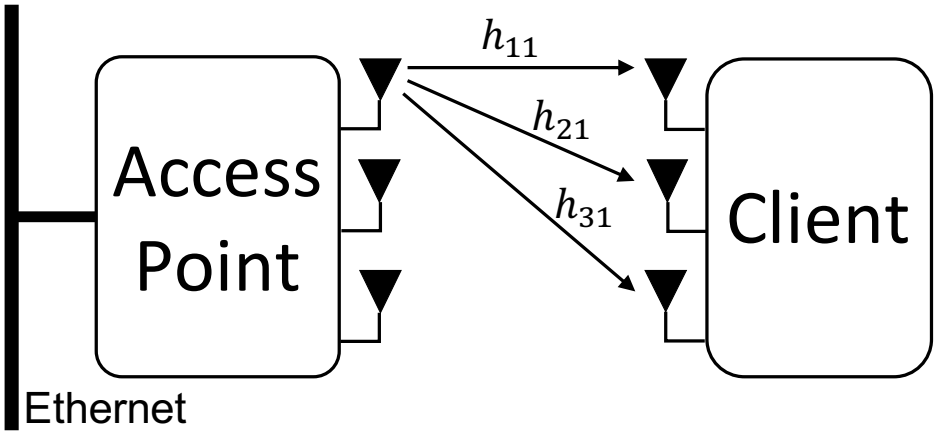


Figure 3-1: **Single User MIMO.**

data have to be sent to or received from one device at a time. Not only that, but the number of streams that can be sent is limited by the minimum number of antennas on the transmitter and the receiver. For example, if the transmitter (Access Point) has 4 antennas and the receiver (phone) has 2 antennas, such a system is only capable of sending 2 streams at a time.

Due to the unsemetric formfactor constraints on the Access Points (APs) and the user devices (cellphone, tablet, etc.), APs typically have larger number of antennas than the number of antennas on a user device. However, the achievable gains from those antennas becomes limited by the number of antennas on the user device. In order to overcome this limitation, we need to use a different technology called Multi-User MIMO.

Figure 3-2 shows an example of a Multi-User MIMO system. The way Multi-User MIMO works is that instead of performing the channel estimation and decoding at the receiver, it is performed at the transmitter. So the transmitter is responsible for precoding the signal transmitted on each of its antennas in a way that when it combines at a given user only the data intended for that user survives and the data for all the other users cancels out. And in order for the transmitter to perform the precoding operation, it needs to know the channel between the different transmit and receive antennas. And that could be either measured at the user devices then fed back to the transmitter, or estimated directly at the transmitter using reciprocity.

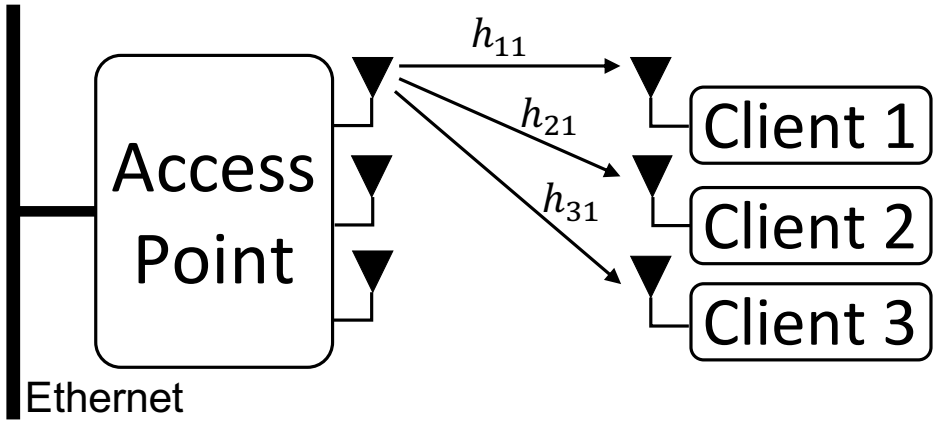


Figure 3-2: **Multi-User MIMO.**

Multi-User MIMO allows a wireless system to utilize all the antennas on the access points even when the users are having less number of antennas. However, the gains from such a system are still limited by the number of antennas that can be placed on a single access point. And in many cases, because the multiple antennas are placed in close proximity of each other their channels become dependent and reduce the degrees of freedom that can be achieved by the system.

In order to reduce the dependancies between the channels, and provide a MIMO system that is capable of scaling indefinitely with the number of antennas, we have to use another technique that is called Distributed Multi-User MIMO, or we can just call it Distributed MIMO. Figure 3-3 shows an example of Distributed MIMO. The main

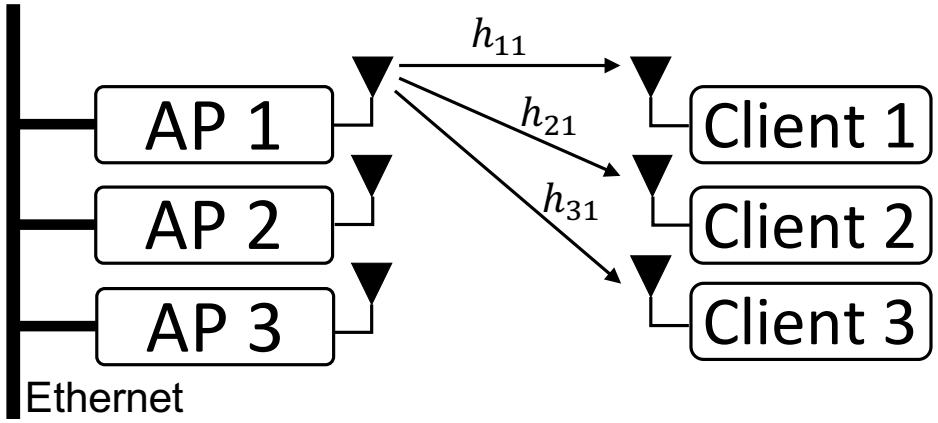


Figure 3-3: **Distributed Multi-User MIMO.**

difference between this system and the one shown in Figure 3-2, is that in Distributed MIMO the antennas are distributed across multiple access points which reduces the dependance between the channels.

The main challenge in Distributed MIMO is that in order for the different access points to be able to transmit and receive at the same time, they need to be perfectly synchronized in time, frequency and phase. And in this thesis we present algorithms and implementations that are capable of synchronizing multiple nodes and showing the gains of Distributed MIMO in the context of WiFi and LTE.

3.2 Differences between WiFi and LTE

It is essential for any proposed technique to be able to fit within the current framework without major changes. Techniques that require significant changes to the standard or even minor changes to the user devices, will need decades to be deployed and in many cases will end up not deployed at all. In order to understand what it takes to integrate distributed MIMO in todays WiFi and LTE networks, in this section we present some background about WiFi and LTE, and discuss some of the differences in the signaling structure between the two standards.

Before we discuss the challenges for Distributed MIMO systems, it is important to understand some of the differences between WiFi and LTE. Understanding those differences will help us set the goals for our systems, and optimize our resources in the

	WiFi	LTE
Network Scale	Local Area Network	Wide Area Network
Coverage Range	outdoor $< 100m$ indoor $< 50m$	macro cells $10s\ km$ small cells $100s\ m$
Duplexing	TDD only	TDD or FDD
Medium Access	Distributed Random Access	Centralized Scheduling
Signal Structure	Packets	Continuous Transmission
Power Control	APs and User devices	User devices only
Turnaround Time	Very tight	More relaxed

Table 3.1: **Significant differences in signaling structure between WiFi and LTE.**

right direction. Table 3.1 shows a summary for the differences that we are discussing in this section.

3.2.1 Network Scale & Coverage Range:

Driven by regulations on the ISM bands and specifications from the WiFi standard, the expected coverage provided by a single WiFi Access Point (AP) would not exceed few $10s$ of meters in the best indoor scenario. On the other hand, an LTE Base Station (BS) is expected to have more than one order of magnitude larger coverage range from a single small cell. Although this might look like a simple scaling problem, it is not the case for distributed MIMO. In order for distributed MIMO to scale the network throughput with user demand, it has to increase the number of APs (for WiFi) or BSs (for LTE) to increase the total throughput of the network. As a result, a distributed MIMO system for LTE is expected to have a larger number of BSs and span a larger geographic area, i.e. multiple buildings or even a city. In other words, it is acceptable in a distributed MIMO system for WiFi to be structured in the form of different clusters of APs, having each cluster using a different channel from the ISM band, and covering an office area or one floor of a building. In LTE however, this would simply not work.

3.2.2 Uplink/Downlink Duplexing:

In WiFi an AP shares the same frequency channel with all user devices connected to it. The APs and user devices divide the resources between them in a Time Division Duplexing (TDD) fashion. Which means that in WiFi both Tx and Rx of all devices are expected to be using the same frequency channel. LTE however, comes in two different flavors, one of them is TDD (similar to WiFi), and the other one is Frequency Division Duplexing (FDD). In FDD, the BSs are transmitting on a different frequency channel than that used for transmission by the user devices. In other words, there is a specific channel for signals going from the BSs to the user devices called DownLink (DL) channel, and a different frequency channel, UpLink (UL), for signals from the

user devices back to the BSs. Hense, in the FDD mode of LTE the Tx and Rx of the BSs are not using the same frequency channel. And that introduces some extra challenges in the case of LTE that we will discuss later.

3.2.3 Medium Access Control (MAC):

Since in both WiFi and LTE multiple devices are expected to be sharing the same medium, there need to be a protocol that grants access for those devices. And that is called the MAC. In WiFi, devices use a distributed random access protocol called Carrier Sense Multiple Access (CSMA). The basic idea in CSMA is that when a device needs to access the medium it has to first listen on the medium and make sure the medium is free, only then it will start transmitting. In LTE, the case is totally different, since network providers are paying to license a channel, they get exclusive access to that channel. Hence, it is guaranteed that all the devices using this channel belong to the same network controlled by that provider. So, in order to minimize the losses, LTE uses a centralized scheduling technique. A central unit is responsible for defining the schedule ahead of time, so each node will know exactly when to transmit and when to expect data.

3.2.4 Signal Structure:

In WiFi the signaling unit used is a packet. Once a device obtains the access to the medium using CSMA, it will transmit a packet or a sequence of packets. And once it is done transmitting, it has no guarantee on when it will gain access to the medium again. And for a WiFi device receiving a packet, it would not know which device is sending this packet, or what is the purpose of it until the packet is fully received and decoded. On the other side, LTE uses a frame structure. An LTE BS will be continuously transmitting control signals inside that frame structure, that helps all devices to track the state of the frame, and know when is the right time for them to transmit or receive.

3.2.5 Power/Gain Control:

In WiFi, in order to provide a reasonable coverage range without increasing the requirements on the hardware. A WiFi AP would use an AGC on its receiver to adapt for the difference in power for signals received from different devices. And since the medium access is random, a WiFi receiver will have to start fresh for every packet to learn what is the proper gain to be used for this specific packet. In LTE gain control is relatively slow, and is performed only on the user devices. A BS would provide a feedback to the user equipment to help it control its transmit power so that signals from all users are received with similar power level at the BS.

3.2.6 Turnaround Time:

In WiFi, the turnaround time for the MAC layer Acknowledgement (Ack) is very tight. A node is expected to turnaround and provide an Ack within 10 microsecond from receiving the last sample in a packet. Moreover, if the medium stays idle for more than 30 microseconds it is likely that another device will obtain the medium and start transmitting. This puts harsh requirements on the coordination between different nodes in a distributed MIMO system for WiFi. In LTE, that requirement does not exist, the Ack could be provided milliseconds after the reception of the data, and the medium access is guaranteed by the centralized scheduling.

Chapter 4

MegaMIMO 2.0: Distributed MIMO for WiFi

4.1 Introduction to MegaMIMO 2.0

It is important for a new technique to fit nicely within the target protocol without significant changes to the standard or user devices. One of the prior work on distributed MIMO for WiFi, MegaMIMO [49], has achieved a significant advancement in that direction. MegaMIMO was designed to address the idea of synchronizing independant WiFi APs and allowing them to jointly transmit to multiple user devices. It does so by organizing the network into a single leader AP and multiple slave APs. The slave APs listen to the leader packets, and use them to correct for the accumulated phase change on their oscillators. They also use those packets as a trigger for the joint transmission.

Although MegaMIMO was successful in solving the synchronization problem in WiFi without any changes to the user devices. It did not address other important challenges that arise once we start considering a real-time implementation of the system. In this chapter we describe the design and implementation of MegaMIMO 2.0, the first real-time 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. Further, in MegaMIMO 2.0 we address problems related to power control and we

present a software/hardware architecture that enables the new functionalities required for synchronization. In the rest of this section we discuss the critical limitations in MegaMIMO, and show how we solve them in MegaMIMO 2.0.

4.1.1 Channel feedback overhead

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 [46]. 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 for WiFi (8 bits magnitude and phase for all OFDM subcarriers, 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. 4-1 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

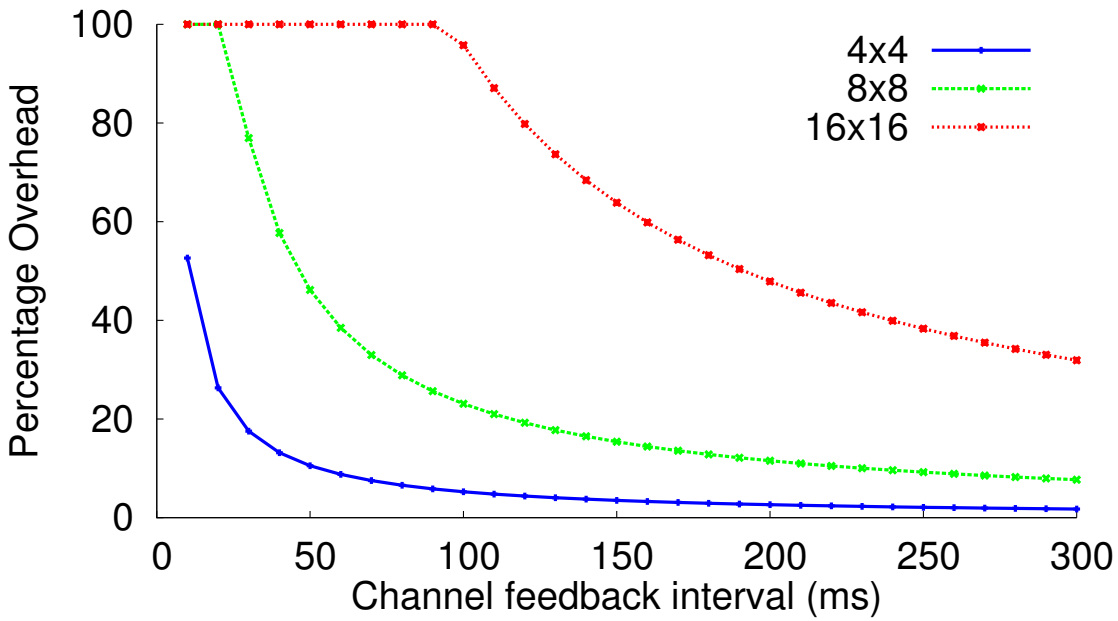


Figure 4-1: **Channel feedback overhead for distributed MIMO system.** The figure shows that at typical coherence times of about 100ms, the channel feedback overhead can significantly limit the gains of distributed MIMO systems, particularly as they scale to more nodes. For mobile clients with lower coherence times, the overhead is even higher.

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 Section 4.2, we demonstrate that in a WiFi 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 MegaMIMO 2.0.

4.1.2 Power control

Practical WiFi systems all use Automatic Gain Control (AGC), an analog module that dynamically adapts for the difference in power of signals received from different devices. 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.

4.1.3 Rearchitecting the baseband and firmware

Distributed MIMO requires redesigning the firmware-hardware interface. Event timing in the existing WiFi 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 WiFi. 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 present a comprehensive system design that delivers the first real-time dis-

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

tributed WiFi MIMO system. Our design provides a full-fledged distributed MIMO PHY layer that address the above challenges, delivering seamless channel adaptation, coordinated power control, and a refactored implementation of the 802.11 baseband and firmware to incorporate distributed coordination. 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 evaluate 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.
- 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.

4.2 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. 4-1. The 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.

4.2.1 Reciprocity in Traditional MIMO

Let us consider the simple example in Fig. 4-2 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

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

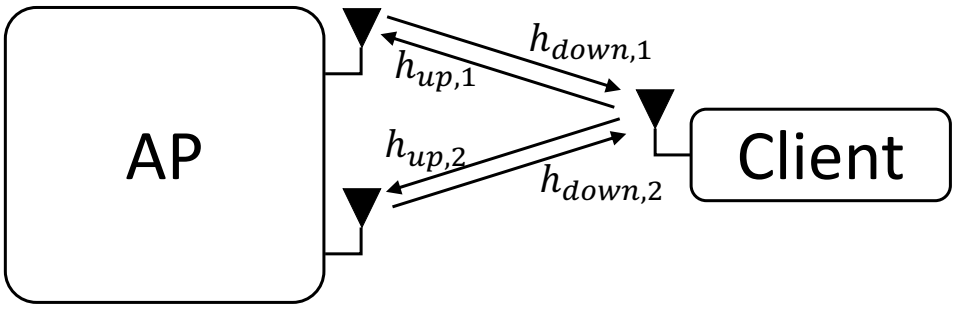


Figure 4-2: **A 2-antenna AP transmitting to a 1-antenna client.** With reciprocity based channel estimation, the AP would need to estimate the downlink channels $h_{down,1}$ and $h_{down,2}$ from the uplink channels $h_{up,1}$ and $h_{up,2}$.

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] [67]. 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.

4.2.2 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

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

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 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

$$C_i(t) = C_i(0) \exp(j2\Delta\omega_{21}t) \tag{4.1}$$

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. 4.1. 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. 4.1 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.

4.2.3 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. 4.1 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 [49], 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 [49] 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 [49]. 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 [49].

4.3 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

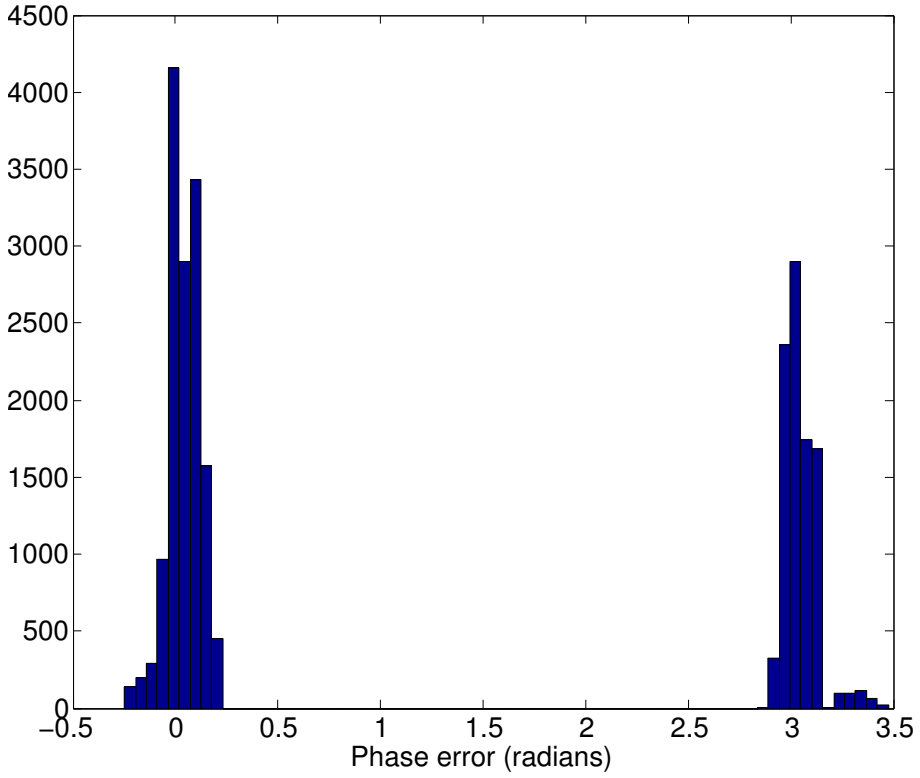


Figure 4-3: **Histogram of the difference in phase between different AGC gains.** The figure shows that there are very large differences in phase across the different gain settings, in fact as large as π radians. Different gain ranges involve activation of different elements of the analog RF front-end, and hence can introduce significantly different phase shifts.

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.

4.3.1 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.

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. 4-3 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 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 Chapter 4.2.3.

4.3.2 Coordinating AGC across space

As described in Chapter 4.2.3, 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

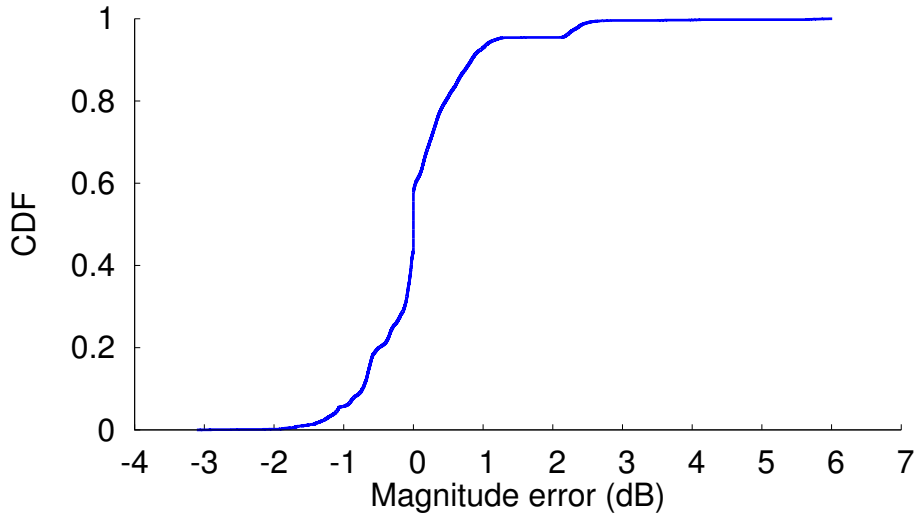


Figure 4-4: **CDF of the difference between nominal and actual AGC gains.** The figure shows that there are significant differences between nominal and actual gains, sometimes as large as 6-7 dB.

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 accurate, 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. 4-4 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.

4.3.3 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.

4.4 MegaMIMO 2.0 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 thesis. This thesis 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.

4.4.1 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 interface 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.

4.4.2 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.

4.4.3 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. 4-5 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.

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

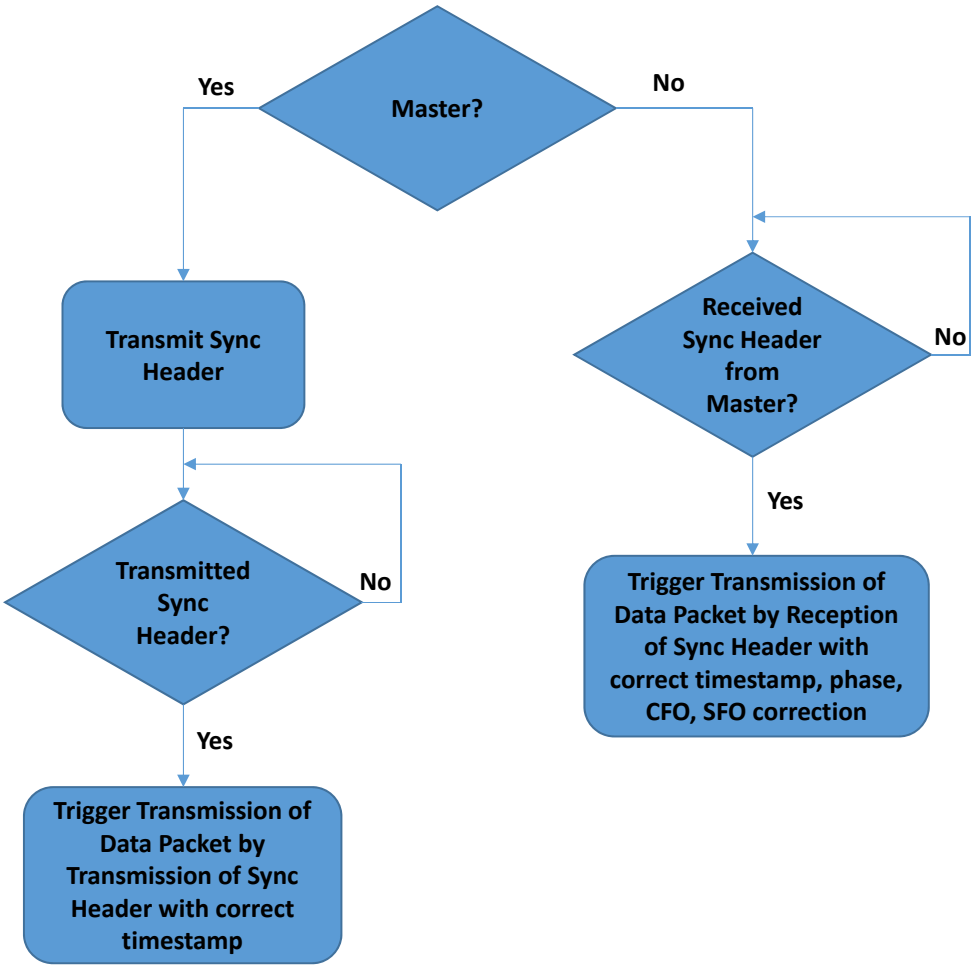


Figure 4-5: **Timing, Frequency and Phase Synchronization Subsystem.** This subsystem of the MAC operates in real-time. It interacts with the PHY and triggers transmission of packets based on transmission or reception of sync headers. It also applies the correct frequency and phase correction at the slave APs for the joint transmission.

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 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.

4.5 Implementation

We implement MegaMIMO 2.0 and evaluate it in an indoor testbed. In this section we discuss the platform the we use to implement our system and the hardware architecture of our implementation.

4.5.1 Hardware Platform

Each node in our system consists of a Zedboard connected to an Analog Device FMCOMMS3 transceiver card both shown in Figure 4-6. The Zedboard is equipped

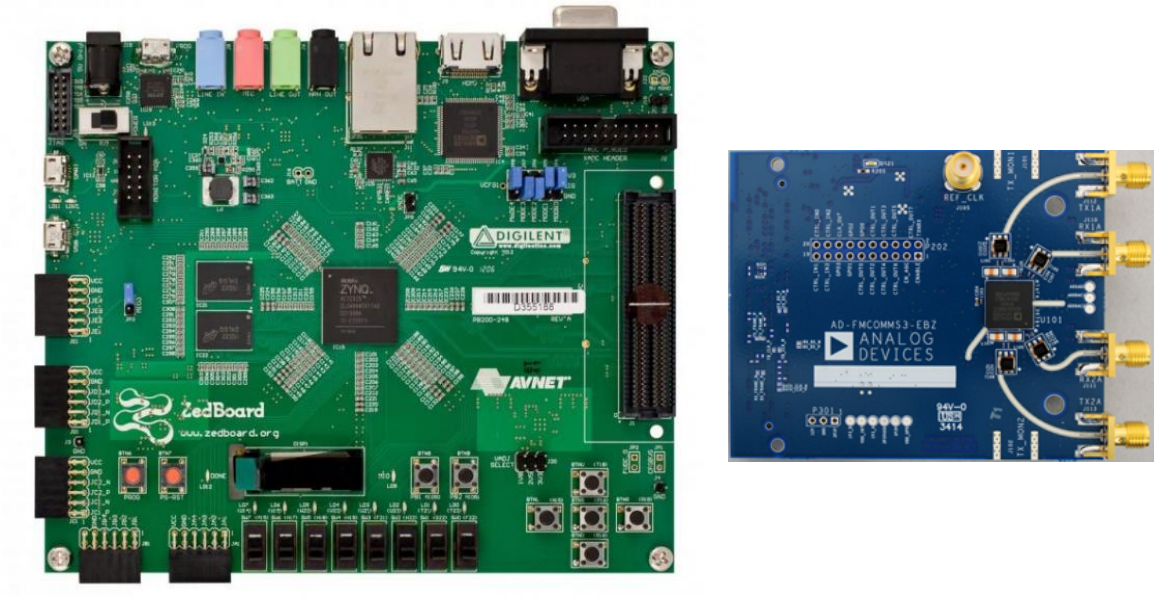


Figure 4-6: **Hardware Platform** The figure shows the Zedboard on the left and the FMCOMMS3 board on the right.

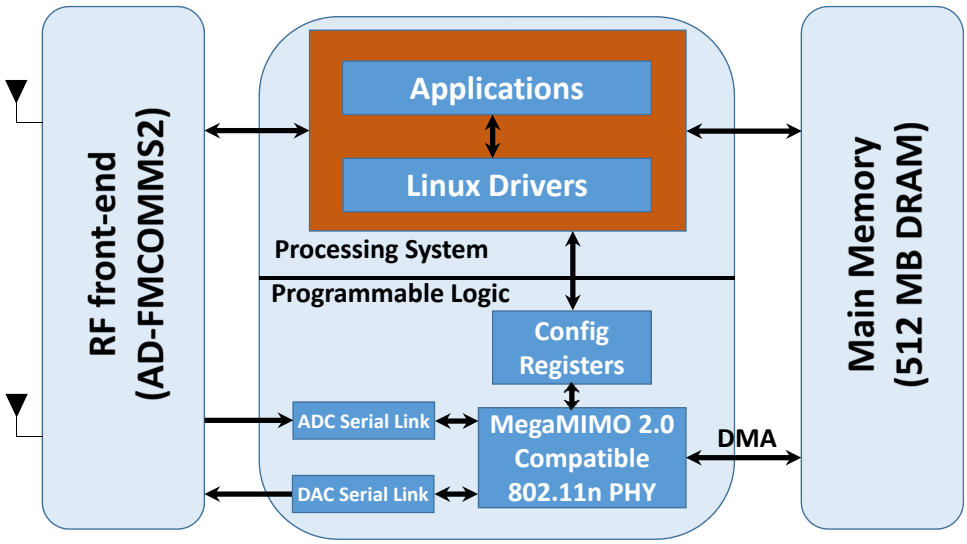


Figure 4-7: **Platform Architecture** The figure shows the software-hardware architecture of our platform. The PHY on the FPGA implements an 802.11n MIMO system as well as the real time synchronization facilities needed for distributed MIMO. The software on the ARM core configures the PHY and manages data transfer to and from the device.

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. Figure 4-7 shows the high level architecture of our system and the interfaces between the different components of our 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 FMCOMMS3 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.

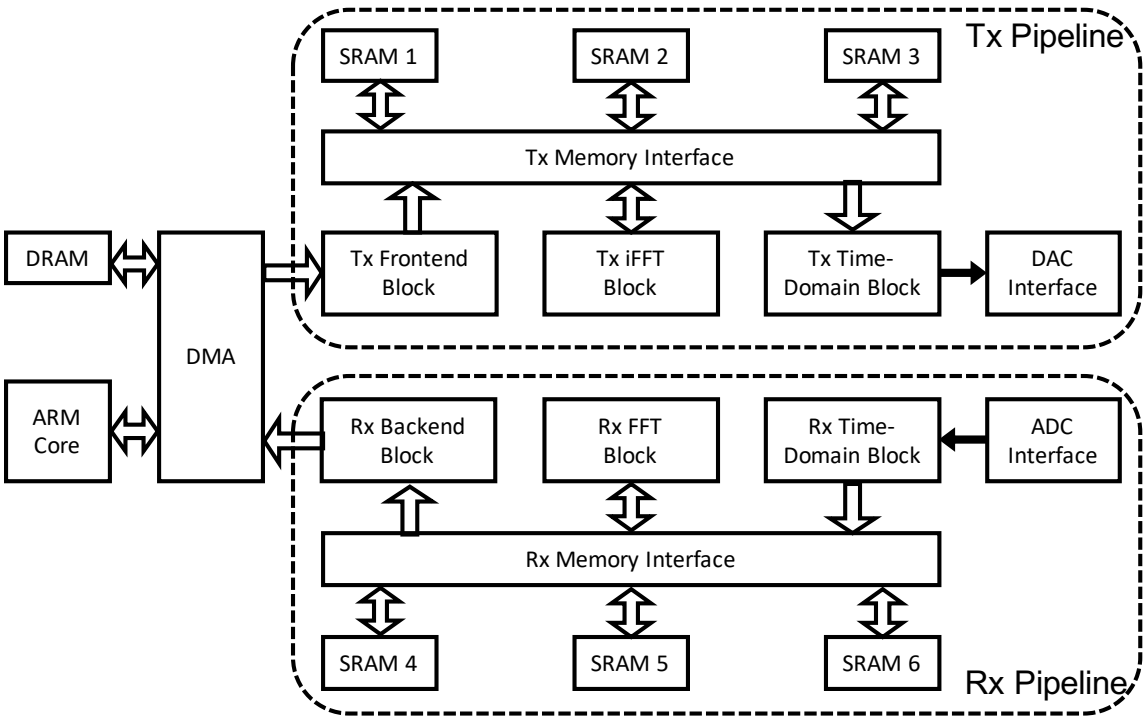


Figure 4-8: **MegaMIMO 2.0 Transceiver Architecture** The figure shows the high level architecture of our MegaMIMO 2.0 transceiver.

4.5.2 Hardware Architecture

Figure 4-8 shows the high level architecture of the MegaMIMO 2.0 transceiver. The upper part of the figure shows the transmitter pipeline and the lower part shows the receiver pipeline. The transmit packets are stored by the ARM core in a shared memory section in the DRAM, and then the transmitter pipeline directly reads it from the DRAM through the DMA. Similarly, the receive packets are directly stored in the DRAM by the receiver pipeline, and then read and processed after that by the ARM core.

Both transmitter and receiver pipelines are three stages symbol based pipeline each. Which means that during the transmit/receive operations, it handles three different OFDM symbols concurrently. All the three stages of the pipeline are designed to be capable of finishing the symbol processing in less than the symbol time of the 802.11 standard, which is $4\mu s$. The data is passed between the different pipeline stages through shared SRAMs that are rotated across the different pipeline stages.

FFT and iFFT Blocks

The MegaMIMO 2.0 transceiver design has two different FFT blocks, one for the transmitter to perform the iFFT operation and the other one is for the receiver to perform the FFT operation. The FFT size is a 128 points and the bit-width is 16-bits real and 16-bits imaginary. The FFT implementation uses an in place architecture with a block floating point. The block floating point technique allows each FFT frame to have a different exponent to prevent overflow and increase the dynamic range, but at the same time avoids the complexity of the floating point design by sharing the same exponent across all the samples inside the frame.

Tx Frontend Block

Figure 4-9 shows the Tx frontend block. Based on the size of the FPGA, our current implementation supports up to 4 distributed transmitters transmitting simultaneously to 4 independent clients. The DMA reader block is responsible for reading the data for all the users from the DMA and making sure that the user buffers always has data ready for processing. It is also responsible for for reading the precoding Matrix. Each user has a separate pipeline to perform the scrambling, encoding and interleaving

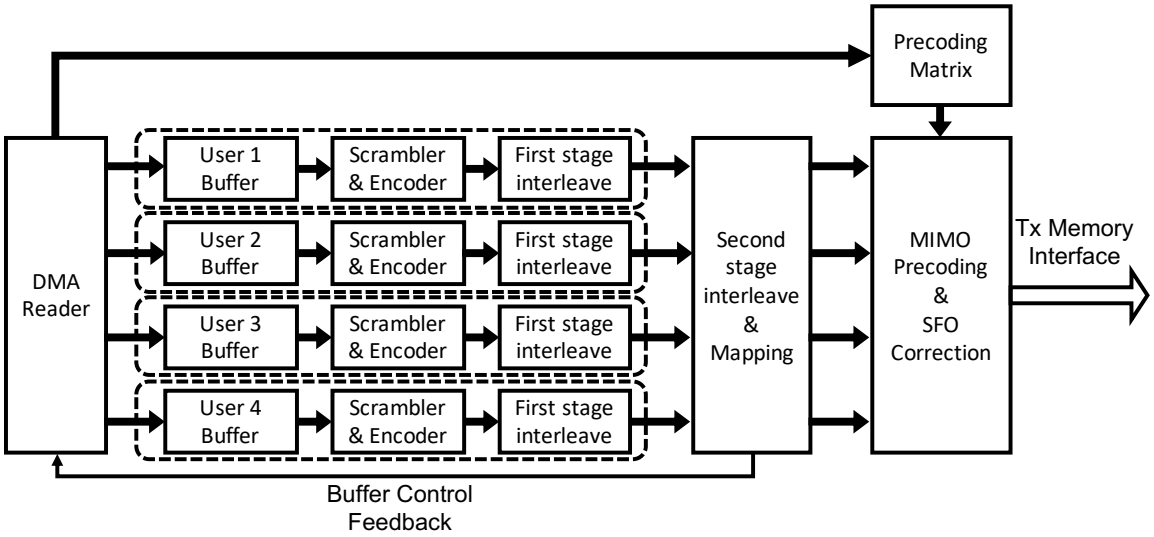


Figure 4-9: **Tx Frontend Block** The figure shows the architecture the Tx Frontend block.

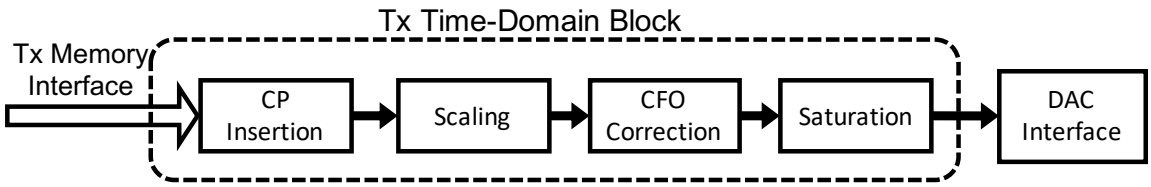


Figure 4-10: **Tx Time-Domain Block** The figure shows the architecture the Tx Time-Domain block.

as specified by the 802.11 standard. The different streams are then combined in the second stage of the interleaving to reduce the memory overhead, and in that block the bit mapping and pilot insertion is done. This block is also responsible for controlling the flows for the different users through a feedback to the DMA reader block. Then the output of the mapping is passed to the precoding block where the 4 streams are multiplied by the precoding matrix then SFO correction is applied on the different subcarriers using the MegaMIMO 2.0 estimated frequency offset. Finally, the corrected subcarriers are then written to the shared memory to be processed by the iFFT block.

Tx Time-Domain Block

Figure 4-10 shows the Tx Time-Domain block. This block reads the output of the iFFT which corresponds to the time-domain samples. It appends the Cyclic Prefix (CP) to it as specified by the 802.11 standard supporting both normal and short CP modes. After that, the scaling block accounts for the block floating point exponent and multiply by a packet specific magnitude scale. Then the output of the scaling block is passed through the CFO correction block where it accounts for the frequency offset measured in the MegaMIMO 2.0 algorithm. Finally, before the signal is send to the DAC interface, it is passed through a saturation block where it performs digital clipping.

Rx Time-Domain Block

Figure 4-11 shows the Rx Time-Domain block. The first block in the pipeline of the receiver is the packet detection. Our packet detection algorithm is based on

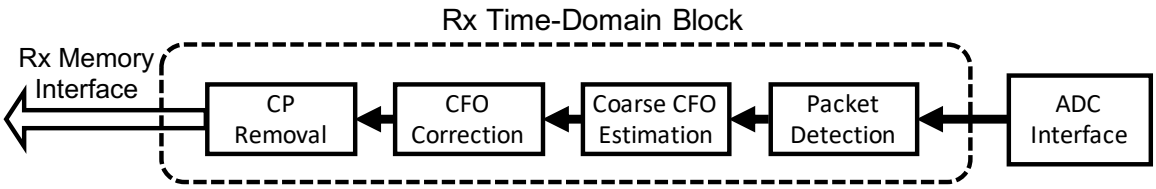


Figure 4-11: **Rx Time-Domain Block** The figure shows the architecture the Rx Time-Domain block.

autocorrelation. The output of the auto correlation is also used for coarse CFO estimation. After that CFO correction is performed on the time domain signal using the coarse CFO estimation. The cyclic prefix is removed, and the data gets buffered in the shared memory at the granularity of symbols to be processed by the Rx FFT block.

Rx Backend Block

Figure 4-12 shows the Rx backend block. The Symbol Reader block reads the output of the FFT from the shared memory and it consumes one subcarrier every 4 clock cycles. If the current symbol is from the channel estimation header it forwards it to the channel estimation block, otherwise the subcarrier is fed through the normal pipeline. The second stage in the pipeline is the frequency offset correction, and this stage performs compensation for the SFO as well as the residual CFO. The output of the

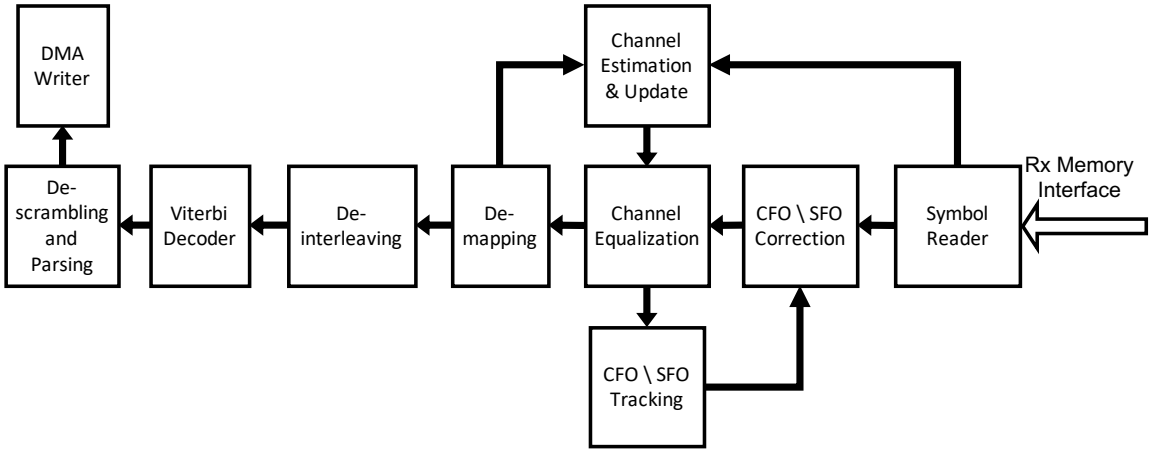


Figure 4-12: **Rx Backend Block** The figure shows the architecture the Rx Backend block.

frequency offset correction is then fed to the channel equalization block which divides by the estimated channel. Then if the current subcarrier is a pilot subcarrier, it will be routed to the CFO tracking block to update the estimate of the frequency offset, otherwise it forwards it to the de-mapping block. The de-mapping block performs soft-decision de-mapping and produce a 3-bit value for each transmitted bit, then it feeds that through to the de-interleaver. It also remaps a hard-decision of the bits and feeds it back to the channel estimation block to update the estimate of the channel. The de-interleaving block undo the interleaving operation that was done at the transmitter and feeds the soft bits to the Viterbi decoder. After that the output of the decoder is de-scrambled and parsed to remove the PHY headers and check the CRC. Then finally the data of the packet will be written to the DRAM to be post processed by the software.

FPGA Utilization

Table 4.1 shows the resource utilization of our real-time PHY and MAC implementation on our current FPGA platform.

4.6 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

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

Table 4.1: **FPGA Utilization on Xilinx Zynq Z7020**. This table shows the utilization of different FPGA elements by our real-time PHY and MAC implementation.

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 traditional 802.11 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 [49].

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.

4.6.1 Accuracy of Reciprocity

Reciprocity eliminates the overhead of channel feedback, which tends to be excessive in distributed MIMO systems (see Fig. 4-1). 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.

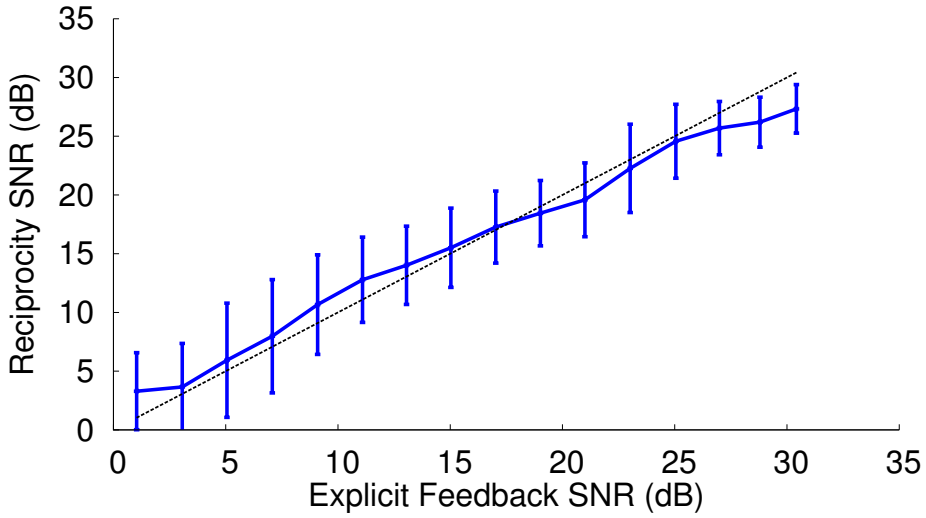


Figure 4-13: **SNR with reciprocity (MegaMIMO 2.0) and explicit feedback (MegaMIMO) based distributed MIMO systems.** The 45^o degree line is shown in dotted black. The figure shows that reciprocity based distributed MIMO can achieve the same SNR as explicit feedback across the range of SNRs.

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. 4-13 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 thesis to avoid the excessive overhead of explicit channel feedback.

4.6.2 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

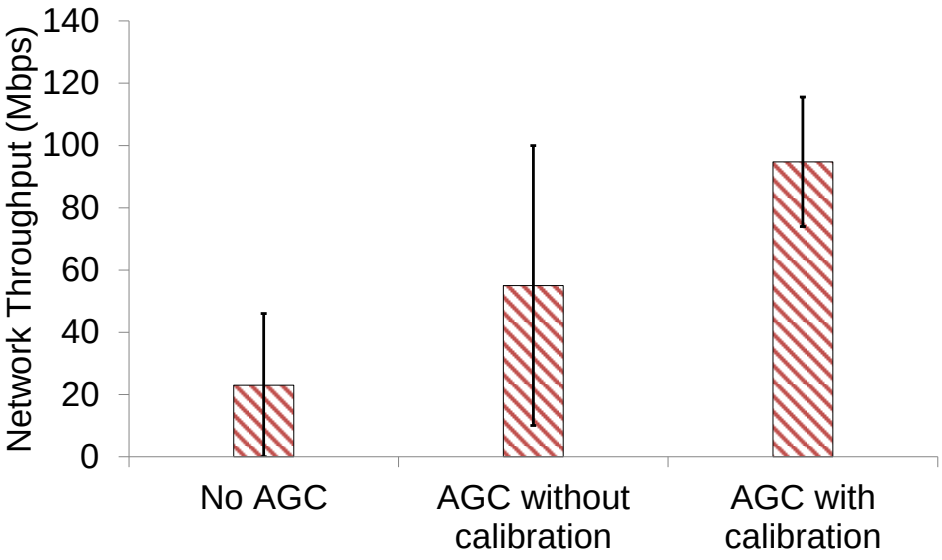


Figure 4-14: **Throughput comparison of MegaMIMO 2.0 with full AGC calibration, MegaMIMO 2.0 using AGC without calibration, and MegaMIMO (fixed gain).** The figure shows that distributed MIMO needs the use of AGC with full calibration in order to achieve high gains with low variance.

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. 4-14 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 4.3.2), 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 gain during channel estimation at any of the APs (the impact of gain errors is analyzed in 4.3.2 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.

4.6.3 Real-time Performance

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

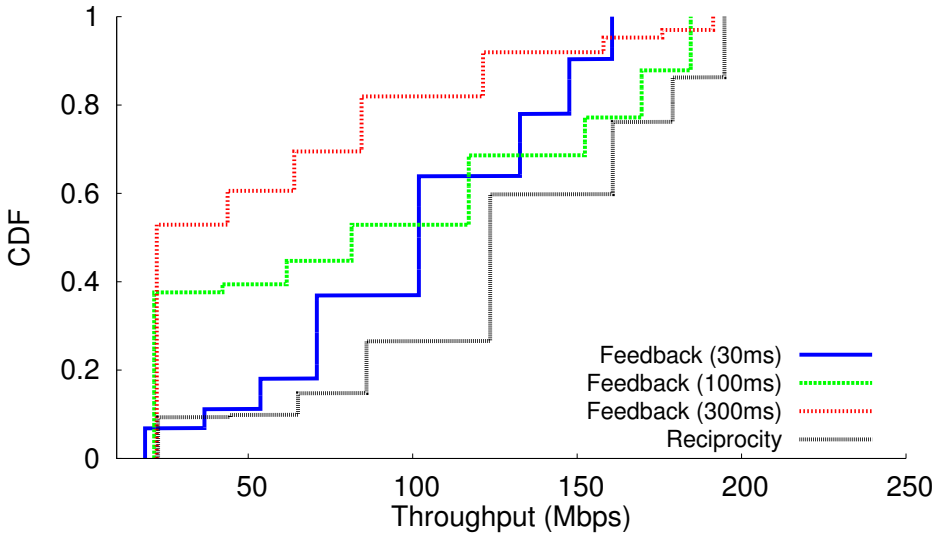
Method: We consider dynamic environments common in indoor settings. We

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].

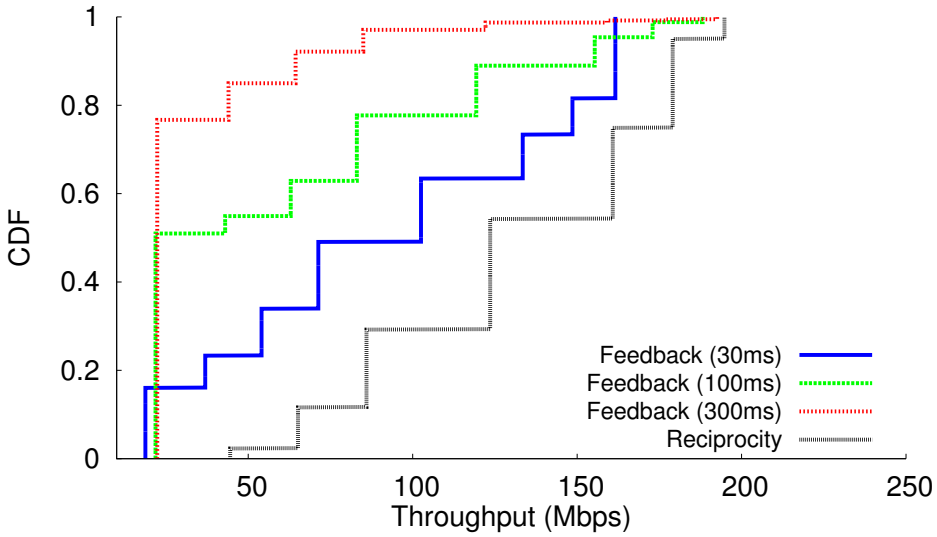
We deploy 4 nodes acting as APs in our testbed, and 4 nodes acting as clients. We compare two schemes: distributed MIMO with explicit channel feedback, and distributed MIMO with reciprocity. Note that both schemes are running MegaMIMO 2.0's real-time PHY with all of its components (AGC, calibration *etc.*, 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. 4-15(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



(a) Environmental Mobility



(b) Client Mobility

Figure 4-15: **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.

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 staleness. 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. 4-15(a) and (b) is worse than the simulation results of a 4×4 system in Fig. 4-1. 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.

4.6.4 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. 4-16 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.

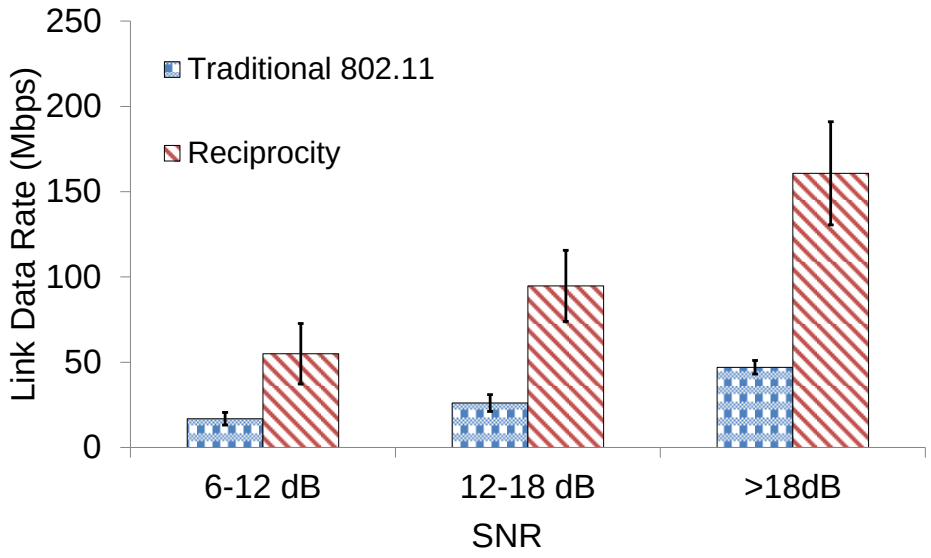


Figure 4-16: **Comparison of throughput obtained with traditional 802.11, and MegaMIMO 2.0 with reciprocity.** The figure shows that MegaMIMO 2.0's implementation can scale throughput linearly with the number of nodes. At all SNRs, the throughput of MegaMIMO 2.0 with 4 nodes is 3.6× the throughput of a single 802.11 link.

Chapter 5

Chorus: Distributed MIMO for LTE

5.1 Introduction to Chorus

In the context of LTE, prior work on distributed MIMO can be divided into two main categories.

- The first category relies on a central unit that is responsible for constructing and receiving the RF signals for all the antennas. An example for this approach is Distributed Antenna Systems (DAS) [53]. In DAS the RF signals are distributed inside buildings or across the area of intrest through high quality RF cables and using splitters and power amplifiers to overcome the losses. Despite the expensive infrastructure and very limited throughput gains, DAS based solutions and market has witnessed a significant growth over the past few years, and is expected to reach a market worth of 10 Billion USD by 2022 [13]. This is purely driven by more use of smartphones and tablets etc., and the increasing demands on better coverage and high speed connectivity.
- The second category tries to achieve better coverage by distributing the transmission functionality across multiple devices. The concept of Coordinated MultiPoint (CoMP) was introduced in the 3GPP LTE standard several years ago to improve network performance at cell edges. And different modes where added to CoMP in order to benefit other scenarios of coordination across small cells. Examples of

systems that try to benefit from CoMP are PCell [26, 1] and a demonstration by Ericsson [15]. These systems assume a shared clock across all nodes either via GPS or a wire. As such, these systems do not deliver the benefits of a truly distributed design as we will discuss in the rest of this section, and they wouldn't work in scenarios where the GPS signal is poor like tunnels and dense urban areas.

Both categories described above fail to deliver the promised gains of distributed MIMO in the context of LTE. The question that now arises is, can we use techniques similar to what was proposed by MegaMIMO 2.0 for WiFi to provide distributed MIMO for LTE?

Although the basic idea of synchronization is very much the same in WiFi and LTE, the challenges to have a practical distributed MIMO system are not. The techniques described earlier to provide distributed MIMO for WiFi were mainly focused on enabling the transmission functionality of signals to be distributed, which was not an easy task given the random access nature of WiFi. As a result, the control plane which coordinates the transmitters became fairly centralized. Specifically, in the WiFi case, the transmitters are typically organized in clusters, each with a single leader and multiple slaves. All the slaves listen to the leader signal and synchronize the phase of their signals to match that of the leader. Such an architecture prevents distributed MIMO networks from enjoying key desirable features in an LTE network: scalability, resilience, and ease of management. Further, the clustering architecture used in WiFi introduce another limitation on the system, since now the system has to deal with interference across clusters. That issue could be avoided in the WiFi case by using three different frequency channels of the ISM band - we discuss that in more details in Section 5.2. In LTE however, this would tripple the cost of the spectrum license, which is clearly undesirable by the network providers.

In this thesis, we introduce Chorus, a system the removes these limitations and builds a scalable distributed MIMO system for LTE. In rest of this section we will discuss the challenges for distributed MIMO in the context of LTE in more details and show how we address them in Chorus through the rest of this thesis.

5.1.1 Scalability, Resilience and Manageability

Bringing scalability to distributed MIMO is particularly important for 5G small cell networks. All major cellular equipment manufacturers and operators expect massive deployment of small cells in 5G in order to meet capacity requirements, especially in dense urban settings, such as Manhattan, downtown Tokyo *etc.* [12, 65, 25]. Such dense small cell deployment will naturally increase the interference between transmitting nodes, and emphasize the need for distributed MIMO, which both eliminates interference and increases throughput. Further, such small cell networks will naturally span large geographic scale (e.g., Manhattan), and hence will need a distributed architecture that does not assume the presence of one node that can be heard by all others. Finally, small cells are typically deployed in third-party premises with limited access and control for the operators of the network. This will make management a nightmare with existing distributed MIMO solutions, as nodes cannot be easily replaced and changes to the wireless environment are often out of operator control, making it very hard to ensure that all slaves always hear the leader.

Chorus addresses this issue by distributing the synchronization role across all the nodes. In Chorus, there are no special roles, *i.e.*, no leader and slaves. All Chorus nodes transmit a synchronization signal, and all synchronize their oscillator phases by listening to the synchronization signals transmitted by other nodes in their vicinity. There are also no special constraints on topology or connectivity. Thus, the coordination protocol is easy to manage and naturally scales to large networks with transmitters that are no longer in hearing range of each other. It also makes the system resilient to node failure, addition and removal.

Distributing the synchronization role solves the scalability issue and makes the system more resilient and manageable. However, naively applying this design leads to synchronization loops –e.g., a node may be synchronizing with a second node, that is synchronizing with a third node, which is synchronizing with the first node in the loop. Such loops are destabilizing, *i.e.*, they prevent the system from converging [42]. To prevent loops, Chorus has a distributed protocol to organize the nodes in the

form of a tree (specifically a fat tree). Nodes at the same depth of the tree transmit the synchronization signal in the same frequency, and this composite synchronization signal is used by nodes at the next lower depth to synchronize themselves. In addition to this resilient fat tree architecture, Chorus also has a special acyclic structure at the root to make the synchronization tree resilient to failures of the root. We describe the details of our protocol and resilient architecture in Section 5.3.

5.1.2 Robust Phase Tracking

In the proposed architecture, the synchronization signal is no longer a clean transmission from one leader - it is a composition of synchronization signals from multiple nodes. Those signals from multiple nodes could destructively interfere at the receiver resulting in a low signal strength for synchronization. Furthermore, having nodes joining and leaving the network at different points in time would increase the variability on the synchronization signal, and disturb the phase tracking. Moreover, compared to past systems, where the phase difference is computed at the time of transmission and applied immediately. In contrast, in Chorus the synchronization signal can be sent only sporadically due to LTE frame structure. As a result, the measured phase difference can be outdated.

To deal with its more stringent synchronization requirements, Chorus combines techniques from signal processing and control theory to improve the resilience of the system while minimizing the phase error. Specifically, Chorus allows different nodes to send different synchronization signals, then at the receiver instead of estimating the channel, it computes the change in phase that was accumulated over the period between the two measurements. This phase change is then fed to a controller which minimizes the error by tracking the drift of the local oscillator.

5.1.3 LTE Compatibility

Similar to MegaMIMO 2.0 for WiFi, we would like Chorus to be directly applicable to LTE small cells without having to change the LTE protocol or user devices. To

do so, we leverage that LTE's OFDM modulation divides the frequency band into subcarriers, which themselves get divided into timeslots called resource elements. Chorus allocates some of these resource elements for transmitting synchronization signals. Chorus also schedules the resource elements used for synchronization so that they look to user devices as if they were yet another user in the system. Only small cells participating in Chorus need to interpret the synchronization signal.

We implement Chorus in a hardware platform composed of an FPGA connected to a high speed ARM core. We use the srsLTE open source LTE stack implementation [61] and augment the eNodeB with Chorus. Our implementation therefore provides an LTE small cell that is capable of synchronized operation and distributed joint transmission. We perform our experiments in the white space frequency bands (680 MHz), which is very close to the 700-800 MHz where major US operators such as Verizon and AT&T run their networks. Our results show:

- Chorus's distributed synchronization is scalable, and synchronizes small cells that are not within range of each other. Specifically, the median phase variance between two such small cells is less than 0.004 $radians^2$.
- Chorus is resilient to the loss of any single node by enabling multiple nodes to simultaneously transmit the synchronization signal. Specifically, Chorus achieves synchronization within a phase variance of 0.002 even when 10 independent small cells jointly transmit the synchronization signal. This means that the interference between concurrent transmissions from our distributed MIMO nodes is less than 0.5 dB.
- Chorus's controller is resilient to variations in synchronization signal SNR. Specifically, it delivers accurate synchronization even when synchronization SNR is as low as 6 dB. The resilience to loss of any single node, as well as variation in synchronization SNR, ensure that Chorus's deployments are easy to manage.
- We evaluate Chorus in a 20 node testbed deployment, and compare its performance with a clustering based approach. Chorus delivers 2.7× higher throughput than a clustering based approach in our testbed. This gain is the result of Chorus's ability to build a large distributed MIMO system across nodes that cannot hear a single

leader, and allow all these nodes to transmit together. In contrast, a clustering based approach must divide the network into multiple distributed MIMO clusters, each within range of a single leader. These clusters have to transmit in different time slots to avoid mutual interference, and as a result misses out on large throughput gains.

- We evaluate the scaling performance of Chorus in a larger geographic setting using simulation. Specifically, we consider an example deployment of 25600 small cells in an 8×8 sq. km. area, which is slightly larger than the size of Manhattan. We show that Chorus can synchronize cells within a median radius of 5 km to within a phase variance of 0.004. Nodes outside this range show higher phase variance. These differences however are irrelevant, because these nodes are much more distant from each other than the range at which nodes interfere in the network.

5.2 Gains of a Distributed Design

Prior distributed MIMO systems deliver their throughput gain by enabling APs (or base stations) to synchronize their phase to one leader that is heard by all APs. A natural question is what throughput gains accrue from synchronizing the phase of APs that do not hear a single leader. In this section, we use a couple of simple examples to answer this question, and illustrate the benefits of a leaderless distributed MIMO.

Consider the simple topology shown in Fig. 5-1 with 6 APs. Each AP hears its adjacent APs, and each client hears the two APs closest to it.

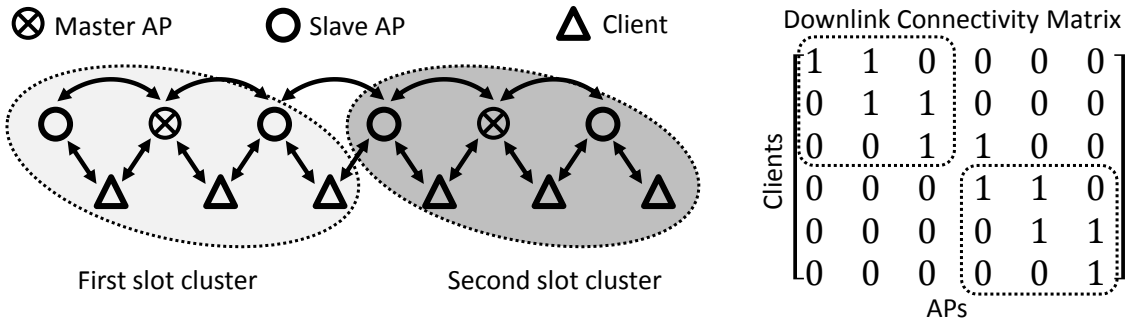


Figure 5-1: **1D clusters sparse**

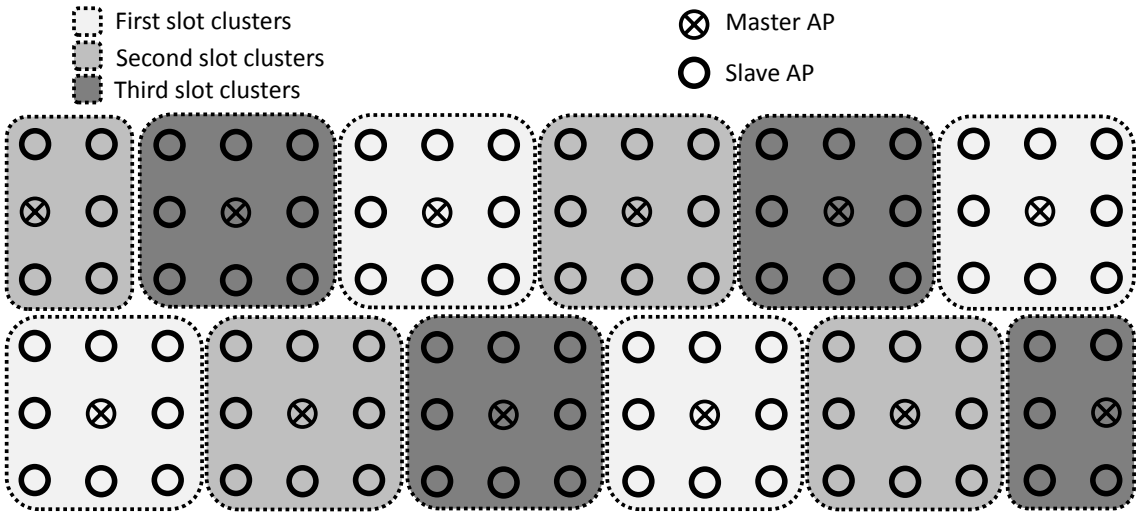


Figure 5-2: **2D clusters**

A leader-based distributed MIMO architecture for this network would therefore group every three adjacent APs into a cluster. Each cluster will have one leader and two slaves. It would transmit concurrently to the three clients closest to the APs in that cluster. The shaded areas in Fig. 5-1 show the potential leader-based clusters. Note that adjacent clusters cannot transmit simultaneously in this network, since the transmissions of the edge APs of the two clusters would interfere at the clients that fall near the edge of the two clusters. As a result, this network supports 3 simultaneous transmissions.

In contrast, consider a leaderless design that synchronizes the phase of all APs, even when all the APs do not hear each other. Such a design can support a distributed MIMO cluster that allows all APs to transmit simultaneously to all clients. It would therefore support 6 simultaneous transmissions, delivering a throughput gain of 2 over traditional leader-based distributed MIMO.

The gains would be higher if we extend this to two dimensions. In particular, consider the example in Fig. 5-2, where the APs are laid out in a two dimensional grid, and each AP can hear the adjacent APs along the grid. The dotted lines depict the leader-based distributed MIMO clusters. In order to avoid interference between adjacent clusters, only one in every three clusters can transmit at one time, with each shading depicting the clusters transmitting at a particular time slot. In

contrast, leaderless distributed MIMO would allow APs in all clusters to transmit simultaneously to their associated clients, delivering a throughput gain of 3x.

It is important to note a few points:

- The benefit described above is not particular to the specific interference pattern described between individual APs, or APs and clients. In particular, as the deployment scales, it is natural to have APs that do not hear each other, or all clients. In such a situation, leader-based distributed MIMO is inherently limited in the sizes of clusters it can support, and therefore the throughput gains it can deliver, compared to leaderless distributed MIMO.
- One can get significant gains even with small cluster sizes, as demonstrated in the figure that compares a cluster with 3 APs, and a cluster with 6 APs.
- Additionally, the presence of a synchronization fabric that ensures that APs are synchronized does not necessitate that all the APs in the entire network need to transmit together. Instead, it provides the ISP flexibility in picking which APs can transmit with each other as part of distributed MIMO. In addition to the throughput gain provided by this flexible picking, it can also allow the ISP to adapt distributed MIMO to dynamic congestion in the network. For instance, the operator might choose to use distributed MIMO on demand simply to deliver throughput gains in congested areas, and change which APs are part of these on-demand distributed MIMO clusters. Leader based distributed MIMO systems do not provide such flexibility.

5.3 Self Organizing Tree Architecture

Chorus achieves leaderless synchronization by making all nodes equally responsible for propagating the synchronization signal. Specifically, every node continuously listens to the synchronization signal, compensates for its own phase shift with respect to the synchronization signal, and transmits a synchronization signal that can then be used by other nodes to measure their phase shifts and synchronize their oscillators.

However, if nodes do this naively, they could end up with synchronization loops.

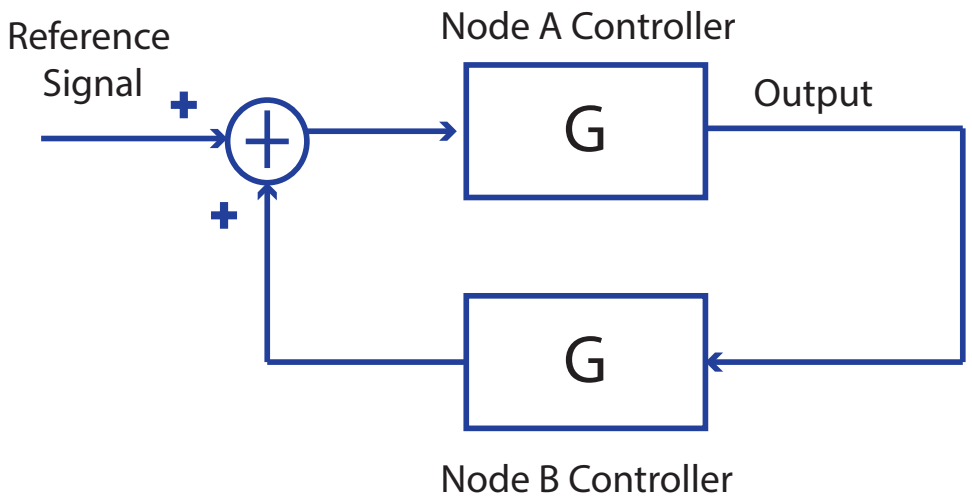


Figure 5-3: **Positive Feedback between two nodes.** The figure shows two nodes transmitting and receiving synchronization signals to each other. Each node can be modeled as a control system with transfer function G. Node A receives a reference synchronization signal from node B, and transmits a synchronization signal, which in turn is used by Node B as its input synchronization signal, producing a positive feedback loop.

For example, consider the following topology – node 1 transmits a synchronization signal that is used by node 2, which transmits a synchronization signal used by node 3, which in turn, transmits a synchronization signal that is used by node 1. Such synchronization loops can destabilize the system, *i.e.*, prevent the network from converging to a coherent phase.

To understand why, let us model the system using control theoretic concepts. Each node in our system has a reference signal which is the synchronization signal received by the node. The node internally has some controller which aims to match the node's phase to the reference phase. Since the details of the controller are irrelevant to this argument, let us abstract the controller inside the node by the function G. Since the goal of the controller is to ensure that the output synchronization signal matches the reference, the transfer function G should be as close to 1 as possible.

Now, let us see what happens when there is a loop. We will take a simple case with two nodes A and B. Node A receives a synchronization signal from node B, which it uses to synchronize and transmit its own synchronization signal. Node B will hear the synchronization signal transmitted by node A, use it to synchronize itself, and in

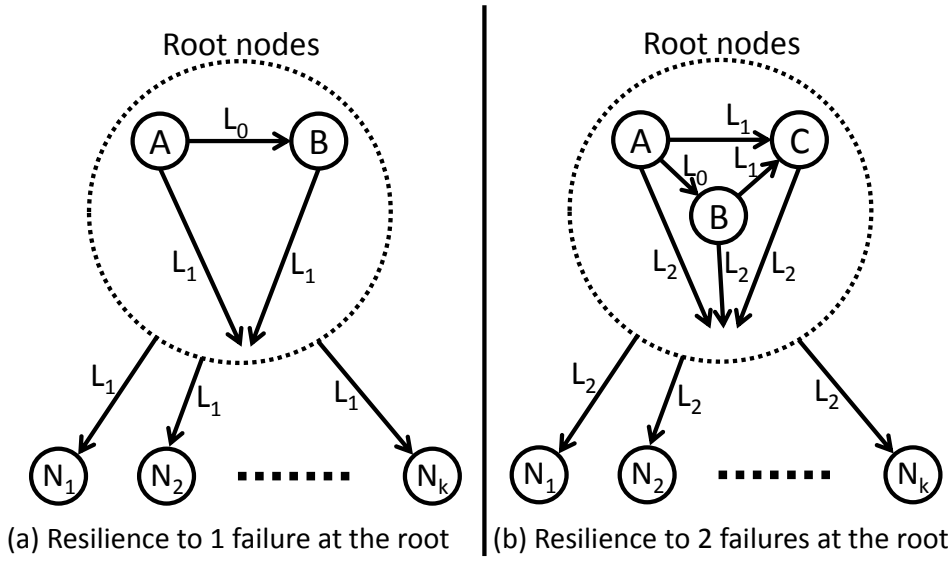


Figure 5-4: **Resilience to Root Failures:** (a) shows a topology for the synchronization tree that is resilient to either node A or B failing. (b) shows a topology that is resilient to any two nodes of {A,B,C} failing.

turn transmit the synchronization signal that node A uses. This leads to a feedback loop, as shown in Fig. 5-3. From basic control theory [42], the transfer function of such a loop is $\frac{G}{1-G^2}$. Since, accurate tracking requires G to be as close to 1 as possible, the loop is bound to be unstable. The same argument generalizes to larger loops.

5.3.1 Chorus's Layering Protocol

Since our objective is to eliminate loops, by definition, our synchronization topology must be a tree. We will refer to the nodes at a particular depth in the synchronization tree as a layer, with the root being at depth 0, and so on. Each layer has its own frequencies in which it transmits its synchronization signal. Each node transmits its synchronization signal in the frequencies corresponding to its layer, and synchronizes itself by listening to the synchronization signal on the frequencies corresponding to its parent layer directly above it. Thus, the tree structure is really a fat tree.

Chorus's synchronization tree is self-organizing. All that the administrator has to do is to pick the root of the tree. Once the administrator nominates the root, the tree self organizes as follows: The root starts transmitting a signal in layer 0. Each node who wants to join the system listens for synchronization signals on the frequencies

associated with all layers. The node picks the lowest layer from which it receives an adequately strong synchronization signal. Specifically, each node internally is running a controller (described in 5.4) whose objective is to match its phase with the phase of the synchronization signal. The controller needs a minimum synchronization signal SNR in order to provide robust phase coherence. Thus, when picking a layer, the node picks the lowest layer whose SNR is above this threshold. We evaluate the performance of Chorus's controller in 5.9.2, and describe how we can determine this SNR threshold. Intuitively, one can see these layers in space as concentric rings starting from the single layer 0 node in the network.

Since the synchronization signals will attenuate with distance, Chorus can reuse synchronization frequencies. For example, say that we have 8 sets of synchronization frequencies, which are used by layers 0 through 7. Layer 8 can re-use the frequencies corresponding to layer 0, assuming nodes at this layer are sufficiently distant from the node in layer 0. By default, Chorus uses 8 distinct sets of synchronization frequencies. Thus, layer i, for $i \geq 1$, receives on synchronization frequency $(i-1) \mod 8$, and transmits on synchronization frequency $i \mod 8$. In 5.5, we describe how these synchronization frequencies can be provided cheaply in the LTE framework without allocating dedicated frequency bands.

5.3.2 Resilience

Chorus is resilient to node addition and removal, as well as changes in channel quality and, therefore, topology of the synchronization tree. In particular, say that a node fails. Such a failure typically has no impact on the other nodes. Specifically, all nodes who are synchronizing to the layer on which the failed node transmits are likely to continue receiving the synchronization signal from the rest of the nodes in that layer. In the unlikely case, where the failed node has a descendant who cannot hear any other nodes from that layer, the descendant will immediately discover the loss of its synchronization signal from its parent layer, and therefore pick the next layer with sufficiently high SNR. This process could, in principle, cascade across several nodes, and naturally resolve itself with each node moving to the appropriate layer. Node

addition works similarly.

In our description so far, we have addressed the resilience at all layers except layer 0. We now address how to make layer 0 resilient. Specifically, consider the topology in Fig. 5-4(a). This is a modified tree topology where node A transmits a synchronization signal on layers 0 and 1, node B listens to the synchronization signal on layer 0, and transmits a synchronization signal on layer 1, and nodes $N_1, N_2, \ldots, N_k$ listen to the synchronization signal on layer 1, and transmit a synchronization signal on layer 2. In this case, if node A dies, node B automatically becomes the root of the network and the system continues to operate as usual. Further, if node B dies, nodes $N_1, N_2, \ldots, N_k$ transparently continue to synchronize with parent layer 1, but using node A alone. Thus, this system can withstand one root failure, at the expense of an additional layer at the root alone. We can extend this idea to multiple root failures. For instance, Fig. 5-4(b) shows a root topology that can withstand up to two nodes at the root failing, *i.e.*, any two nodes of A, B, and C can fail. While we expect that the system administrator should pick robust nodes for the root, such as base stations, the fault tolerant topology shown here enables resilience to the transient failure of one or more of the root nodes.

5.4 Robust Phase Update Algorithm

Chorus embeds the synchronization signal in the LTE frame. LTE divides the frequency band into subcarriers, which themselves are divided into timeslots called resource elements. To maintain low overhead, the synchronization signal appears in certain resource elements, once every 5 milliseconds (the details are in 5.5). Every time the synchronization signal is available, the small cell obtains new measurements, which it uses to correct any phase drift caused by it's own oscillator.

Chorus's phase update algorithm has to work under stringent requirements. Specifically, it needs to provide continuous phase tracking to accommodate LTE's continuous data transmission, while using infrequent phase measurements in order to integrate into the LTE framework with low overhead. Furthermore, Chorus's syn-

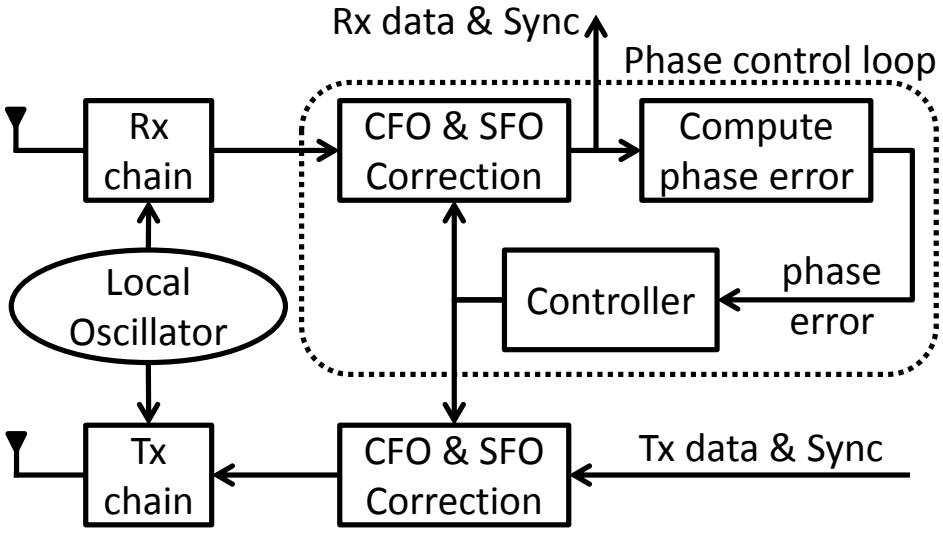


Figure 5-5: **A Block-diagram of Chorus's Phase Control Loop.**

chronization signal is transmitted by multiple nodes as opposed to a single leader, which makes it susceptible to destructive interference resulting in low signal strength. This in addition to being stable while nodes are joining or leaving the network. In order to achieve these requirements, Chorus combines techniques from signal processing and control theory. The phase update algorithm has two main components.

(a) Control Block: In order to cope with the infrequent phase measurements, Chorus formulates phase tracking as a control problem. Fig. 5-5 shows the block diagram for the phase estimation and correction. The Controller takes as input the measured phase error and it produces a correction value that is fed to the correction modules on both the Tx and Rx signal paths. The task of the controller is to minimize the magnitude of the phase error, by tracking the phase drift of the local oscillator. To achieve the optimal control behaviour we use MATLAB control toolbox to design and tweak Chorus's controller. This is done by accurately modelling the phase control loop, and accounting for the delays of the different components in the loop. Chorus's controller is implemented in HW with loadable coefficients from SW. This is to reduce the loop latency and achieve better stability, while providing enough flexibility to adapt for different SNRs of the synchronization signal.

(b) Robust Synchronization Signal: In order to provide resilience against destructive interference and changes in the network structure, Chorus designs the syn-

chronization signal sent from multiple nodes to be uncorrelated. Specifically, each node gets to choose a random signal to be transmitted on each resource element used for synchronization, then it keeps sending the same signal repeatedly every 5 milliseconds. At the receiver, instead of estimating the channel, the receiver computes the change in phase that was accumulated over the 5 milliseconds on each of the resource elements used for synchronization, and uses that phase change to correct for it's own oscillator drift. Now, since the different nodes are sending different signals, it becomes unlikely to have destructive interference on all the resource elements.

Now to show how this would increase resilience to changes in the network structure, consider the following example. Say that N nodes are transmitting the synchronization signal at the same layer L. And the synchronization signal is using K resource elements. The synchronization signal transmitted by node number n on resource element number k is denoted by S_k^n. When a node is being synchronized to the combination of these signals, and it's oscillator had a phase drift of $\Delta\phi$ over the 5 milliseconds, then the phase drift at the receiver is computed as $angle(\psi)$, where ψ is given by Equation (1).

$$\psi = \sum_{k=1}^{K} conj(\sum_{n=1}^{N} S_k^n) \times (\sum_{n=1}^{N} S_k^n e^{j\Delta\phi})$$

$$\psi = e^{j\Delta\phi} \times \sum_{k=1}^{K} \left| \sum_{n=1}^{N} S_k^n \right|^2$$

If a new node joins the network during the 5 milliseconds between the two measurements and starts transmitting S_k^{N+1}. In this case the ψ is given by Equation (2).

$$\psi = \sum_{k=1}^{K} conj(\sum_{n=1}^{N} S_k^n) \times (\sum_{n=1}^{N+1} S_k^n e^{j\Delta\phi})$$

$$\psi = e^{j\Delta\phi} \times (\sum_{k=1}^{K} \left| \sum_{n=1}^{N} S_k^n \right|^2 + \sum_{k=1}^{K} (S_k^{N+1} \times conj(\sum_{n=1}^{N} S_k^n)))$$

Comparing this with the previous case, there is a new term that is added here.

However, since S_k^n is selected at random with zero mean, and K is typically large 80 resource elements in our implementation. This will result in a significant rejection for the disturbance introduced by the new node.

5.5 LTE Compatibility

In this section, we describe how Chorus can be implemented within the LTE protocol structure. Specifically, there are four issues that need to be addressed in order to integrate Chorus with LTE. We discuss these issues below.

5.5.1 Making Synchronization Signals Transparent to End-User Devices

In earlier sections, we described how Chorus assigns different synchronization frequencies to different layers. Of course, we would like to transmit these synchronization signals in-band. Furthermore, we would like these synchronization signals to be transparent to existing end-user devices (called UEs in LTE).

Our basic idea is to make the synchronization signal look like yet another user in the system. Only the small cells participating in the Distributed-MIMO system understand how to interpret the synchronization signal, and process it appropriately, whereas regular user devices simply steer clear of these frequency bands because they are not intended for them.

To achieve this, Chorus leverages the structure of LTE transmissions [18, 21, 22]. LTE transmissions are organized as frames, with each frame being 10 ms long. Each frame can be viewed as a time-frequency map. In particular, each frame consists of 10 1 ms subframes. It is also divided in the frequency domain into many subcarriers which are combined using OFDM.

Users are allocated at the granularity of groups of subcarriers for a subframe (Resource Block). Thus, we allocate the virtual synchronization user two resource blocks each in subframe 0, and subframe 5. A typical LTE channel of 10 MHz has

500 resource blocks in a frame. This amounts to a total overhead of 4/500, which is less than 1%. The overhead further decreases with increasing channel width. Having allocated these resource blocks for synchronization, Chorus maps the synchronization signals for different layers to these resource blocks. Specifically, we interleave the synchronization signals for the different layers in these resource blocks, similar to the way user data is interleaved.

5.5.2 Addressing FDD Systems

LTE systems come in two flavors: TDD (Time Division Duplex) which has the same frequency bands for uplink and downlink and schedules uplink and downlink at different points in time, and FDD (Frequency Division Duplex) which uses different frequency bands for uplink and downlink, with both uplink and downlink operating simultaneously at all times.

As we described above, to make the system transparent to end-user devices, we need the synchronization signal to be transmitted as if it were a virtual user. However, this means that such signals would have to be transmitted on the downlink frequencies for FDD systems. Small cells in FDD systems are not capable of listening to downlink transmissions, and hence cannot use these signals to synchronize. Conceptually, one solution to this problem is to modify end-user devices to listen to the synchronization signal on the downlink, synchronize to it, and transmit a synchronization signal on the uplink. However, this defeats the objective of not modifying end-user devices. So, the solution is to simply have the operator deploy a few special end-user devices that have been modified to support this cascading of synchronization signals. We refer to these boxes as Cascading End Users.[1]

[1]This does not mean that the operator actually deploys cellphones. The operator deploys boxes that are similar in form factor to a picocell, but are significantly simpler – they do not need to do end-user data processing, and hence are not limited by backhaul requirements or other deployment constraints.

5.5.3 Joining The Network

Initially, as a node decides to join the system, its phase might not at all be coherent with other synchronized nodes in the network. The node cannot simply start transmitting a synchronization signal before it reaches a reasonable level of coherence. Thus, when the node starts, it uses existing LTE synchronization signals, specifically the Primary and Secondary Synchronization Signals (PSS and SSS) that are transmitted by other distributed-MIMO nodes, to obtain a coarse frequency offset estimate and time synchronization to a frame boundary. Having obtained these estimates, the Chorus node then determines the synchronization layer to which it belongs, and starts its controller. The controller then uses the synchronization signals to start doing fine tracking of the frequency offset, and reports to the node when it has converged. At this point, the small cell is fully synchronized and ready to join the distributed-MIMO system by transmitting the synchronization signal on its appropriate layer. The node also now joins the rest of the distributed-MIMO nodes in transmitting PSS and SSS signals. For FDD, the Cascading End Users will relay the PSS and SSS on the uplink for use by the joining small cells.

5.5.4 Transmission

Now that small cells are part of the Distributed-MIMO network, they can participate in joint transmission much like with Coordinated Multi Point (CoMP) today. Specifically, the system can use CSI-RS (Channel State Information Reference Signals) to measure downlink channels and get feedback from UEs, the UE-RS (UE-Specific Reference Signals) as pilots to measure the beamformed downlink channel to end users, and the actual data subcarriers for beamformed data transmission.

5.6 Implementation

We prototype Chorus using a joint software-hardware implementation. Our prototype is integrated with srsLTE [61], an open source LTE stack library, and hence it is com-

patible with LTE end user devices. The prototype runs on a custom programmable radio platform, which comprises of a Zedboard integrated with the Analog Devices RF frontend, FMCOMMS3. The Zedboard has an FPGA connected to an ARM core by a high-throughput, low latency bus, hence allowing for real-time processing at the PHY layer. Our hardware implementation is done using Verilog on the FPGA, and our software runs on the ARM core.

We implement the components of Chorus in the different software and hardware elements as follows:

- The layered architecture is implemented across both software and hardware. Specifically, the hardware reports the signal strength for the different synchronization layers to our software, which determines which layer this small cell node should belong to. The hardware then determines the appropriate time-frequency pattern of the transmitted synchronization signal based on this layer. Further, the initialization protocol described in 5.5 to determine time and coarse frequency synchronization are implemented in hardware.
- Chorus's controller that performs continuous synchronization and ongoing tracking of the received synchronization signal is implemented in hardware so as to be real-time and responsive.
- The LTE frames for transmission are created jointly by software and hardware. Specifically, the srsLTE software encodes and modulates the data, and produces protocol compliant LTE frames, with both cell specific and user specific information. The hardware translates this frame into the signal representation after augmenting it with synchronization information depending on the layering information described above.
- For joint transmissions, we provide software knobs that allow the system to pick between diversity, nulling and multiplexing. The hardware executes on these knobs by performing the correct precoding.

5.7 Testbed

Our evaluation testbed is distributed across one floor in our building (80 × 60 feet). Depending on their location, nodes can be separated by elevator banks, passages and inside walls *etc.* The testbed contains a total of 20 nodes, with each node emulating either an LTE small cell or an LTE client. We assign one LTE small cell node to be in layer 0, and then run our layering algorithm to assign layers to other small cells. We control the transmitted power of our nodes to create different interference neighborhoods, and emulate a larger geographic area. For each experiment, we deploy our nodes around the floor to create different topologies and layering of small cells. The exact topology for each experiment is described along with the experiment (See Fig. 5-10 for an example layout for one of the experiments.) In order to replicate LTE conditions as closely as possible, we run our experiments in the white space frequency bands. Specifically, we use the 680 MHz center frequency, which is very close to the 700-800 MHz where major US operators such as Verizon and AT&T run their networks. LTE has multiple possible channel widths, and our prototype is implemented using the 3 MHz channel width. Chorus can support any channel width; however, our choice of 3 MHz is dictated by the FPGA size and processing power of our platform.

5.8 Metrics

In this section, we describe the metrics we use to evaluate Chorus.

(a) Phase Variance: The key function of distributed MIMO protocols is to synchronize the oscillator phase across different nodes. Thus, it is natural to evaluate Chorus in terms of the phase variance across the distributed MIMO transmitters. In an ideal scenario, the phase across multiple independent transmitters will be coherent across time, and the variance will be zero. The smaller the phase variance, the lower the interference caused by misalignment of signals during joint transmission.

(b) Throughput Gain: We compute the ratio of the total throughput that can be

delivered by the network, *i.e.* by concurrent transmissions from multiple independent small cells to multiple end user devices using a distributed-MIMO scheme, to the throughput delivered by the network without distributed-MIMO, *i,e. i.e.* with only one small cell transmitting to a single end user device at a time.

5.9 Results

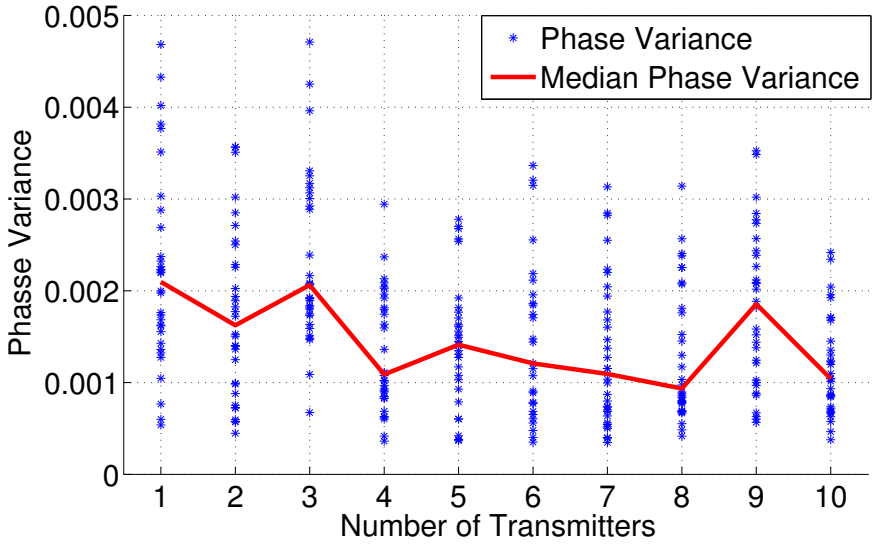
In this section, we evaluate Chorus's performance and scaling behavior.

5.9.1 Resilience to Multiple Nodes Transmitting the Synchronization Signal

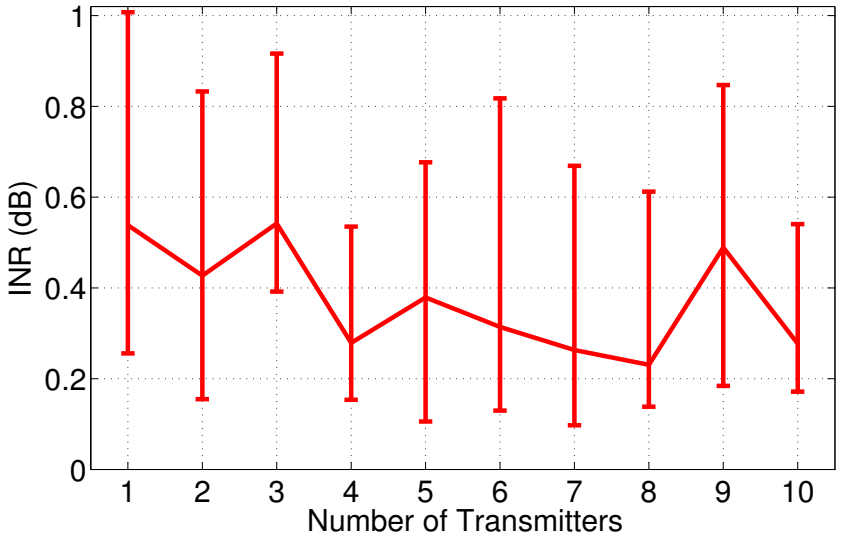
A key property of Chorus is that it achieves resilience to the failure of individual nodes and changes in network connectivity by having multiple nodes simultaneously transmit the synchronization signal. In this section, we verify how the quality of synchronization varies as the number of nodes transmitting the synchronization signal increases.

For this experiment, we deploy the nodes in our testbed such that there are three distinct areas of connectivity. We pick one node and assign it to layer 0. We then run the layering protocol, which then assigns layer 1 to all the nodes that can hear layer 0, and layer 2 to all the nodes that can hear layer 1. By definition, layer 1 nodes synchronize to the layer 0 node, and layer 2 nodes synchronize to the combined signal from layer 1 nodes. We vary the number of LTE small cells in layer 1. We then pick one node in layer 2 to synchronize to the combined layer 1 signal. Our goal is to evaluate how well the node in layer 2 is synchronized to the individual nodes in layer 1, though it is using a combined synchronization signal from all the layer 1 nodes. For this, we pick one node in layer 1 to transmit data concurrently with the node in layer 2, and receive these signals at an auxiliary node.[2] To enable the auxiliary node to measure the phase difference between the layer 1 and layer 2

[2]Note that all nodes in layer 1 are transmitting synchronization signals.



(a) Phase variance



(b) Interference-to-Noise Ratio (INR)

Figure 5-6: **Performance at an auxiliary node as a function of number of joint transmitters in an intermediate synchronization layer.** (a) plots the phase variance observed at an auxiliary node as a function of the number of simultaneous transmitters in a layer. The figure shows that the phase variance observed at an auxiliary node remains constant and low even as the number of transmitters in the intermediate synchronization layer increases. (b) shows the impact of this phase variance on interference, by estimating the Interference to Noise Ratio (INR). The graph shows that the INR stays less than 0.5 dB for up to 10 simultaneous transmitters. This demonstrates that Chorus's strategy of having all nodes in the system transmit the synchronization signal provides robust synchronization.

nodes, we make them transmit in alternate symbols, *i.e.*, the layer 1 node transmits in the odd numbered symbols, and the layer 2 node transmits in the even numbered symbols. The auxiliary node then compares the phase difference between the two nodes using every pair of adjacent symbols, after compensating for the corresponding channels between the layer 1 and layer 2 node to itself. We repeat the measurement with different choices of layer 1 and layer 2 nodes.

Results. Fig. 5-6(a) plots the variance in the phase difference between the data signals from layer 1 and layer 2 nodes as a function of the number of transmitters on layer 1. The variance is measured across 1 sec intervals which is significantly larger than the channel coherence time. If the layers are synchronized perfectly, the phase difference between the adjacent symbols will be zero, since the oscillators are locked with each other. Any error in synchronization will manifest as a non-zero phase difference. The figure shows that the variance in the phase difference stays below 0.0022 radians even as the number of transmitters in layer 1 increases from 1 to 10. The low phase variance shows that Chorus operates properly without a unique leader, and can maintain good synchronization even when many small cells contribute to the synchronization signal.

To understand the implication of phase variance, let us reason about its impact on interference. Say that the distributed MIMO is trying to eliminate interference by nulling the transmitted signals at a particular receiver. If the transmitters are perfectly phase coherent, then they can compensate for the channels between them to the receiver perfectly, and align the signals to eliminate interference. Any phase variance will translate into phase noise and manifest itself as a misalignment of the signals, *i.e.*, interference. We can use our measurements from above to compute the residual interference when the two transmitters apply nulling at the auxiliary node. Fig. 5-6(b) plots these results. Note the similarity in shape between the INR plot and the phase variance plot. This is because of the approximately linear relationship between them at low INR. The figure shows that the residual interference to noise ratio is 0.5 dB, which is small and comparable to previous systems. This shows that despite the fact that Chorus has no leaders and multiple nodes transmitting the

synchronization signal concurrently, it can achieve high performance comparable to past systems that relied on a single leader.

5.9.2 Resilience to Varying Synchronization Link Quality

It is important to understand Chorus's performance as a function of the synchronization link SNR for two reasons. First, recall that Chorus nodes join the lowest numbered layer with adequately strong synchronization signal. Hence, we want to calibrate the performance of the controller across SNRs to determine this value. Second, a key objective of Chorus is to be able to work robustly across a range of synchronization SNRs to ensure resilience to variations in link quality.

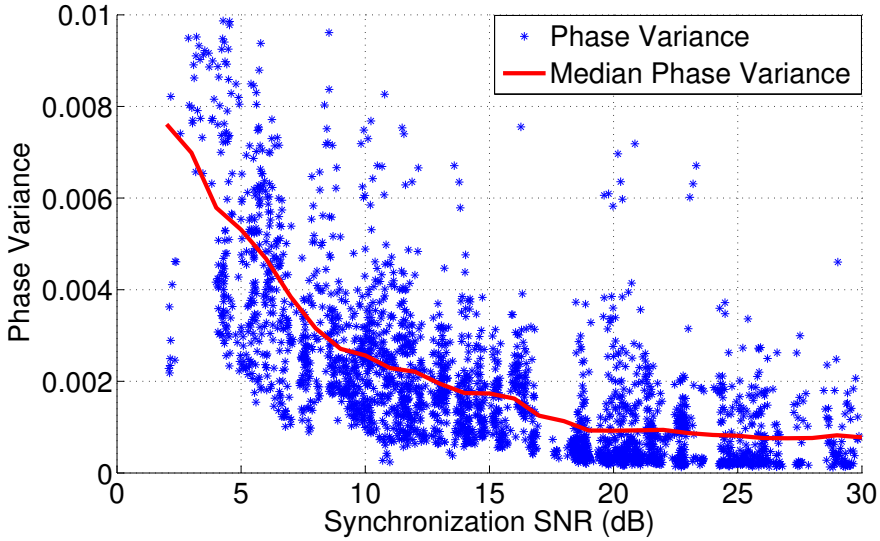
For this experiment, we again deploy the nodes as before such that the layering algorithm divides the network into three layers, layer 0, layer 1, and layer 2. We pick two small cell nodes, both in layer 2, synchronizing to a node in layer 1. Our objective is to investigate whether the two small cell nodes have accurately synchronized their oscillators. To do that, we repeat the same measurement procedure as in 5.9.1, where the two small cells interleave their data transmissions, and an auxiliary mode measures the phase difference between the two small cells.

We repeat this process and characterize the phase synchronization accuracy across different SNRs and locations in our testbed.

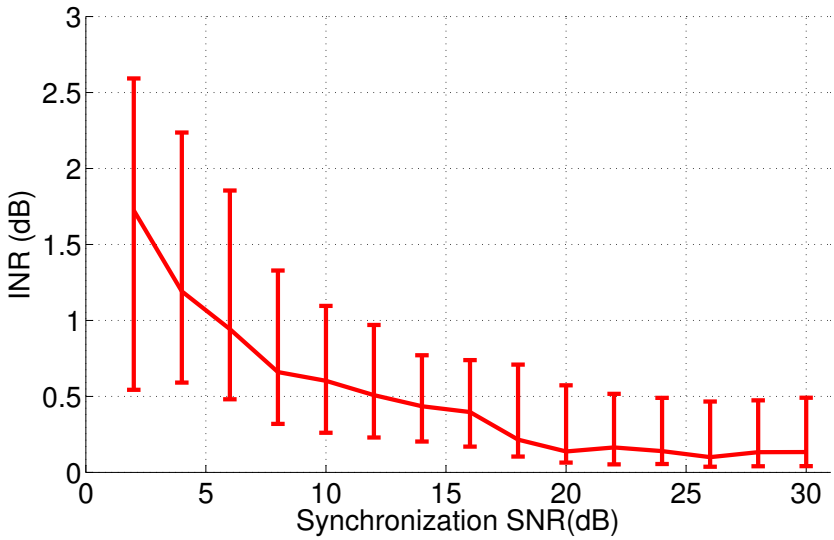
Results. Fig. 5-7(a) plots the variance in phase difference between the two small cells measured across a 1 sec time interval, as a function of the minimum SNR of the synchronization links, *i.e.*, the links from the layer 1 node to the two small cells. The figure shows that even when the synchronization link SNR is as low as 6 dB, the median phase variance stays lower than 0.0045.

As before, we again examine the impact of this phase variance on interference, by measuring the residual interference, *i.e.*, INR. We plot the INR as a function of the minimum SNR of the synchronization links in Fig. 5-7(b). Again, we notice that the median INR is very small, less than 1 dB for synchronization SNR larger than 6 dB.

Figs. 5-7(a) and (b) show that by picking the minimum synchronization link SNR



(a) Phase variance



(b) Interference-to-Noise Ratio (INR)

Figure 5-7: **Performance at an auxiliary node as a function of synchronization SNR.** (a) plots the variance in phase between the signals received from two small cells at an auxiliary node and (b) shows the median Interference to Noise Ratio (INR) for various synchronization SNRs. The figure shows that Chorus can achieve median INR less than 1 dB even for synchronization SNRs as low as 6 dB. This determines the minimum SNR of the synchronization link used by Chorus's layering algorithm.

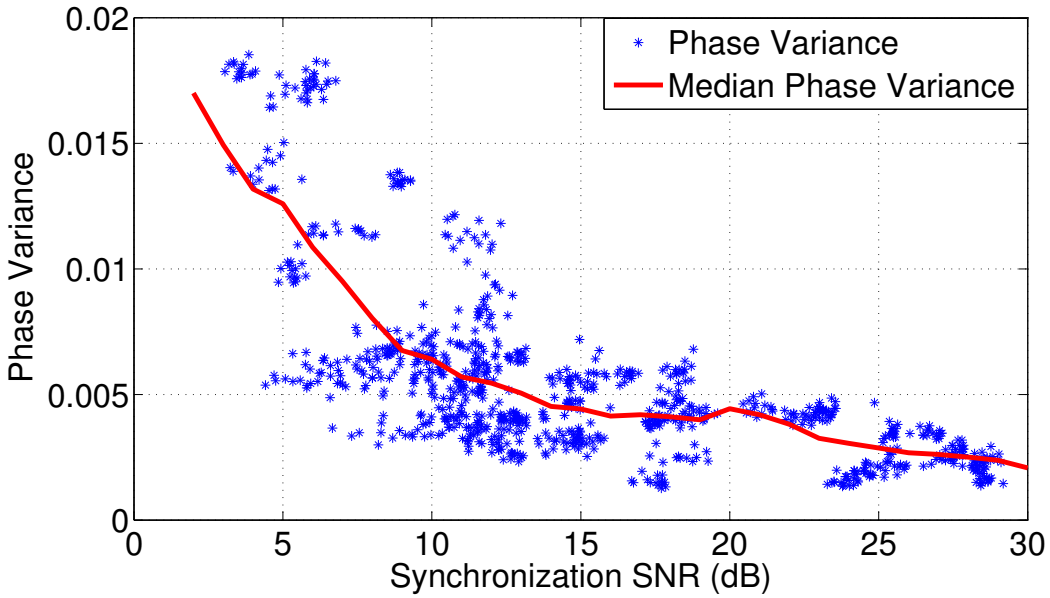


Figure 5-8: **Phase variance between small cells that cannot hear each other and are synchronized through an intermediate node, as a function of synchronization SNR.** The figure shows that Chorus can achieve high level of synchronization even when the nodes cannot hear each other.

to be 6 dB, Chorus can ensure that the maximum interference is less than 1 dB.

5.9.3 Synchronizing Nodes that Cannot Hear Each Other

Chorus's architecture allows it to synchronize nodes that are not within listening range of each other by cascading the synchronization signal across layers. In this section, we evaluate the performance of such cascading.

For this experiment, we deploy our nodes such that the layering algorithm divides the network into four layers, layers 0 through 3. We then pick three nodes, one each in layers 1, 2 and 3, from our testbed. By the definition of our layering algorithm, nodes in layer 3 cannot hear nodes in layer 1. For each run, we pick one node in layer 1, and one node in layer 3, and synchronize via a node in layer 2, as described in 5.3. Our objective is to investigate whether the nodes in layer 1 and layer 3 have accurately synchronized their oscillators. We therefore repeat the same measurement as in 5.9.2, but with the small cells in two different layers.

Results. Fig. 5-8 plots the variance in phase difference between the two small cells

measured across a 1 sec time interval, as a function of the minimum SNR of the synchronization links, *i.e.*, the links between the layer 2 node to the two small cells. The figure shows that the quality of synchronization stays high even as the SNR of the synchronization signal drops.

5.9.4 Comparison with Leader-Based Distributed MIMO

In this section, we empirically compare Chorus's fully distributed protocol with MegaMIMO, a leader-based distributed MIMO system. MegaMIMO is designed for Wi-Fi networks but can be extended to small cells. The data plane does not need to change since both Wi-Fi and LTE operate over OFDM. For the control plane, MegaMIMO makes the leader node transmit a synchronization header before each packet. This is not possible in LTE since one cannot insert new headers between LTE frames. To make MegaMIMO compatible with LTE, we make the leader transmit its synchronization information in the same resource elements used to transmit Chorus's synchronization signals. Since MegaMIMO has no layers, a MegaMIMO leader transmits its synchronization signal in the channels of all layers.

We consider a deployment across a single floor of our building. The floor consists of both offices and conference rooms, and our deployment consists of 20 nodes, with 10 acting as small cells, and 10 acting as clients. As explained earlier, we reduce the transmission power to emulate a larger geographic area. We measure the pairwise link quality between all the small cells to identify nodes that can hear each other well. Fig. 5-9 shows the link quality matrix with all links with SNR less than 6 dB (the minimum synchronization SNR, as determined in 5.9.2) gr-eyed out. As is clear from the link quality matrix, there is no single node that is heard by all other nodes in the network.

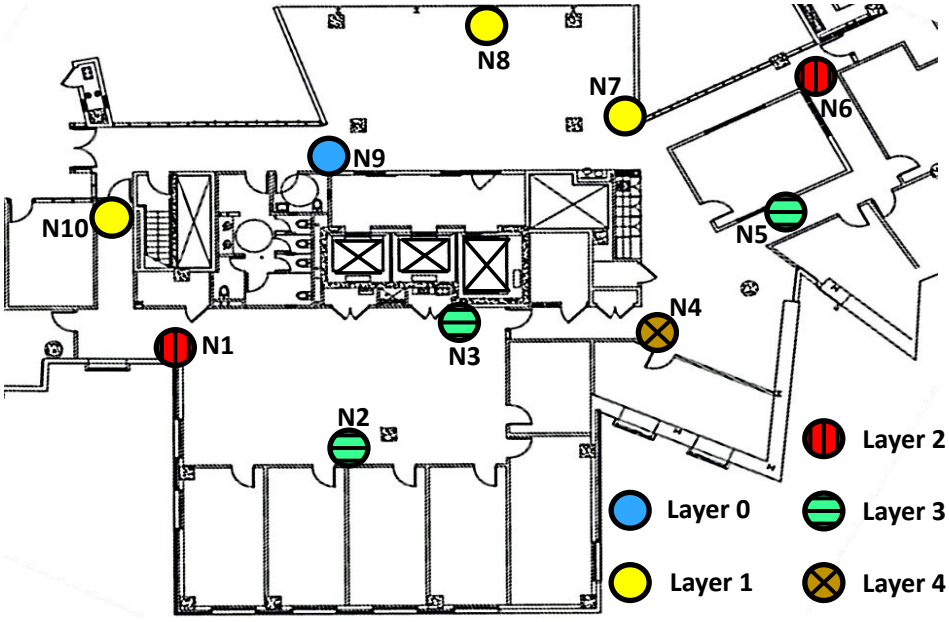
To run Chorus in this testbed, we need only to pick the root node; the synchronization layers self-configure. Fig. 5-10(a) shows the topology of our testbed with each node labeled with its layer as assigned by the layering algorithm. To run MegaMIMO, we need to decide how to divide the network into clusters of distributed MIMO systems where each cluster consists of one leader and its slaves. Based on

	1	2	3	4	5	6	7	8	9	10
1		24	15							14
2	24		21	10						
3	15	21		12						
4		10	12		17					
5				17		17				
6					17		12		7	
7						12		20	19	
8							20		11	
9						7	19	11		13
10	14								13	

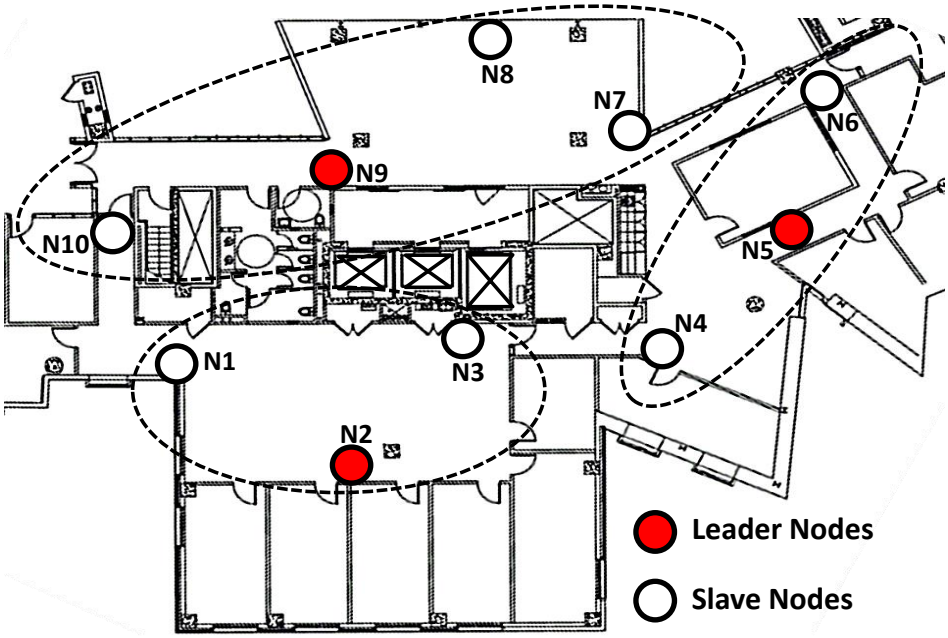
Figure 5-9: **Pairwise link strength between small cells.** The figure shows the pairwise SNR in dB between small cells in our deployment, with links with SNR less than the synchronization SNR threshold (6 dB) grayed out. No single small cell can be heard by all other small cells.

the link-quality matrix in Fig. 5-9, we can see that the minimum number of clusters is 3, i.e., MegaMIMO has to pick 3 leaders and their corresponding slaves. We pick the leader-slave clusters by solving an optimization problem that maximizes the sum of the link quality between each slave and its leader. This results in the following three clusters: The first cluster consists of N1, N2, and N3 with N2 as the leader, the second consists of N4, N5, and N6, with N5 as the leader, and the third is N7, N8, N9, and N10, with N9 as the leader. Fig. 5-10(b) shows our topology (same as (a)), with the MegaMIMO leaders and associated clusters labeled.

In Chorus all nodes in the network are synchronized and hence can transmit concurrently without interference. In contrast, in MegaMIMO, nodes belonging to different clusters are not synchronized and hence cannot transmit concurrently without interference. Thus, a leader-based protocol like MegaMIMO needs to run a MAC protocol to arbitrate the medium between adjacent MegaMIMO clusters. Designing such a distributed inter-cluster MAC protocol is a complex task. Thus, we take a conservative approach and allow MegaMIMO to use a perfect and centralized oracle that arbitrates the medium between the MegaMIMO clusters.



(a) Layers assigned by Chorus



(b) MegaMIMO clustering and leader assignment. Different clusters cannot transmit simultaneously since they interfere with each other.

Figure 5-10: **Testbed topology, showing the assigned layers for Chorus and the selected leaders and clusters for MegaMIMO.** Note that, unlike MegaMIMO, Chorus is self-organizing.

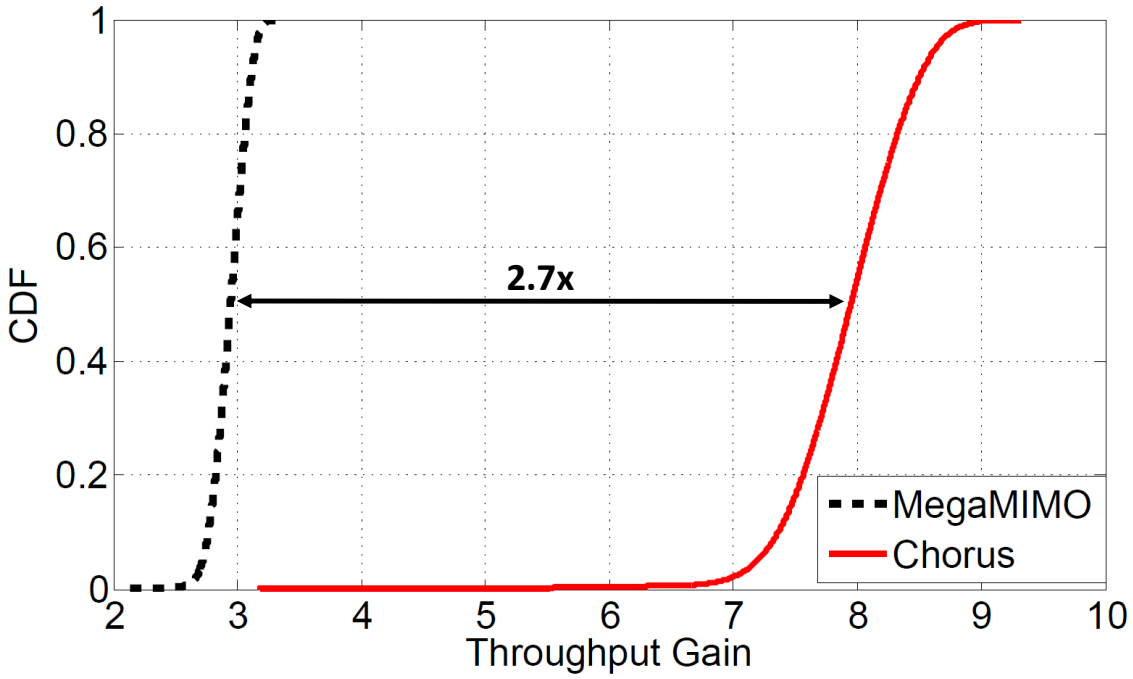


Figure 5-11: **Gains of Chorus in a distributed deployment.** Chorus can allow simultaneous transmission even across nodes that cannot hear a single leader, and hence provide throughput gains over MegaMIMO as the network scales.

We consider the case of 10 clients distributed uniformly across the entire floor. We measure the network throughput obtained with MegaMIMO, and with Chorus. We then repeat this process for various different choices of client locations.

Results. Fig. 5-11 shows the CDF of the network throughput gain of Chorus and MegaMIMO across the different client location choices. The figure shows that Chorus achieves a median throughput gain of 2.7× over MegaMIMO. Chorus achieves this throughput gain since it can synchronize all small cells in the network to transmit jointly to all 10 clients, whereas MegaMIMO will need to time multiplex between the three clusters and transmit to only 3 or 4 clients at a time. Note that, while MegaMIMO has three clusters, the median gain is less than 3× because of imbalances in the client channel matrices for different layouts.

5.9.5 Scaling the Network

Finally, we want to verify that Chorus's architecture can scale to a large geographic area with thousands of nodes. Since we cannot evaluate this large scale in actual deployment, we leverage simulation for this task.

We perform our simulation in a dense urban setting. Each small cell in our system is an urban micro cell with transmit power of 1W, as described in Page 29 of the 3GPP TS 136.104 v 12.7.0 LTE standard [19]. We use the Non Line of Sight Propagation Model for a Manhattan grid layout, as described on Page 93 of 3GPP TR 36.814 V9.0.0 [16]. Receivers have a noise floor of -100 dB, which corresponds to the standard noise figure of 6 dB (defined in the LTE TR 36 931 document [17]) at 27° C for a bandwidth of 5 MHz. With these parameters, the SNR in our network attenuates to 0 dB at around 300 m. In order to evaluate dense deployments, we consider a case where small cells are separated by an average distance of 50 m, in a 8 km $\times$ 8 km grid, which roughly corresponds to the size of Manhattan. Thus, our simulation has 25600 small cell nodes deployed uniformly at random in this grid. We have an oscillator model for each node, with a carrier frequency offset of $\pm 100ppb$, which corresponds to the LTE hardware tolerance defined in TR 36 104 [20, 7], and a phase noise of -88 dBc at 100 KHz for a bandwidth of 5 MHz, which is typical for the voltage, temperature and oven controlled oscillators used in small cells [50]. Each small cell node in our system runs the Chorus controller to ensure phase synchronization in the presence of oscillator offsets. We run the layering algorithm on our simulated deployment, and then simulate the operation of the network for a duration of 5 seconds, which is significantly larger than the typical channel coherence time of 100-200 ms.

Fig. 5-12 visualizes the results of the layering algorithm, showing that this network has 19 layers. Since Chorus has 8 distinct synchronization frequencies, the layers reuse these frequencies as described in 5.3.

Since we have access to the absolute phases of our oscillators in our simulation, we can compute the absolute phase difference, and consequently the phase variance, between pairs of nodes in our system. We plot the median of this phase variance as

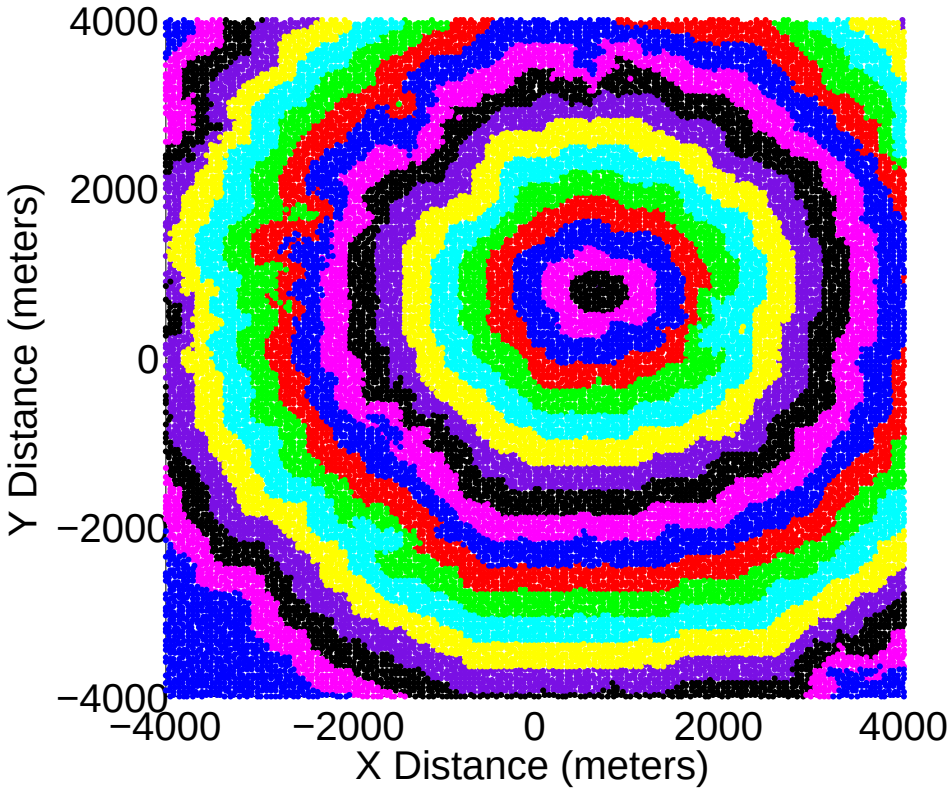


Figure 5-12: **Visualization of Chorus's layering scheme**. The figure depicts the different layers assigned by Chorus's layering algorithm. It shows that across the geographical range of this simulation, Chorus reuses synchronization frequencies.

a function of the distance between node pairs. Since the number of node pairs in our system is very large, the graph is generated by subsampling the node pairs.

Fig. 5-13 shows the median phase variance as a function of distance. As one would expect, the phase variance increases as the distance between node pairs increases. Further, the phase variance stays less than 0.004 for all nodes within distance 5 km of each other. For nodes farther apart than this distance, the higher phase variance is irrelevant because the nodes are separated by many multiples of the interference range and therefore, their signals do not interact. This shows that Chorus's synchronization fabric scales to large networks. Of course this does not mean that thousands of nodes will transmit concurrently to the same client set. Rather it shows that Chorus can create an abstraction of distributed phase coherence across thousands of nodes. The higher layers can then treat the network as they would treat a very large massive

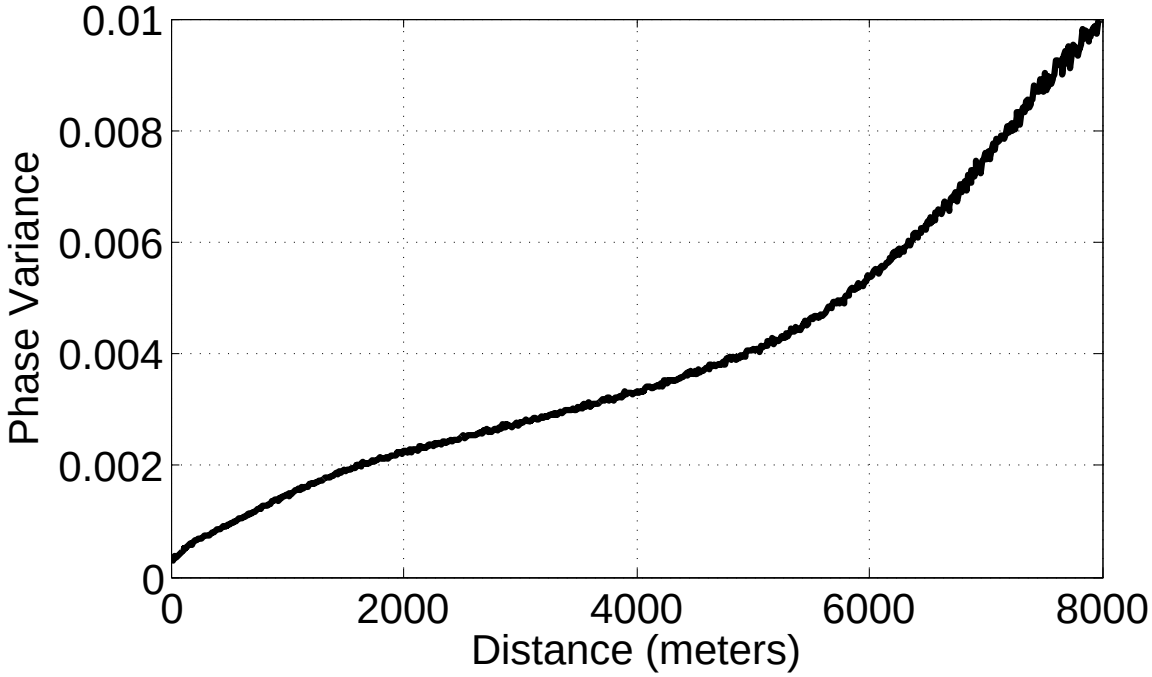


Figure 5-13: **Phase variance as a function of distance.** The figure shows that, while phase variance increases as a function of distance between the nodes, it stays within 0.004 for nodes within 5 km of each other. This implies that Chorus provides tight phase synchronization for nodes significantly longer than the interference range of nodes, and hence enables joint transmission from multiple transmitters.

MIMO system, and become free to combine any subset of nearby base stations to create multiplexing or diversity gains for the clients.

Chapter 6

Conclusion

In thes thesis we presented MegaMIMO 2.0, the first full fledged real-time PHY capable of supporting distributed MIMO for WiFi. 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.

We also presented Chorus, a new design for LTE small cells distributed MIMO. Chorus use a fully distributed phase synchronization protocol, where all nodes contribute equally to propagating the synchronization signal. Chorus allows dense deployments of 5G small cell networks to coordinate their transmissions, eliminate interference, and deliver large throughput gains. The resulting distributed MIMO network is scalable, resilient to failures and changes in connectivity. We have integrated Chorus with an open source LTE stack library, and demonstrated that it can deliver tight synchronization at scale, and distributed-MIMO throughput gains with unmodified end-user devices. We believe that Chorus's flexible and self-healing design will enable distributed-MIMO to truly move from small scale demonstrations to practical, manageable use in large networks.

Bibliography

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

[2] Omid Abari, Hariharan Rahul, and Dina 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), june 2007.

[4] S. M. Alamouti. A simple transmit diversity technique for wireless communications. *IEEE Journal on Selected Areas in Communications*, 16(8):1451–1458, Oct 1998.

[5] H.V. 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.

[6] S. Berger and A. Wittneben. Carrier phase synchronization ofof multiple distributed nodes in a wireless network. In *2007 IEEE 8th Workshop on Signal Processing Advances in Wireless Communications*, pages 1–5, June 2007.

[7] D. BladsjÃű, M. Hogan, and S. Ruffini. Synchronization aspects in lte small cells. *IEEE Communications Magazine*, 51(9):70–77, September 2013.

[8] H. Bolcskei, D. Gesbert, and A. J. Paulraj. On the capacity of ofdm-based spatial multiplexing systems. *IEEE Transactions on Communications*, 50(2):225–234, Feb 2002.

[9] Archana Bommani. A survey on inter-cell interference reduction techniques in lte-a heterogeneous networks. *IJISET - International Journal of Innovative Science, Engineering & Technology*, April 2015.

[10] 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.

[11] L. H. Brandenburg and A. D. Wyner. Capacity of the gaussian channel with memory: The multivariate case. *The Bell System Technical Journal*, 53(5):745–778, May 1974.

[12] Ultra-dense heterogeneous small cell deployment in 5g and beyond. `http://www.comsoc.org/netmag/cfp/ultra-dense-heterogeneous-small-cell-deployment-in-5g-beyond`. IEEE Comsoc.

[13] Distributed antenna systems (das) market worth 10.78 billion usd by 2022. `http://www.prnewswire.co.in/news-releases/distributed-antenna-systems-das-market-worth-1078-billion-usd-by-2022-606265916.html`. PR Newswire.

[14] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM.

[15] 5g live test: Multipoint connectivity with distributed mimo. `https://www.youtube.com/watch?v=jCO68dPoNwA`. Ericsson Inc.

[16] ETSI. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects (Release 9). *ETSI*, 2010.

[17] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Frequency (RF) requirements for LTE Pico Node B (3GPP TR 36.931 version 9.0.0 Release 9). *ETSI*, 2011.

[18] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 10.4.0 Release 10). *ETSI*, 2012.

[19] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (3GPP TS 136.104 version 12.7.0 Release 12). *ETSI*, 2015.

[20] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (3GPP TS 36.104 version 12.6.0 Release 12) . *ETSI*, 2015.

[21] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 13.0.0 Release 13). *ETSI*, 2016.

[22] ETSI. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 13.0.0 Release 13) . *ETSI*, 2016.

[23] Y. Fei, Y. Fan, B. K. Lau, and J. S. Thompson. Optimal single-port matching impedance for capacity maximization in compact mimo arrays. *IEEE Transactions on Antennas and Propagation*, 56(11):3566–3575, Nov 2008.

[24] In 2016, how much lte spectrum do verizon, at&t, t-mobile and sprint have – and where? `https://www.fiercewireless.com/wireless/2016-how-much-lte-spectrum-do-verizon-at-t-t-mobile-and-sprint-have-and-where`.

[25] Verizon sees small cell, dark fiber strategies as differentiators in 5g world. `http://www.fiercewireless.com/tech/verizon-sees-small-cell-dark-fiber-strategies-as-differentiators-5g-world-report`. Fierce Wireless Verizon report.

[26] Antonio Forenza, Robert W. Heath Jr., and Stephen G. Perlman. *System and Method For Distributed Input-Distributed Output Wireless Communications*. U.S. Patent Application number 20090067402.

[27] G. J. Foschini. Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. *Bell Labs Technical Journal*, 1(2):41–59, Autumn 1996.

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

[29] D. Gesbert, H. Bolcskei, D. A. Gore, and A. J. Paulraj. Outdoor mimo wireless channels: models and performance prediction. *IEEE Transactions on Communications*, 50(12):1926–1934, Dec 2002.

[30] M. Guillaud, D.T.M. 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.

[31] L. Hanlen and M. Fu. Wireless communication systems with-spatial diversity: a volumetric model. *IEEE Transactions on Wireless Communications*, 5(1):133–142, Jan 2006.

[32] J. Hoydis, S. ten Brink, and M. Debbah. Massive mimo in the ul/dl of cellular networks: How many antennas do we need? *IEEE Journal on Selected Areas in Communications*, 31(2):160–171, February 2013.

[33] V. Jungnickel, S. Jaeckel, L. Thiele, L. Jiang, U. Kruger, A. Brylka, and C. von Helmolt. Capacity measurements in a cooperative mimo network. *IEEE Transactions on Vehicular Technology*, 58(5):2392–2405, Jun 2009.

[34] V. Jungnickel, S. Jaeckel, L. Thiele, U. Krueger, A. Brylka, and C. von Helmolt. Wlc06-1: Capacity measurements in a multicell mimo system. In *IEEE Globecom 2006*, pages 1–6, Nov 2006.

[35] A. Kaye and D. George. Transmission of multiplexed pam signals over multiple channel and diversity systems. *IEEE Transactions on Communication Technology*, 18(5):520–526, October 1970.

[36] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[37] Leslie Lamport. Acm sigact news. *SIGACT News*, 32(4):18–25, December 2001.

[38] Ke Liu, V. Raghavan, and A. M. Sayeed. Capacity scaling and spectral efficiency in wide-band correlated mimo channels. *IEEE Transactions on Information Theory*, 49(10):2504–2526, Oct 2003.

[39] S. L. Loyka. Channel capacity of mimo architecture using the exponential correlation matrix. *IEEE Communications Letters*, 5(9):369–371, Sept 2001.

[40] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang. An overview of massive mimo: Benefits and challenges. *IEEE Journal of Selected Topics in Signal Processing*, 8(5):742–758, Oct 2014.

[41] David A. B. Miller. Communicating with waves between volumes: evaluating orthogonal spatial channels and limits on coupling strengths. *Appl. Opt.*, 39(11):1681–1699, Apr 2000.

[42] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition, 2001.

[43] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA, 2014. USENIX Association.

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

[45] Arogyaswami Pauraj and Thomas Kailath. *Increasing Capacity in Wireless Broadcast Systems Using Distributed Transmission/Directional Reception (DTDR)*. U.S. Patents no. 5,345,599 (Sep. 1994).

[46] Eldad Perahia and Robert Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. Cambridge University Press, 2013.

[47] T. S. Pollock, T. D. Abhayapala, and R. A. Kennedy. Antenna saturation effects on mimo capacity. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 4, pages 2301–2305 vol.4, May 2003.

[48] A. S. Y. Poon, R. W. Brodersen, and D. N. C. Tse. Degrees of freedom in multiple-antenna channels: a signal space approach. *IEEE Transactions on Information Theory*, 51(2):523–536, Feb 2005.

[49] Hariharan Rahul, Swarun Kumar, and Dina Katabi. MegaMIMO: Scaling Wireless Capacity with User Demands. In *ACM SIGCOMM 2012*, Helsinki, Finland, August 2012.

[50] OCXO Overview. `http://hypertech.co.il/wp-content/uploads/2015/12/Frequency-Control-Solutions-OCXO.pdf`. Rakon.

[51] G. G. Raleigh and J. M. Cioffi. Spatio-temporal coding for wireless communication. *IEEE Transactions on Communications*, 46(3):357–366, Mar 1998.

[52] R. Rogalin, O. Y. Bursalioglu, H. Papadopoulos, G. Caire, A. F. Molisch, A. Michaloliakos, V. Balan, and K. Psounis. Scalable synchronization and reciprocity calibration for distributed multiuser mimo. *IEEE Transactions on Wireless Communications*, 13(4):1815–1831, April 2014.

[53] A. A. M. Saleh, A. Rustako, and R. Roman. Distributed antennas for indoor radio communications. *IEEE Transactions on Communications*, 35(12):1245–1251, December 1987.

[54] M. Sellathurai and S. Haykin. Turbo-blast for high-speed wireless communications. In *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*, volume 1, pages 315–320 vol.1, 2000.

[55] Wei-Liang Shen, Kate Ching-Ju Lin, Ming-Syan Chen, and Kun Tan. Client as a first-class citizen: Practical user-centric network MIMO clustering. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016.

[56] Clayton Shepard, Narendra Anand, and Lin Zhong. Practical performance of mu-mimo precoding in many-antenna base stations. In *Proceeding of the 2013 Workshop on Cellular Networks: Operations, Challenges, and Future Design*, CellNet '13, pages 13–18, New York, NY, USA, 2013. ACM.

[57] Clayton Shepard, Hang Yu, Narendra Anand, Erran Li, Thomas Marzetta, Richard Yang, and Lin 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.

[58] Jing Shi, Qinglin Luo, and Mingli 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.

[59] Da-Shan Shiu, G. J. Foschini, M. J. Gans, and J. M. Kahn. Fading correlation and its effect on the capacity of multielement antenna systems. *IEEE Transactions on Communications*, 48(3):502–513, Mar 2000.

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

[61] srsLTE. Open source 3gpp lte library. https://github.com/srsLTE.

[62] V. Tarokh, H. Jafarkhani, and A. R. Calderbank. Space-time block codes from orthogonal designs. *IEEE Transactions on Information Theory*, 45(5):1456–1467, Jul 1999.

[63] V. Tarokh, N. Seshadri, and A. R. Calderbank. Space-time codes for high data rate wireless communication: performance criterion and code construction. *IEEE Transactions on Information Theory*, 44(2):744–765, Mar 1998.

[64] Emre Telatar. Capacity of multiâĂŘantenna gaussian channels. *European Transactions on Telecommunications*, 10(6):585–595, 1999.

[65] 10x, 100x, 1000x capacity growth will require small cells. `https://www.qualcomm.com/invention/technologies/1000x/small-cells`. Small Cells World Summit.

[66] I. Thibault, G. E. Corazza, and L. Deambrogio. Phase synchronization algorithms for distributed beamforming with time varying channels in wireless sensor networks. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 77–82, July 2011.

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

[68] W. van Etten. Maximum likelihood receiver for multiple channel transmission systems. *IEEE Transactions on Communications*, 24(2):276–283, Feb 1976.

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

[70] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela. V-blast: an architecture for realizing very high data rates over the rich-scattering wireless channel. In *1998 URSI International Symposium on Signals, Systems, and Electronics. Conference Proceedings (Cat. No.98EX167)*, pages 295–300, Sep 1998.

[71] Vivek Yenamandra and Kannan 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.

[72] Hang Yu, Oscar Bejarano, and Lin Zhong. Combating inter-cell interference in 802.11ac-based multi-user mimo networks. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 141–152, New York, NY, USA, 2014. ACM.

[73] Hongyuan Zhang and Huaiyu Dai. On the capacity of distributed mimo systems. In *Proc. Conf. Inform. Sciences and Systems (CISS)*, 2004.

[74] Xinyu Zhang, Karthikeyan Sundaresan, Mohammad A. (Amir) Khojastepour, Sampath Rangarajan, and Kang G. Shin. Nemox: Scalable network mimo for wireless networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, 2013.