Chorus: Truly Distributed Distributed-MIMO

Ezzeldin Hamed
Microsoft
ezhamed@microsoft.com

Hariharan Rahul
MIT CSAIL
rahul@csail.mit.edu

Bahar Partov
Wavelite
bahar@wavelite.net

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. However, these systems only distribute the transmission function across the multiple nodes. The control fabric that synchronizes the nodes to a common reference phase still fundamentally requires a single leader that all nodes in the network are capable of hearing.

This paper presents Chorus, a truly distributed distributed-MIMO system. Chorus is leaderless – all nodes are peers, and jointly transmit the synchronization signal used by other nodes to synchronize to a common reference phase. The participation of all nodes in the network in the synchronization signal enables Chorus to scale to large networks, while being resilient to node failures or changes in network connectivity, and without imposing onerous management burdens on network administrators. We implement and evaluate Chorus and demonstrate that it can synchronize effectively without the need for a single leader, scale to large networks where no leader node can be heard by all others, and provide 2.7× throughput improvement over traditional leader-based systems.

CCS CONCEPTS

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

KEYWORDS

Wireless Networks, Multi-user MIMO, Distributed MIMO, LTE, Synchronization

ACM Reference Format:
Ezzeldin Hamed, Hariharan Rahul, and Bahar Partov. 2018. Chorus: Truly Distributed Distributed-MIMO . In *SIGCOMM '18: ACM SIGCOMM 2018 Conference, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3230543.3230578

SIGCOMM '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230578

1 INTRODUCTION

Distributed-MIMO can eliminate interference and deliver dramatic throughput improvements in wireless networks [2, 31, 40, 45]. It does so by synchronizing the oscillators on independent nodes, allowing a network of transmitters to act as if they were one huge MIMO transmitter. The theory underlying distributed MIMO has been around for several decades [48]. Recent years however have seen significant advances in moving distributed-MIMO from theory to practice [1, 5, 23, 33, 36, 46]. Multiple systems have developed coordination protocols to synchronize the phase of distributed oscillators, thereby allowing them to transmit together without interference.

Yet, existing systems only focus on enabling the transmission functionality of signals to be distributed. The control plane itself, which coordinates the transmitters, is still fairly centralized [1, 5, 23, 33, 36, 49]. Specifically, in past systems, the transmitters are typically organized around an architecture consisting of 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 expected in network protocols: scalability, resilience, and ease of management. First, they do not easily scale to large networks where the transmitters cannot all hear one node. Second, they are not resilient - they fail if the leader fails or any of the slaves becomes disconnected from the leader. Third, they are difficult to deploy and manage. The network administrator has to pick the nodes' positions carefully to ensure they all hear the leader. The administrator cannot simply add or remove nodes, and has to monitor the system to ensure that the connectivity constraints always hold.

Bringing scalability, resilience, and manageability 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.* [10, 21, 44]. 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.

In this paper, we introduce Chorus, a system that removes these limitations and builds a truly distributed Distributed-MIMO network. 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. This makes the design resilient to node failure, addition and removal. 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.

Chorus is different from standard distributed consensus protocols (*e.g.*, Paxos or Raft [25, 26, 30]). In those protocols, the state of the system advances or rolls back only based on interactions between the nodes in the system, and is controlled only by the protocol. In contrast, Chorus's distributed protocol tracks an underlying analog state of the world (specifically, a reference oscillator phase) that advances independently of the protocol. This analog state cannot be controlled or rolled back by the distributed protocol, instead the role of the protocol is to ensure that all nodes in the system accurately track the state of the reference oscillator to within tens of nanoseconds.

The design of Chorus has the following three components that together deliver a fully distributed phase synchronization protocol for distributed-MIMO.

(a) Self-Organizing Tree Architecture: As described earlier, 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 [29]. 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 §5.

(b) Robust Phase Update Algorithm: In past work on distributed MIMO, each transmitter listens to the leader's signal, computes the difference between its phase and that of the leader, and adds the difference to its own phase [1, 5, 23, 33, 36, 49]. This simple algorithm however does not work for Chorus. In Chorus, the synchronization signal is no longer a clean transmission from one leader – it is a composition of synchronization signals from multiple nodes. Thus, there is more phase variability due to potential misalignment between the transmitters of the composite signal. Furthermore, in past systems, the phase difference can be computed at the time of transmission and applied immediately. In contrast, in Chorus, the synchronization signal can be sent only sporadically due to the frame structure of LTE. As a result, the measured phase difference can be outdated. To deal with multiple transmitters, as well as its more stringent synchronization requirements, Chorus uses tools from signal processing and control theory. Specifically, instead of using a known synchronization signal, Chorus randomizes the synchronization signal sent by each transmitter in order to ensure resilience to channel conditions. Additionally, Chorus explicitly models measurement variability and delays, and incorporates its model within the framework of robust control, which is known to account for these uncertainties. In §6, we describe our controller formulation.

(c) LTE Compatibility: We would like Chorus to be directly applicable to 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 [42] 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 $radians^2$ 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 MegaMIMO [23, 33], a state-of-the-art leader-based distributed MIMO system.[1] Chorus delivers 2.7× higher throughput than MegaMIMO 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, MegaMIMO 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 $radians^2$. 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.

2 RELATED WORK

Related work falls in five categories:

(a) Distributed MIMO Schemes for Wi-Fi and LTE: The typical design of distributed MIMO systems assume a leader/coordinator that plays a special role in synchronizing the oscillators of the nodes. Such a design is used in distributed MIMO schemes proposed for both Wi-Fi and LTE, such as MegaMIMO [23, 33], AirSync [5], AirShare [1] for Wi-Fi, as well as Co-ordinated Multi Point (CoMP) for LTE, which is implemented in example systems such as PCell [22, 32] and a demonstration by Ericsson [13]. In particular, MegaMIMO and AirSync are organized around a single leader that all slaves must be able to hear. AirShare [1] transmits the synchronization signal over a separate out-of-band channel, using a network of lead emitter and slave emitters. The CoMP schemes assume a shared clock, distributed either via GPS or a wire, and a dedicated central server that creates the signals for all participating base stations, and delivers them to all antennas using a dedicated fiber backhaul infrastructure with very high throughput and carefully controlled latencies. As such, these systems do not deliver the benefits of a truly distributed design. They do not scale to a large network of small cells, they are not resilient to network faults, and are complicated to manage.

There are some previous schemes that have considered scaling beyond a single leader. For example, NEMOx [49] 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 the leader 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 [35] 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 [46] 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.

(b) Distributed Coordination Protocols: 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 [11, 25, 26, 30]. 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 Chorus. 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 Chorus 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.

(c) Massive MIMO: A popular recent trend in LTE is massive MIMO [24, 28, 38, 39]. For instance, systems such as Argos have demonstrated designs where a large number of MIMO antennas are packed densely on a single node. Chorus 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

[1] MegaMIMO is designed for a Wi-Fi network with one leader, but we adapt it to larger small cell networks as we describe in §11.4.

chains. In contrast, Chorus is targeted at small cells, which have a much smaller form factor and lower power budget.

(d) Inter-Cell Coordination without Frequency Reuse: 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* [7]. Similarly, schemes have been proposed to achieve such inter-cell interference elimination in 802.11ac [47]. 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.

(e) Other network synchronization techniques: There is a significant literature on time synchronization for wired and wireless networking applications. However, all those techniques focus only on time synchronization and are not capable of providing phase synchronization, which is essential for beamforming. Those techniques for network synchronization fall into two major categories.

Theoretical: Dorfler et. al [12] discuss various theoretical models of oscillators and the performance bounds associated with different levels of synchronization. The paper however does not describe algorithms that can achieve the different levels of synchronization, nor does it explore the system issues of designing and building a network of synchronized transmitters. Antonioni et. al. [4] discuss the game theoretic tradeoff of synchronization in natural systems, such as fireflies, based on the costs of synchronizing and the benefits that would accrue. As such, this work neither provides algorithms that achieve accurate synchronization, nor do these tradeoffs apply to MIMO beamforming.

Practical: Chen et. al. [8] , Sommer et. al [41] evaluate various gradient clock synchronization algorithms in sensor networks. These algorithms provide tight time synchronization by flooding the network with messages and then using these message timestamps to discipline their local clocks. However, they do not provide phase synchronization, which is needed for beamforming. NTP is similar, and further, achieves time synchronization only at the accuracy of ms, which is inadequate for joint MIMO transmission. Further, none of these algorithms have been evaluated across a wide range of SNRs, which is actually necessary for a synchronization system to work robustly in practice, nor do they address the practical challenges of interoperability with standards like LTE.

3 SCOPE

As in prior work on Distributed-MIMO, this paper focuses on delivering phase coherence across distributed independent nodes [1, 5, 23, 33, 46], thereby allowing those nodes to act as one huge MIMO node which can perform the basic primitives of multi-user transmission, such as nulling, diversity, and multi-user beamforming. All of these systems focus on the physical layer signal transmission, and the associated necessary coordination protocols.

In today's multi-user and massive MIMO systems, there are additional functions that are performed by the higher layers. For example, the higher layers decide the frequency and time slots in which each user receives its data, as well as the choice of transmit antennas, and the combinations of beamforming, nulling, and diversity to be used for the different users. Much work has been done in both academia and industry to develop algorithms for these higher layers [3, 9, 27, 36, 37, 43]. The higher layers in distributed-MIMO also have to address these questions. Given the similarity, they can leverage the past work for existing MIMO systems, but have to account for the greater complexity due to the bigger scale of a distributed-MIMO system and the additional system constraints such as the amount of backhaul bandwidth available for each small cell. Dealing with these issues is beyond the scope of this paper, and is left for future work.

4 OVERVIEW

Chorus is designed for 5G cellular networks with dense small cell deployments containing tens of thousands of small cells spanning multiple tens of square kilometers. These dense networks can obtain dramatic throughput gains from distributed-MIMO as their throughput is currently limited by interference. However, their size and geographic scale make them unsuited for current distributed-MIMO solutions, which require a single leader that needs to be reliably heard by all other small cells in order to perform phase synchronization.

Chorus addresses these challenges by designing a scheme that does not require a single leader transmitting a synchronization signal. Instead, all nodes act as leaders, jointly transmitting the synchronization signal. Each node locally has a controller that listens to the synchronization signal transmitted by other nodes, and synchronizes the phase of its oscillator to the signal. As a result, Chorus can synchronize the phase of hundreds to thousands of nodes in a large geographic area, creating a distributed massive MIMO fabric that can be leveraged by higher layers. Note however that this does not mean that thousands of nodes have to transmit data together to the same set of clients. The distributed MIMO fabric allows the higher layers to treat any subset of the network as a large massive MIMO system. They can freely pick groups of nodes to deliver multiplexing or diversity gain to specific client sets. They can also change those groups from one transmission to another without worrying about which small cells hear each other.

Chorus's protocol and architecture is designed to be compatible with LTE, and operates without requiring any changes to LTE end user devices. We describe the different components of Chorus in the subsequent sections.

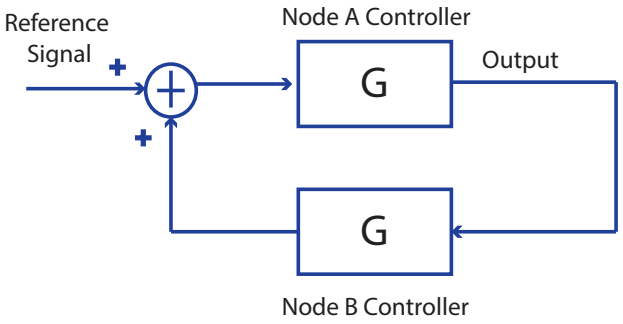


Figure 1—Positive Feedback between two nodes. The figure shows two nodes transmitting and receiving synchronization signals to each other. Each node can be modeled as a control system with transfer function G. Node A receives a reference synchronization signal from node B, and transmits a synchronization signal, which in turn is used by Node B as its input synchronization signal, producing a positive feedback loop.

5 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. 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 turn transmit the synchronization signal that node A uses. This leads to a feedback loop, as shown in Fig. 1. From basic control theory [29], 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.

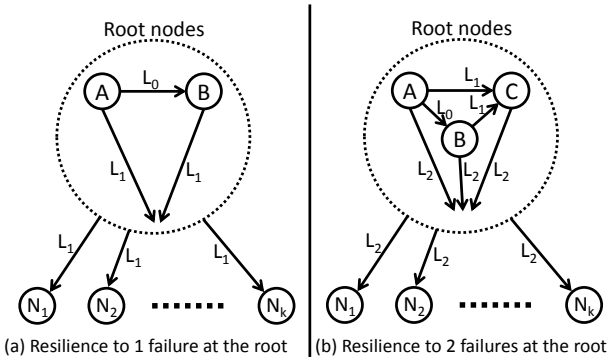


Figure 2—Resilience to Root Failures: (a) shows a topology for the synchronization tree that is resilient to either node A or B failing. (b) shows a topology that is resilient to any two nodes of {A,B,C} failing.

5.1 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's synchronization signals are associated with a different set of frequencies. 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 §6) 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 §11.2, 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 §7, we describe how these synchronization frequencies can be provided cheaply in the LTE framework without allocating dedicated frequency bands.

5.2 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. 2(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. 2(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.

6 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 §7). Every time the synchronization signal is available, the small cell obtains new measurements, which it uses to update its phase so that it maintains phase coherence with the received synchronization signal. But, how should a Chorus node use the measurements to update its phase?

Past distributed MIMO update rules are not suitable for Chorus. Specifically, past work belongs to two categories. Systems like AirSync [5] have the synchronization signal all the time on a dedicated channel. Thus, each node can continuously compute the phase difference between its signal and the received synchronization signal, and compensate for that difference. Systems like MegaMIMO [23, 33] transmit the synchronization signal immediately before every packet and use it only for the duration of that packet – i.e., they synchronize on demand and immediately before transmission. Chorus's requirements are more stringent than either system – it needs to provide continuous phase tracking to accommodate LTE's continuous data transmission, but can afford only infrequent synchronization signals (every few milliseconds) in order to integrate into the LTE framework with low overhead. Furthermore, Chorus's synchronization signal exhibits a higher variability because it is transmitted by multiple nodes as opposed to a single leader.

To deal with these more stringent conditions, Chorus combines techniques from signal processing and control theory. First, Chorus designs its synchronization signal to be resilient to channel conditions in the presence of multiple transmitters. It then uses principles from robust control to perform phase correction using this synchronization signal.

(a) Resilient Synchronization Signal: In order to provide resilience against destructive interference and changes in the network structure, Chorus designs the synchronization signal sent from multiple nodes to be uncorrelated. Specifically, each node chooses a random signal to be transmitted on each resource element used for synchronization, and sends this signal as a repeated pattern in the synchronization subcarriers every 5 milliseconds. Sending a different random signal from each node ensures that the synchronization signals from different transmitters do not destructively combine in all the subcarriers independent of the channels from the different transmitters to every receiver. The receiver then correlates successive copies of the synchronization signal to compute the phase change accumulated in the 5 millisecond duration.

(a) Controller: Chorus formulates phase tracking as a control problem. it then addresses it within the framework of robust control, which is particularly suitable for dealing with uncertainties and noisy feedback. Specifically, Chorus models the system as a combination of a **control block** and a **plant**. The objective of the control loop is to make the *error signal*, which is the difference between the phase of the actual received synchronization signal and the output phase of the plant (i.e., the node), go to zero. The output phase of the plant is used to

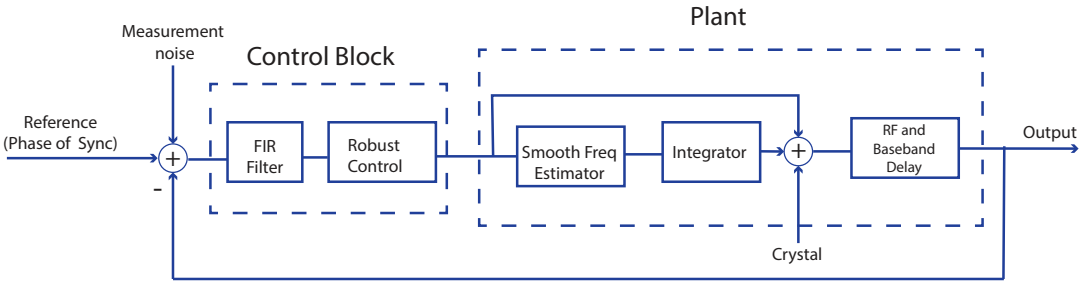


Figure 3—A Block-diagram of Chorus's Controller. Chorus uses a robust controller which takes as input a smoothed version of the phase error. The controller output is used to update a frequency drift and phase update. The plant refers to the controlled system which models the delays in the baseband.

modulate and demodulate the transmitted and received signals at the node, and is intended to track the phase of the received synchronization signal.

Fig. 3 shows a schematic of our system.

- The **control block** takes as input an error signal corresponding to the difference between the actual phase of the received synchronization signal and the output phase of plant (with added measurement noise). It then uses a Finite Impulse Response (FIR) filter to reduce variability and smooth this input signal. It feeds this smoothed value to a robust controller that produces a phase correction to be used by the plant.
- The **plant** consists of the RF and baseband systems which incur some processing delay, and a correction module that utilizes the phase output of the control block to perform frequency and phase correction of the baseband system. Specifically, the correction module uses the phase correction output of the control block to produce a smoothed frequency estimate, which it then feeds through an integrator to produce an additional phase correction per sample. The output of the integrator is added to the control block phase correction to produce a combined phase correction. The local crystal has frequency drift due to temperature and other variations, and can be modeled as an additive disturbance to this combined phase correction, which feeds into the baseband system.

Modeling tools provide support for designing such controllers; specifically, the MATLAB Simulink toolbox can take as input a model of the system, and produce a robust controller [29]. However, typically, these controllers require the system to be Linear Time Invariant (LTI). Unfortunately, phase is non-linear since it wraps around. Hence, in order to use these controllers effectively, one has to ensure that the system operates in a range where the measured phase does not wrap around. To ensure this, Chorus has an initialization phase that estimates the coarse frequency and timing offset between the synchronization signal and the local node, and corrects for it. Chorus uses the standard OFDM CFO correction algorithm for coarse frequency offset correction, and leverages the LTE frame structure for time synchronization as described in §7. It then applies the coarse corrections in the baseband of the plant before starting the controller.

7 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.

7.1 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 [16, 19, 20]. 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.

7.2 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.[2]

7.3 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.

[2]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.

7.4 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.

8 IMPLEMENTATION

We prototype Chorus using a joint software-hardware implementation. Our prototype is integrated with srsLTE [42], an open source LTE stack library, and hence it is compatible 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 §7 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.

9 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. 8 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.

10 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.* with only one small cell transmitting to a single end user device at a time.

11 RESULTS

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

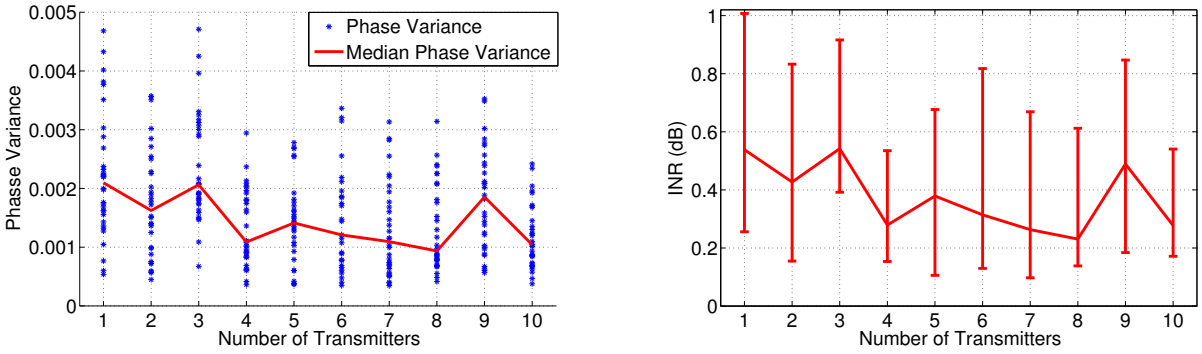
11.1 Resilience to Multiple Nodes Transmitting the Synchronization Signal

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.[3] To enable the auxiliary node to measure the phase difference between the layer 1 and layer 2 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.

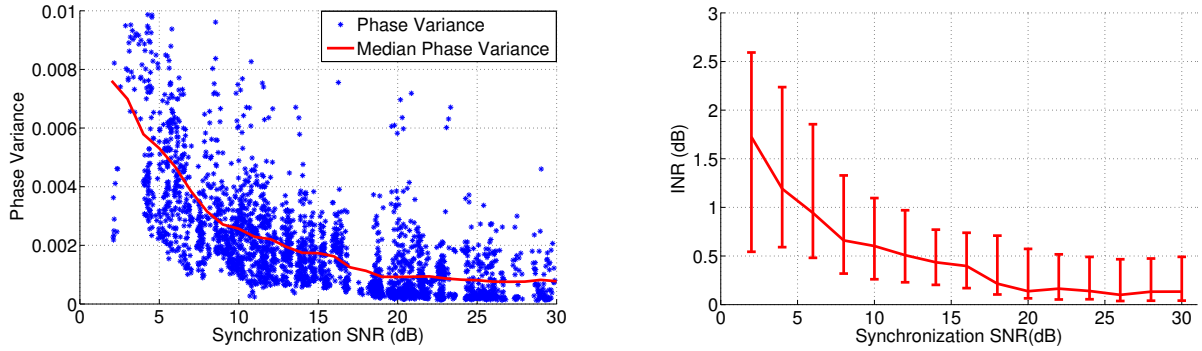
Results. Fig. 4(a) plots the variance in the phase difference between the data signals from layer 1 and layer 2 nodes as a function of the number of transmitters on layer 1. The variance is measured across 1 sec intervals which is significantly larger than the channel coherence time. If the layers are 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.

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



(a) Phase variance (b) Interference-to-Noise Ratio (INR)

Figure 4—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.



(a) Phase variance (b) Interference-to-Noise Ratio (INR)

Figure 5—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.

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. 4(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.

11.2 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 §11.1, 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(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(b). Again, we notice that the median INR is very small, less than 1 dB for synchronization SNR larger than 6 dB.

Figs. 5(a) and (b) show that by picking the minimum synchronization link SNR to be 6 dB, Chorus can ensure that the maximum interference is less than 1 dB.

11.3 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. 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 §11.2, but with the small cells in two different layers.

Results. Fig. 6 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.

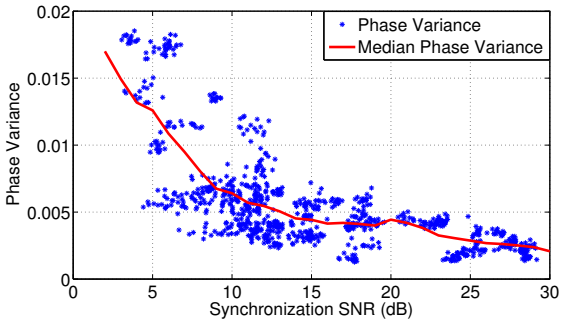


Figure 6—Phase variance between small cells that cannot hear each other and are synchronized through an intermediate node, as a function of synchronization SNR. The figure shows that Chorus can achieve high level of synchronization even when the nodes cannot hear each other.

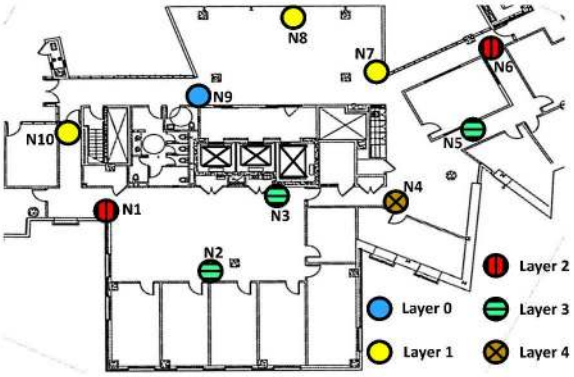
	1	2	3	4	5	6	7	8	9	10
1		24	15							14
2	24		21	10						
3	15	21		12						
4		10	12		17					
5				17		17				
6					17		12		7	
7						12		20	19	
8							20		11	
9						7	19	11		13
10	14								13	

Figure 7—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.

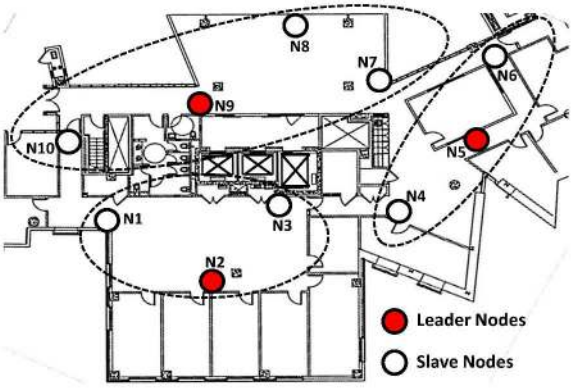
11.4 Comparison with Leader-Based Distributed MIMO

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.



(a) Layers assigned by Chorus



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

Figure 8—Testbed topology, showing the assigned layers for Chorus and the selected leaders and clusters for MegaMIMO. Note that, unlike MegaMIMO, Chorus is self-organizing.

Fig. 7 shows the link quality matrix with all links with SNR less than 6 dB (the minimum synchronization SNR, as determined in §11.2) 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. 8(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 the link-quality matrix in Fig. 7, 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. 8(b) shows our topology

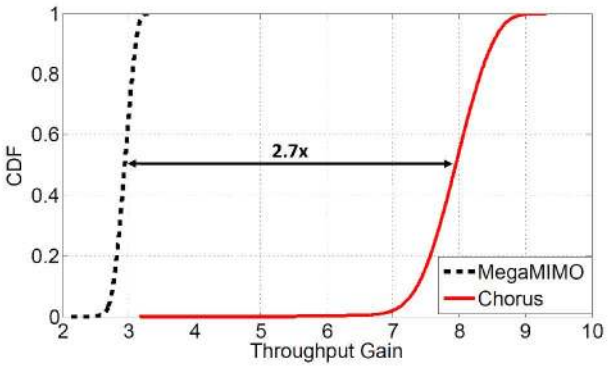


Figure 9—Gains of Chorus in a distributed deployment. Chorus can allow simultaneous transmission even across nodes that cannot hear a single leader, and hence provide throughput gains over MegaMIMO as the network scales.

(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.

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. 9 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.

11.5 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

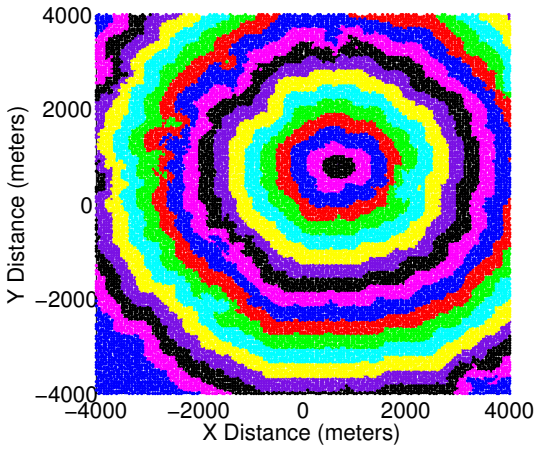


Figure 10—Visualization of Chorus's layering scheme. The figure depicts the different layers assigned by Chorus's layering algorithm. It shows that across the geographical range of this simulation, Chorus reuses synchronization frequencies.

v 12.7.0 LTE standard [17]. 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 [14]. 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 [15]) 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 × 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 [6, 18], 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 [34]. 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. §10 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.

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 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.

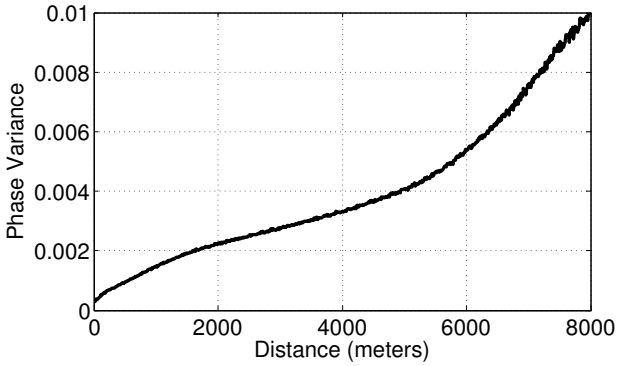


Figure 11—Phase variance as a function of distance. The figure shows that, while phase variance increases as a function of distance between the nodes, it stays within 0.004 for nodes within 5 km of each other. This implies that Chorus provides tight phase synchronization for nodes significantly longer than the interference range of nodes, and hence enables joint transmission from multiple transmitters.

Fig. 11 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. Given this large distance, no client will hear these far away nodes and hence there is no danger from having these nodes synchronized. In contrast, the phase difference between nearby nodes is bounded. 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 MIMO system, and become free to combine any subset of nearby base stations to create multiplexing or diversity gains for the clients.

12 CONCLUSION

We have presented Chorus, a new design for distributed MIMO that has 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.

REFERENCES

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

[2] 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.

[3] N. Anand, J. Lee, S. J. Lee, and E. W. Knightly. Mode and user selection for multi-user mimo wlans without csi. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 451–459, April 2015.

[4] A. Antonioni and A. Cardillo. Coevolution of synchronization and cooperation in costly networked interactions. *Phys. Rev. Lett.*, 118:238301, Jun 2017.

[5] H. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire. Airsync: Enabling distributed multiuser mimo with full spatial multiplexing. *Networking, IEEE/ACM Transactions on*, 21(6):1681–1695, Dec 2013.

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

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

[8] J. Chen, Q. Yu, Y. Zhang, H. H. Chen, and Y. Sun. Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach. *IEEE Transactions on Vehicular Technology*, 59(6):2963–2973, July 2010.

[9] R. Chen, Z. Shen, J. G. Andrews, and R. W. Heath. Multimode transmission for multiuser mimo systems with block diagonalization. *IEEE Transactions on Signal Processing*, 56(7):3294–3302, July 2008.

[10] 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.

[11] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. 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.

[12] F. DÃČÂűrfler and F. Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539 – 1564, 2014.

[13] 5g live test: Multipoint connectivity with distributed mimo. https://www.youtube.com/watch?v=jCO68dPoNwA. Ericsson Inc.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

[23] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi. Real-time distributed mimo systems. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference*, SIGCOMM '16, pages 412–425, New York, NY, USA, 2016. ACM.

[24] 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.

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

[26] L. Lamport. Acm sigact news. *SIGACT News*, 32(4):18–25, Dec. 2001.

[27] C. H. Lin, Y. T. Chen, K. C. J. Lin, and W. T. Chen. Fdof: Enhancing channel utilization for 802.11ac. *IEEE/ACM Transactions on Networking*, PP(99):1–13, 2018.

[28] 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.

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

[30] D. Ongaro and J. 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.

[31] A. Ozgur, O. Leveque, and D. Tse. Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. *IEEE Trans. on Info. Theor.*, 2007.

[32] pCell. An Introduction to pCell. Artemis, February 2015.

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

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

[35] 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.

[36] W. Shen, K. C. Lin, M. Chen, and K. 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.

[37] W. L. Shen, K. C. J. Lin, M. S. Chen, and K. Tan. Sieve: Scalable user grouping for large mu-mimo systems. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1975–1983, April 2015.

[38] C. Shepard, N. Anand, and L. 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.

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

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

[41] P. Sommer and R. Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *2009 International Conference on Information Processing in Sensor Networks*, pages 37–48, April 2009.

[42] srsLTE. Open source 3gpp lte library. https://github.com/srsLTE.
[43] An overview of algorithms for downlink transmit beamforming. https://ll.mit.edu/asap/asap_04/DAY1/16_PR_SWINDLEHURST.PDF. BYU.
[44] 10x, 100x, 1000x capacity growth will require small cells. https://www.qualcomm.com/invention/technologies/1000x/small-cells. Small Cells World Summit.
[45] S. Venkatesan et al. A WiMAX-based implementation of network MIMO for indoor wireless. *EURASIP*, '09.
[46] V. Yenamandra and K. Srinivasan. Vidyut: Exploiting power line infrastructure for enterprise wireless networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 595–606, New York, NY, USA, 2014. ACM.
[47] H. Yu, O. Bejarano, and L. 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.
[48] H. Zhang and H. Dai. On the capacity of distributed mimo systems. In *Proc. Conf. Inform. Sciences and Systems (CISS)*, 2004.
[49] X. Zhang, K. Sundaresan, M. A. A. Khojastepour, S. Rangarajan, and K. G. Shin. Nemox: Scalable network mimo for wireless networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, 2013.