A Real-time 802.11 Compatible Distributed MIMO System

Ezzeldin Hamed, Hariharan Rahul, Mohammed A. Abdelghany, and Dina Katabi
Massachusetts Institute of Technology
{ezz,rahul,melmotaz,dina}@csail.mit.edu

ABSTRACT

We present a demonstration of a real-time distributed MIMO system, DMIMO. DMIMO synchronizes transmissions from 4 distributed MIMO transmitters in time, frequency and phase, and performs distributed multi-user beamforming to independent clients. DMIMO is built on top of a Zynq hardware platform integrated with an FMCOMMS2 RF front end. The platform implements a custom 802.11n compatible MIMO PHY layer which is augmented with a lightweight distributed synchronization engine. The demonstration shows the received constellation points, channels, and effective data throughput at each client.

CCS CONCEPTS

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

1. INTRODUCTION

Recent years have seen the demand for wireless data increase by leaps and bounds, putting tremendous pressure on our limited supply of wireless spectrum. In recent years, distributed multi-user beamforming has been proposed as a solution for this spectrum crunch since it allows wireless througput to scale with the number of transmitters in the network.

Our demonstration builds on recent advances in practical distributed multi-user beamforming. Specifically, MegaMIMO [2] demonstrated an implementation using USRPs that achieved low overhead distributed time, frequency and phase synchronization across transmitters to provide distributed beamforming, and scaled throughput with the number of transmitters. However, these prior systems were not real-time or implemented in hardware.

In contrast, our system, DMIMO, implements a complete 802.11n compatible MIMO physical layer and the synchronization subsystem in hardware. A hardware implementation requires us to address real-time constraints such as integration with carrier sense and response to interference unlike previous systems. Further, we need to ensure that our time, frequency and phase synchronization subsystems are simple enough to meet the timing and resource constraints inherent to hardware, while still achieving the desired accuracy for scaling throughput with the number of transmitters.

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.
DOI: http://dx.doi.org/10.1145/2785956.2790042

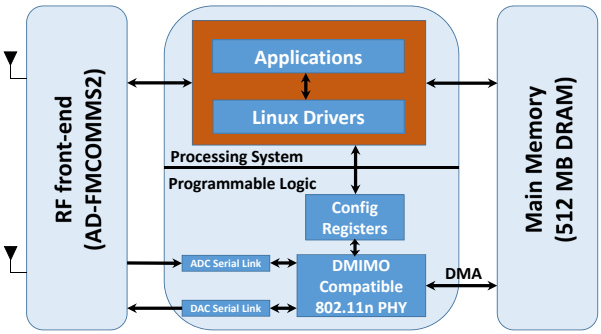

Figure 1—Platform Architecture. The figure shows our software-hardware architecture. The PHY on the FPGA implements an 802.11n MIMO system and the real time synchronization facilities needed for distributed MIMO. The software on the ARM core configures the PHY and manages data transfer to and from the device.

2. SYSTEM ARCHITECTURE

The DMIMO system consists of a hardware access point platform that performs physical layer processing of data and the received signals, and a software controller that coordinates the different transmitters to perform distributed multi-user beamforming.

2.1 Hardware

We implement DMIMO on the Xilinx Zynq platform [3] integrated with the Analog Devices FMCOMMS2/3 RF front end card [1]. The Zynq platform consists of an Artix based FPGA connected by a high speed bus to an ARM dual-core Cortex A9 processor. We implement the PHY layer for our 802.11n compatible MIMO transceiver subsystem using Verilog on the Zynq FPGA. Each MIMO transmitter can precode up to 8 independent streams to 2 antennas. Four such transmitters can therefore coordinate to jointly emulate a single 8x8 MIMO transmitter. We also run Linux on the ARM core with drivers to interact with the PHY layer. The PHY layer exposes read and write registers that enable real time configuration and monitoring by drivers, as well as DMA for bulk data transfer for packet transmission and reception. Fig. 1 shows the hardware architecture of our system.

In addition to the typical transmission and reception functionalities, our PHY has these additional configurable capabilities:

- Reporting the decoded constellation points on all the subcarriers.
- Reporting the estimated channel from the packet headers as well as decoded data.
- Joint transmission with a master transmitter, along with time, frequency, and phase synchronization.

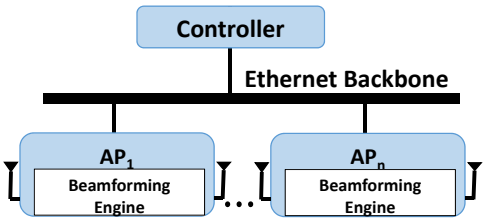

Figure 2—DMIMO Architecture. The controller orchestrates the beamforming engines on the APs.

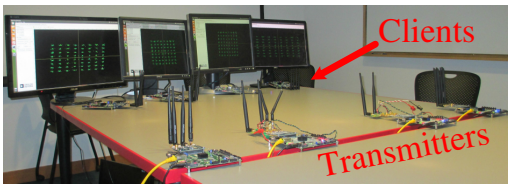

Figure 3—Testbed. The testbed consists of 4 2-antenna MIMO transmitters connected to an Ethernet backhaul, and 4 2-antenna MIMO *wireless* clients. Each client is equipped with a monitor to display parameters associated with its received signal.

2.2 Software Controller

Fig. 2 shows the DMIMO architecture. The software controller coordinates the transmitters in order to measure the channels to clients, as well as to perform distributed multi user beamforming. In particular, it schedules joint channel measurement transmissions from the master and slave transmitters as described in [2]. It then obtains the measured channel information from the clients, and computes a precoding matrix to be used by the transmitters. The controller distributes the appropriate rows of the precoding matrix to each transmitter along with the data for joint transmission, and sets registers to ensure that the master and slave transmissions are synchronized in frequency and phase. The PHY layer then performs the beamforming and synchronized transmission in hardware.

3. Demonstration

The demonstration consists of four 2-antenna MIMO transmitters capable of distributed MIMO and four 2-antenna MIMO clients as shown in Fig. 3. Each transmitter is capable of participating, along with other transmitters, in an 8 stream distributed MIMO transmission. One of the transmitters is configured as a master, and the others are configured as slaves. The transmitters are connected to an Ethernet backhaul, as can be seen by the yellow ethernet cables to the left of each transmitter device. The clients, on the other hand, are not connected to any wired network and communicate wirelessly with the transmitters. Each client is connected to a monitor, which is used to display various statistics related to the received data, for instance, constellations, channel plots, throughput *etc.*

The transmitters periodically transmit channel measurement packets to the clients. The clients use these measurement packets to estimate the channels from the different transmitters, and then transmit these measured channels back to the transmitters. The software controller, which runs on the master, computes the precoding matrix based on these measured channels and transmits the relevant rows of the precoding matrix to each transmitter. Each transmitter then precodes the 8 streams intended to the clients with its rows of the precoding matrix. Note that the clients themselves are not aware that the transmissions are joint to multiple clients, and each will decode its streams individually.

The system supports 1 and 2 stream 802.11n (*i.e.* 56 subcarriers with 4 pilots) and all the 1 and 2 stream 802.11n modulations and

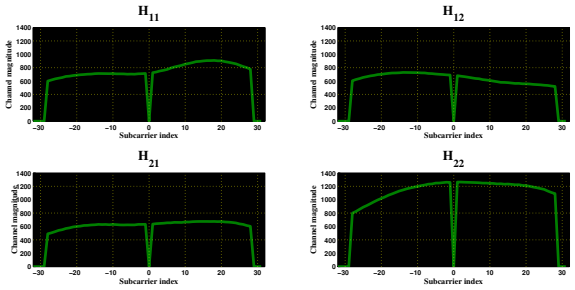

Figure 4—Channel Matrix. The green line shows the 2x2 channel matrix observed at one of the clients.

code rates (BPSK 1/2 rate, 4-QAM 1/2 rate, 4-QAM 3/4 rate, 16-QAM 1/2 rate, 16-QAM 3/4 rate, 64-QAM 2/3 rate, 64-QAM 3/4 rate, and 64-QAM 5/6 rate).

An application on the transmitters chooses the rate for joint transmission to the different clients, as well as the beamforming algorithm to be used (*e.g.* zero forcing beamforming, conjugate beamforming, no beamforming).

The clients demonstrate the following statistics:

(a) Received constellation points: This is as shown on the monitors in Fig. 3 above. The clustering of the constellation points shows the impact of the channel feedback quality and timeliness on the accuracy of beamforming, as well as the performance of the different beamforming algorithms.

(b) Channel at each client: This shows the channel magnitude across all subcarriers for the 2x2 channel matrix at each client. Note that each client receives only a 2 stream packet after the combination of all the transmissions and hence effectively sees only a 2x2 channel. Fig. 4 shows an example channel matrix at a client.

The demonstration shows how this channel matrix will vary as the controller uses different precoding techniques. For instance, when the controller uses a form of zero-forcing that nulls the contribution of all other streams at each antenna, each antenna will effectively see only one stream. As a result, the off-diagonal entries in the effective channel matrix (*i.e.*, H_{12} and H_{21}) will be close to 0.

However, such a restriction is too strict as the interference between two streams destined to the same client need not be nulled relative to each other. Providing this additional degree of freedom to the controller will allow it to compute a channel matrix which can deliver higher throughput than the strict nulling system described above. In such a case, the off-diagonal entries will be non-zero as our demonstration will show.

(c) Throughput as a function of time: The demonstration shows a graph of the effective throughput, *i.e.*, the rate of correctly received and decoded data as a function of time. The graph shows the effect of the transmitted modulation on the received throughput as a function of the quality and timeliness of the channel measurement. Further, it demonstrates the impact of various kinds of interference (single tone and wideband) on the received throughput.

References

[1] Analog devices FMCOMMS2. http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-AD-FMCOMMS2.html.

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

[3] Xilinx Zynq. http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Zynq All Programmable SOC.