Real-time Distributed MIMO Systems

Hariharan Rahul

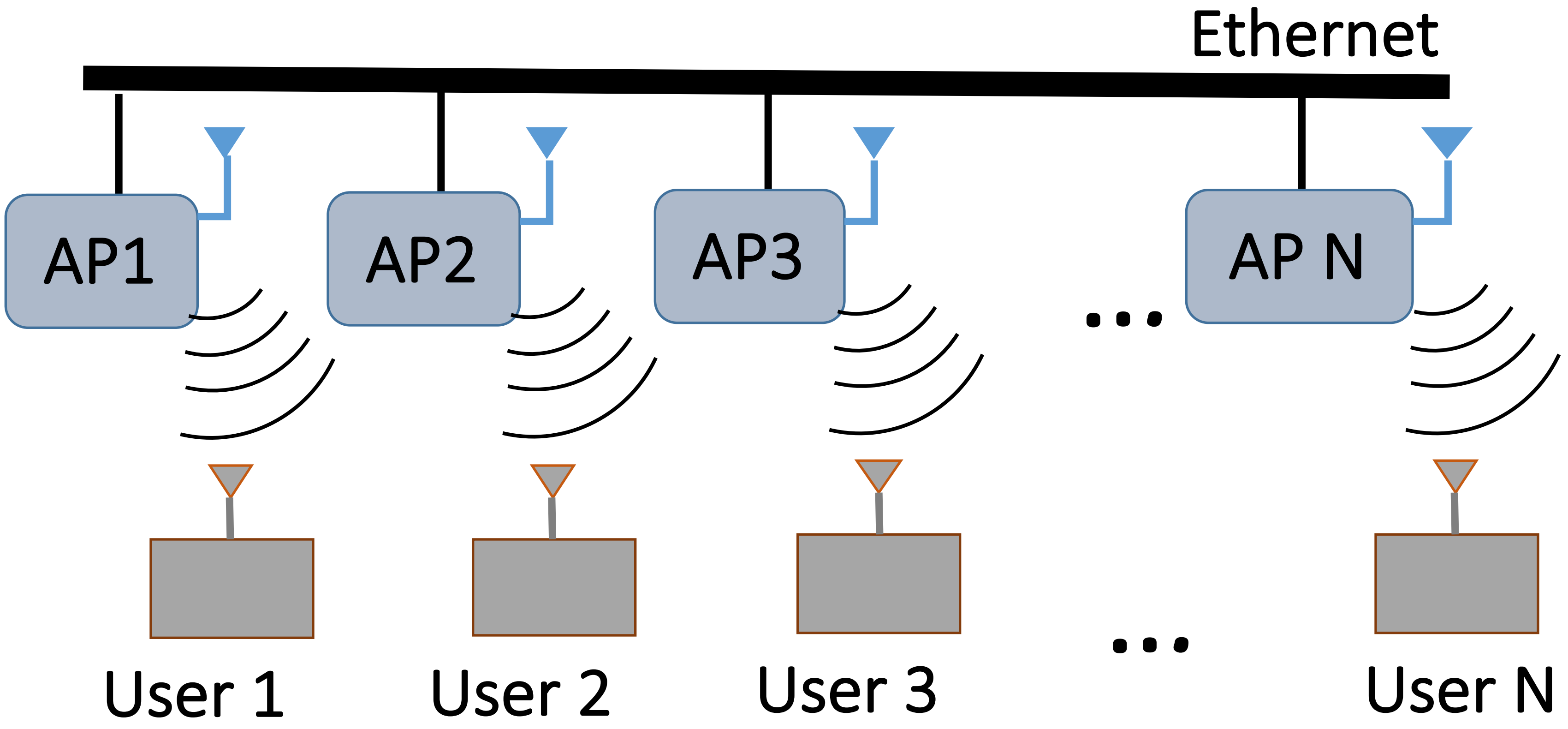
Ezzeldin Hamed, Mohammed A. Abdelghany, Dina Katabi

Dense Wireless Networks

- Stadiums
- Concerts
- Airports
- Malls

Interference Limits Wireless Throughput

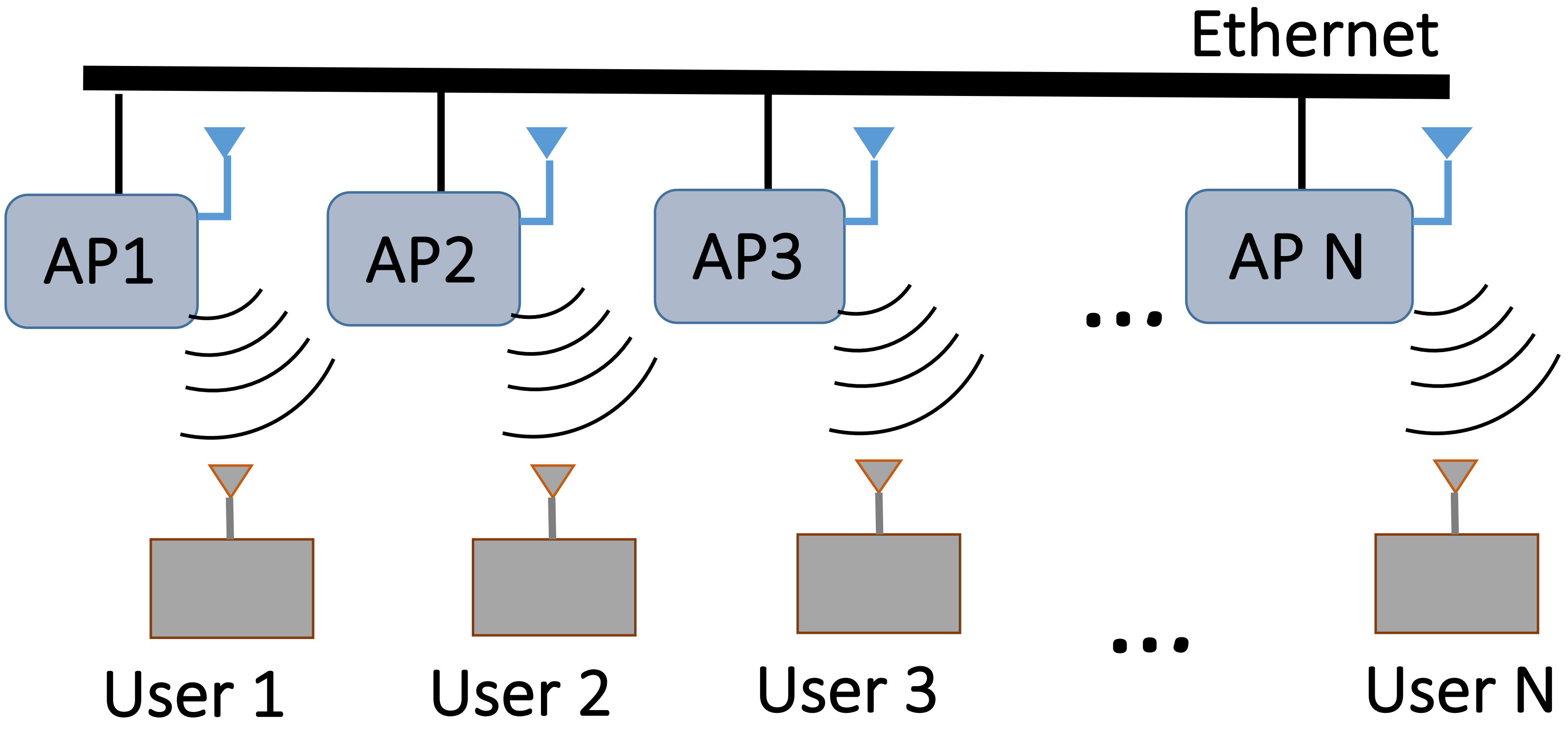
APs cannot transmit at the same time, in the same frequency
→ Take turns to avoid collisions



Total Wireless Throughput Stays Constant → Each AP gets 1/N of the total throughput

Distributed MIMO is the Holy Grail

Distributed protocol for APs to act as a huge MIMO transmitter with sum of antennas



N APs → N times higher throughput

Much recent work in moving distributed MIMO from theory to practice

However, we still do not have real-time distributed MIMO systems operating on independent devices with their own clocks!

Why aren't we there yet?

- Distributed MIMO needs accurate channel estimation → High overhead process that could eat up all the gains.

- Need distributed power control.

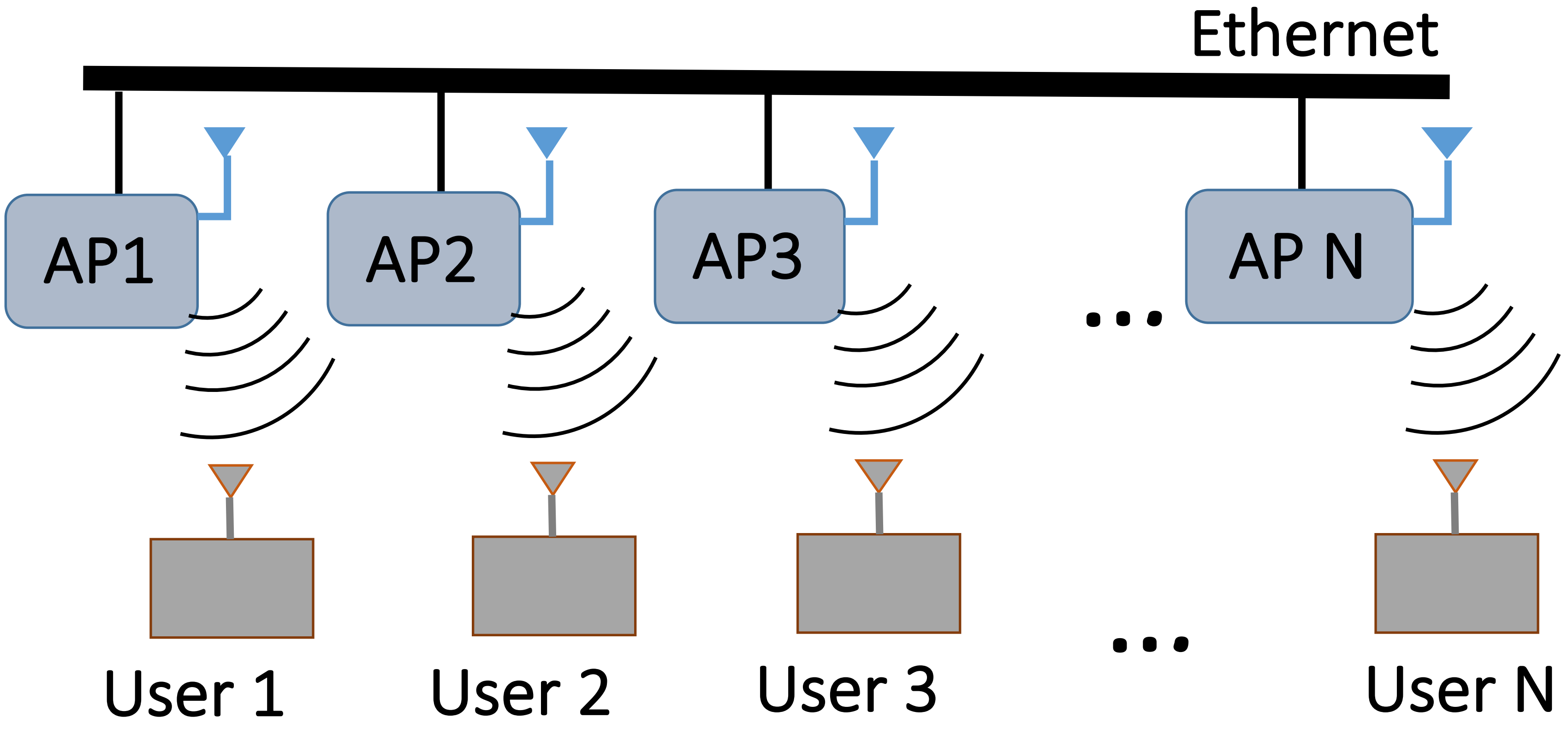
- Need an architecture that can support these complex operations in real-time.

Here the team controls 4 APs transmitting to 4 users

Why aren't we there yet?

- Distributed MIMO needs accurate channel estimation → High overhead process that could eat up all the gains.
- Need distributed power control.
- Need an architecture that can support these complex operations in real-time.

Channel Estimation and Feedback

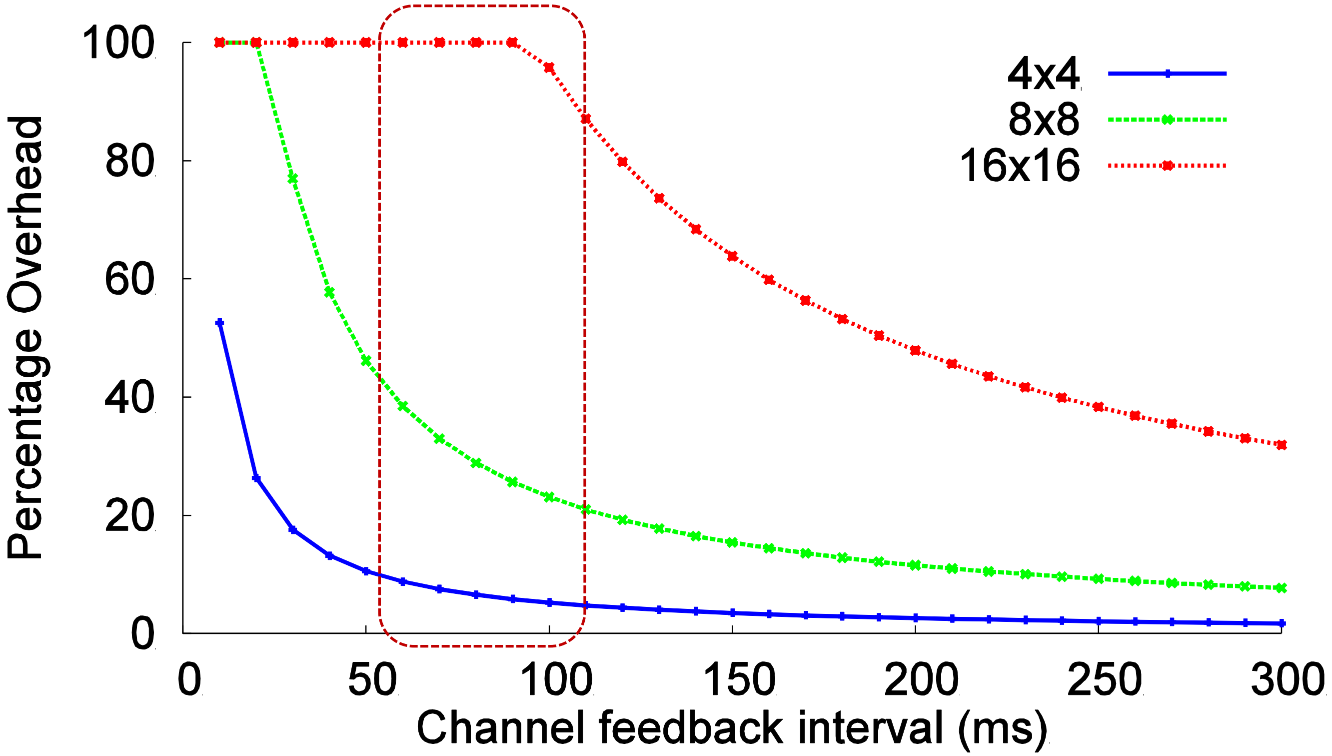


N Channel Estimation Packets

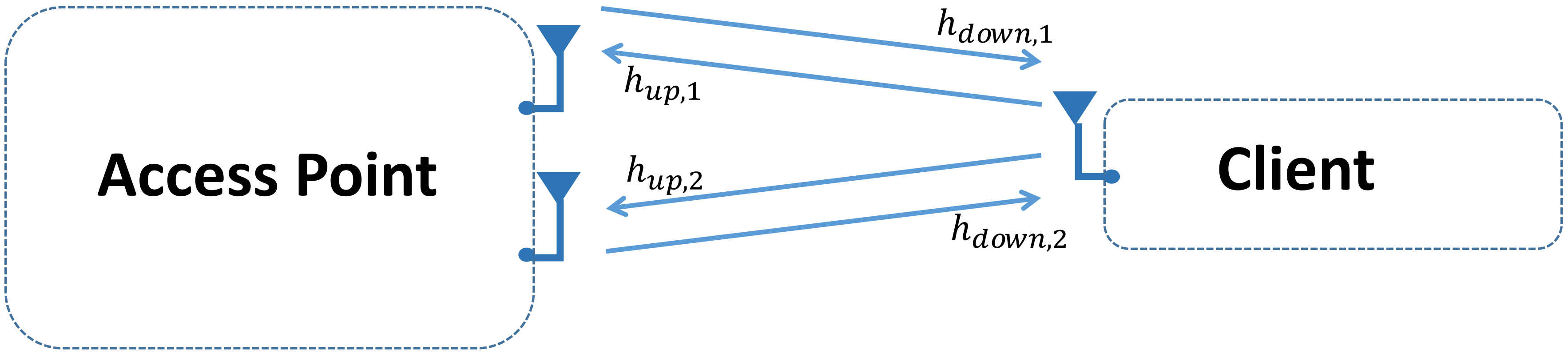
N^2 Channel Measurements

Need to do this periodically as environment changes

Channel Feedback Overhead

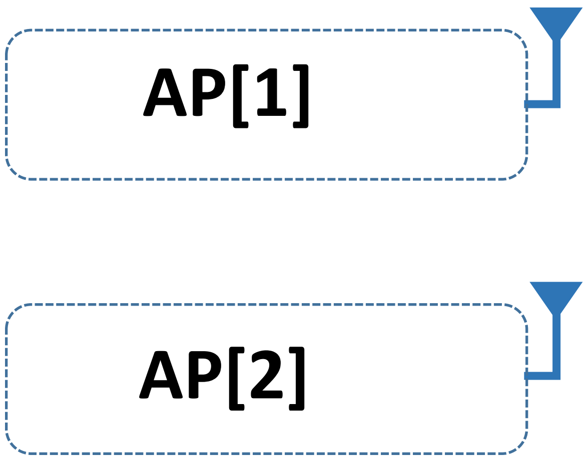


Reciprocity in Traditional MIMO



- Reciprocity is the property that the ratio of downlink channels is equal to the ratio of uplink channels up to a constant.
- This constant is the ratio between hardware chains of AP antennas.
- Allows us to estimate this constant once and use it for all future uplink transmissions and across clients.

What happens with Distributed MIMO?



Separate devices → Different Crystals

→ RF chains have oscillator offset relative to each other

Traditional Reciprocity does not work with Distributed MIMO

The "constant" is no longer constant, but changes rapidly with time.

Theorem:

The downlink and uplink channel ratios can be written as:

$$\frac{h_{down,2}}{h_{down,1}} = C_2(t) \times \frac{h_{up,2}}{h_{up,1}}$$

where $C_2(t) = C_2(0) \times e^{j2\Delta\omega_2 t}$

Reciprocity and Distributed MIMO Calibration

Calibration Parameter is rapidly time varying → Cannot do one-time calibration

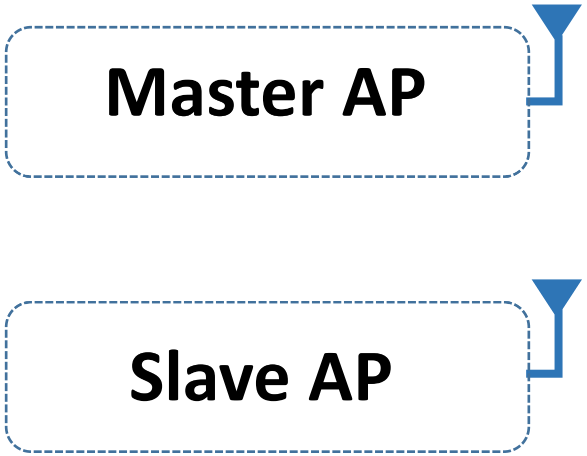
Need to repeatedly calibrate:

- for ***uplink transmissions from every client***
- at ***every AP***

MegaMIMO 2.0 Calibration for Reciprocity

- Avoids the overhead of repeated calibration
- Distributed mechanism for updating calibration parameters at slaves with no overhead

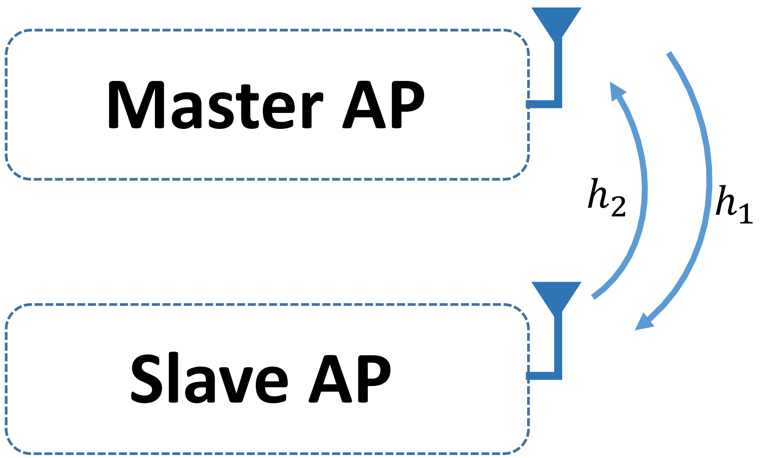
MegaMIMO 2.0 Calibration Formulation



$$C_2(t) = C_2(0) \times e^{j2\Delta\omega_2 t}$$

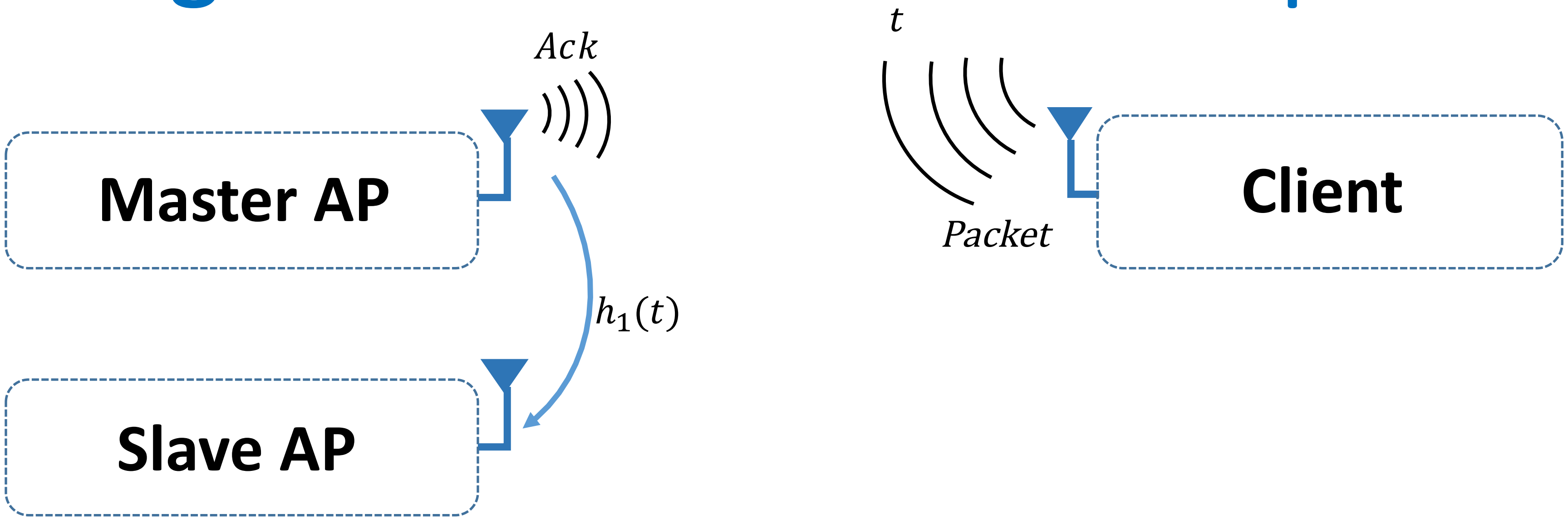
- Compute the initial calibration parameter, $C_2(0)$
- Update the calibration parameter at time t by estimating $e^{j2\Delta\omega_2 t}$

MegaMIMO 2.0 Initial Calibration



1. Measure channel h_1 from Master AP to Slave AP
2. Measure channel h_2 from Slave AP to Master AP
3. Compute Initial Calibration Parameter $C_2(0)$ as $C_2(0) = \dfrac{h_2}{h_1}$
4. At slave, store $C_2(0)$ and h_1 as $h_1(0)$

MegaMIMO 2.0 Calibration Update



1. Client transmits packet → Master and Slave measure uplink channels from client
2. Master sends sync trailer (Can leverage Wi-Fi ack)
3. Slave measures channel $h_1(t)$ from master. $h_1(t) = h_1(0) \times e^{j\Delta\omega_2 t}$
4. Recall that each slave has $h_1(0)$. Each slave computes $e^{j2\Delta\omega_2 t} = \left(\frac{h_1(t)}{}\right)^2$

Consistent channel estimates using reciprocity at all APs

MegaMIMO 2.0 Procedure

- **Preparing Calibration Constants**
  - Master AP transmits a reference packet
  - All slaves follow with a response
  - Each slave calculates its calibration parameter
- **Channel Estimation**
  - Performed for each uplink transmission from a client
  - The master AP follows with an ACK (Sync trailer)
  - Each slave calculates its downlink channel using the corrected calibration parameter
- **Joint Transmission**
  - The same as MegaMIMO 1.0

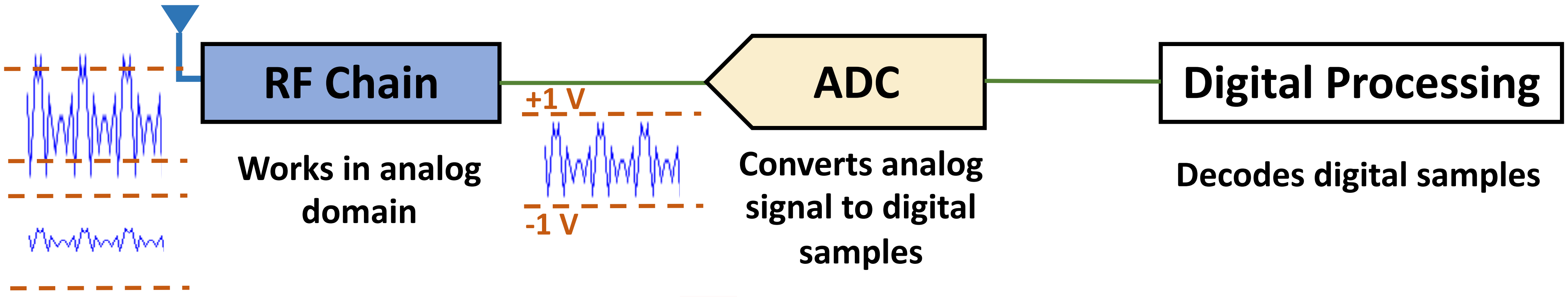
Why aren’t we there yet?

- Distributed MIMO needs accurate channel estimation → High overhead process that could eat up all the gains.

- Need distributed power control.

- Need an architecture that can support these complex operations in real-time.

The Need for Automatic Gain Control (AGC)



- ADC accepts signals in a specific range
- RF chain converts received signal to ADC range
- AGC is an adaptive algorithm to perform this conversion

AGC in Traditional MIMO

AP applies the same gain to all receive antennas

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

AGC in Traditional MIMO

AP applies the same gain to all receive antennas

$$\begin{bmatrix} \alpha\, h_{11} & \alpha\, h_{12} & \alpha\, h_{13} & \alpha\, h_{14} \\ \alpha\, h_{21} & \alpha\, h_{22} & \alpha\, h_{23} & \alpha\, h_{24} \\ \alpha\, h_{31} & \alpha\, h_{32} & \alpha\, h_{33} & \alpha\, h_{34} \\ \alpha\, h_{41} & \alpha\, h_{42} & \alpha\, h_{43} & \alpha\, h_{44} \end{bmatrix}$$

AGC in Distributed MIMO

Each AP-client link has an independent gain

$$\begin{bmatrix} \alpha_{11}h_{11} & \alpha_{12}h_{12} & \alpha_{13}h_{13} & \alpha_{14}h_{14} \\ \alpha_{21}h_{21} & \alpha_{22}h_{22} & \alpha_{23}h_{23} & \alpha_{24}h_{24} \\ \alpha_{31}h_{31} & \alpha_{32}h_{32} & \alpha_{33}h_{33} & \alpha_{34}h_{34} \\ \alpha_{41}h_{41} & \alpha_{42}h_{42} & \alpha_{43}h_{43} & \alpha_{44}h_{44} \end{bmatrix}$$

We need a protocol for ensuring that the multipliers are the same despite being applied on different boxes

Compensating for the AGC

- AGC typically has a coarse power setting → Need to convert to a complex α value.
- This conversion is not known *a priori*.
- MegaMIMO 2.0 learns this conversion factor.
  - Each antenna transmits a signal.
  - Receiver sets gain to a particular coarse value, and measures received channel
  - Repeats across all coarse gain settings
- Needs to be recalibrated infrequently to account for drift of analog components.

Why aren’t we there yet?

- Distributed MIMO needs accurate channel estimation → High overhead process that could eat up all the gains.
- Need distributed power control.
- Need an architecture that can support these complex operations in real-time.

MegaMIMO 2.0 PHY-MAC Architecture

- 802.11 PHY is a complex system: power adaptation, rate adaptation, encoding and decoding at various modulations and code rates *etc.*
- Traditional PHY layers only have local control and coordination with an on-board MAC.
- Distributed MIMO requires distributed control and coordination across multiple transmit and receive chains.
- We design an architecture that provides hooks to/from the PHY to enable this distributed control efficiently in hardware.

Performance

Implementation

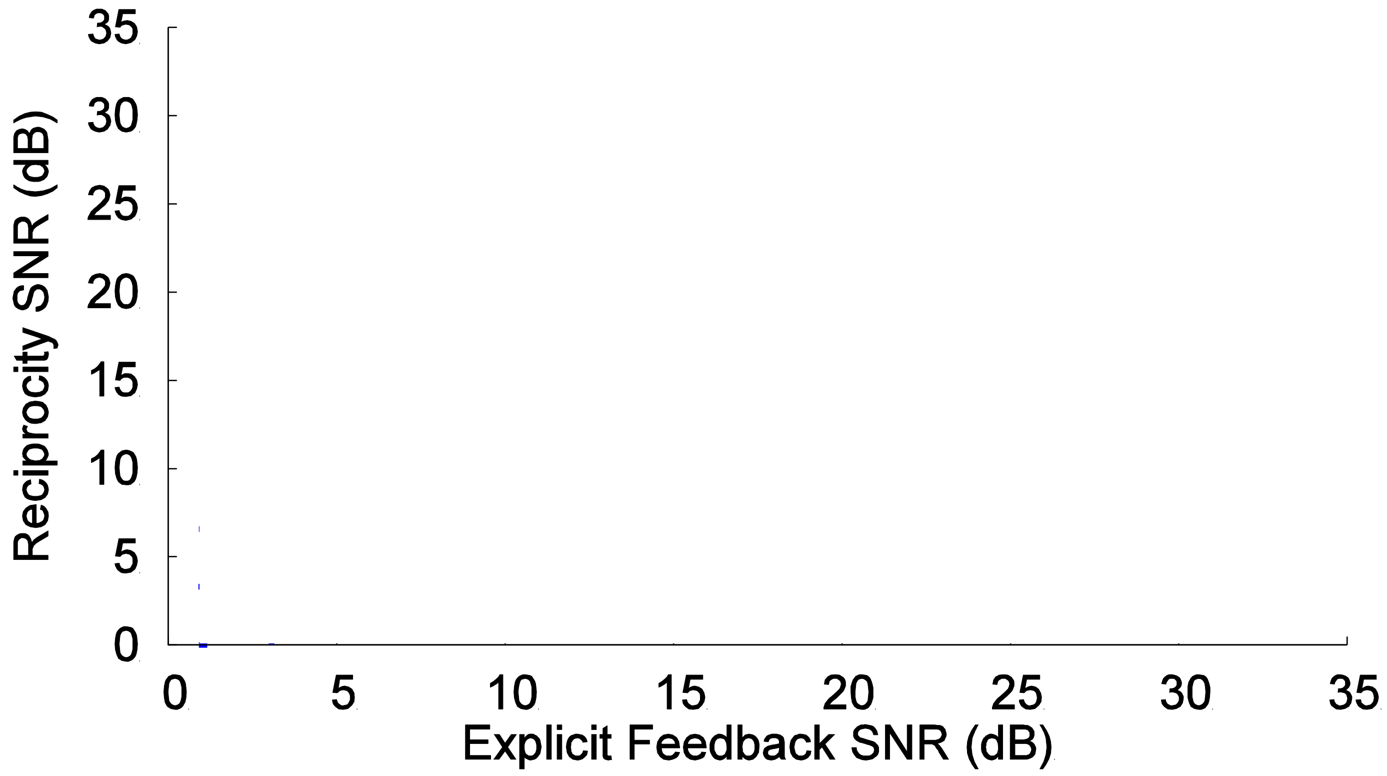
- Implemented on Zed Board and FMCOMMS2 RF Front End
- PHY and real time MAC implemented on Zynq FPGA
- Control Plane implemented on embedded ARM core

Evaluation

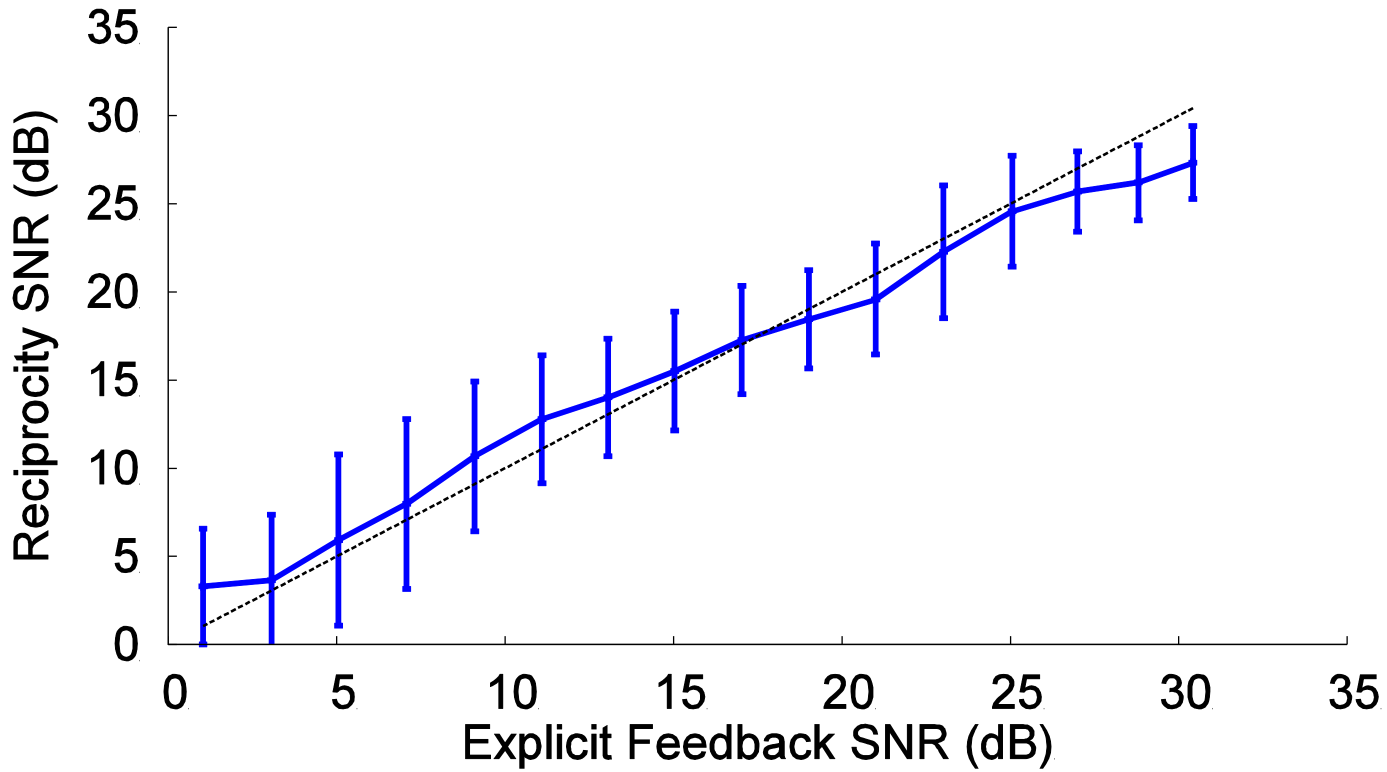
- Indoor Testbed simulating a conference room
- 4 APs transmitting to 4 clients
- Line of sight and non line of sight scenarios
- Mobility
  - Environment
  - Users
- Metrics
  - SNR obtained by users during joint transmission
  - Total throughput

Reciprocity vs. Feedback

Reciprocity vs. Feedback



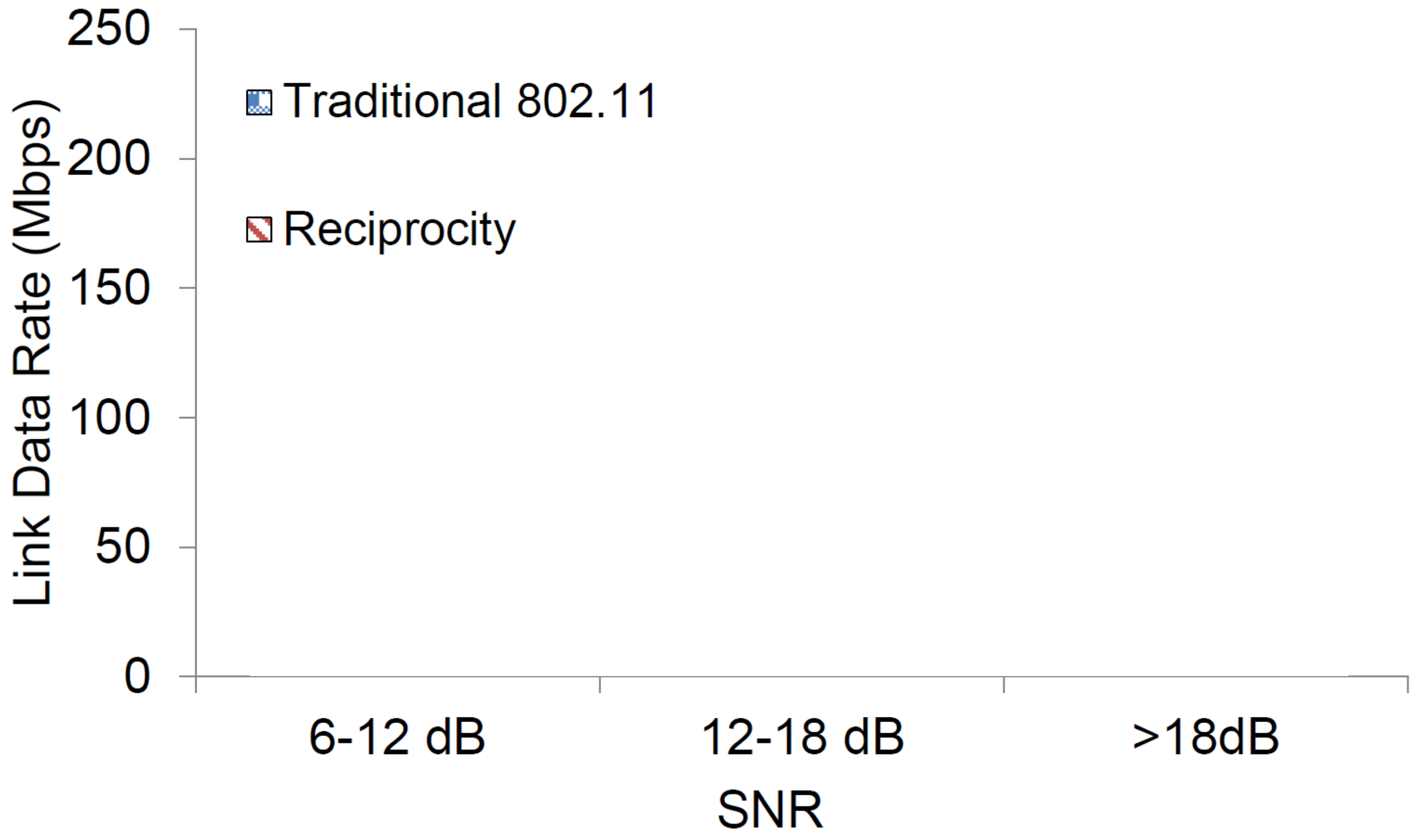
Reciprocity vs. Feedback



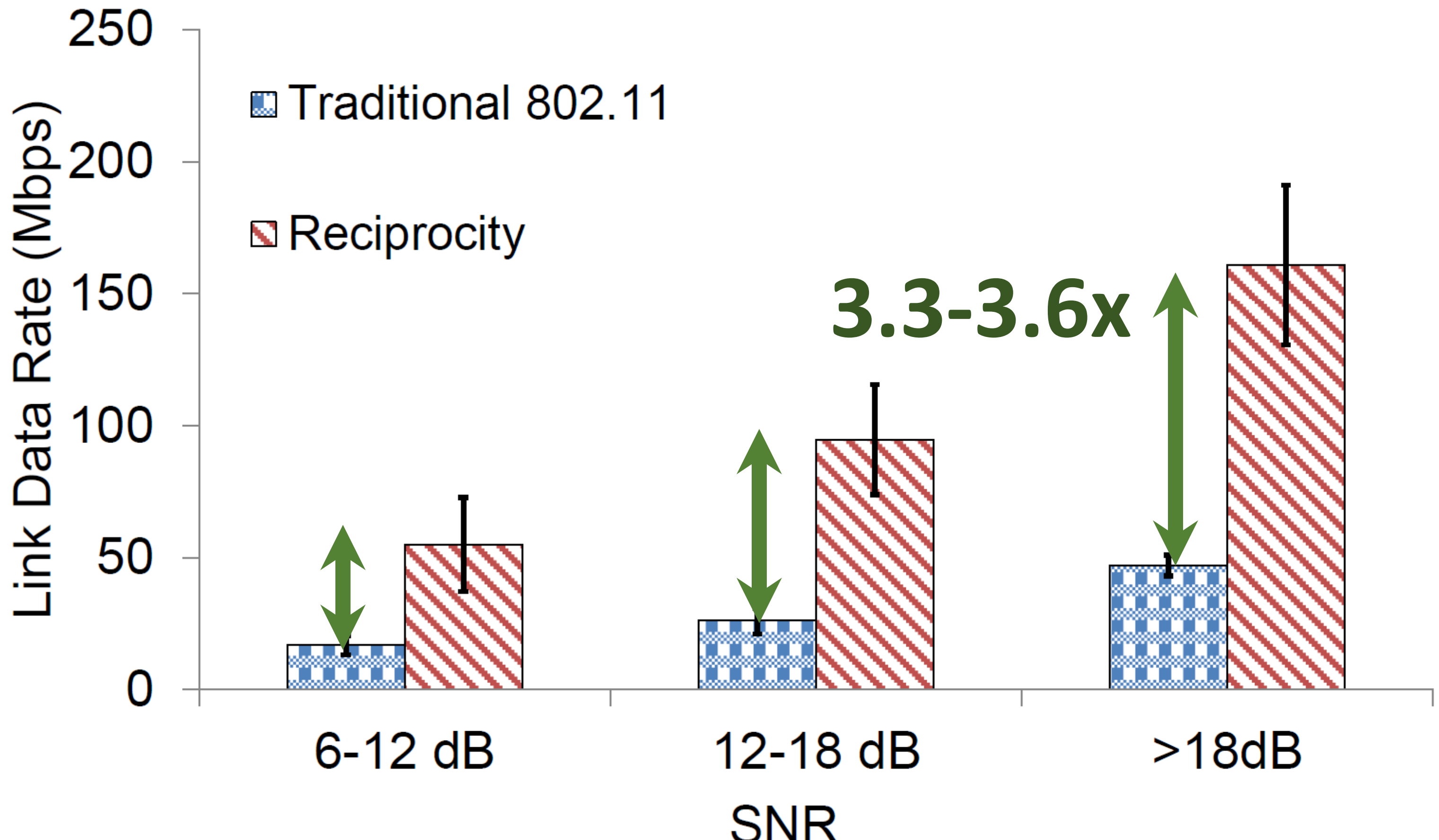
Reciprocity matches feedback across the range of SNRs → Calibration is accurate

MegaMIMO 2.0 vs. Traditional 802.11

MegaMIMO 2.0 vs. Traditional 802.11

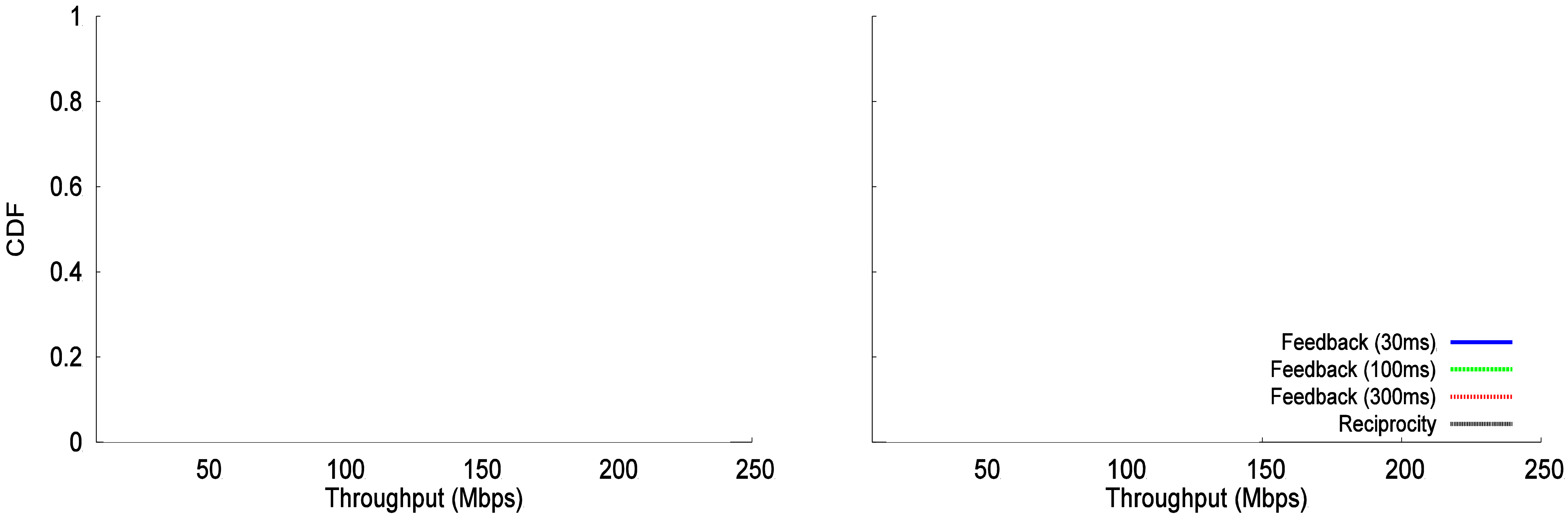


MegaMIMO 2.0 vs. Traditional 802.11



MegaMIMO 2.0 with reciprocity provides the expected scaling gains across the range of SNRs

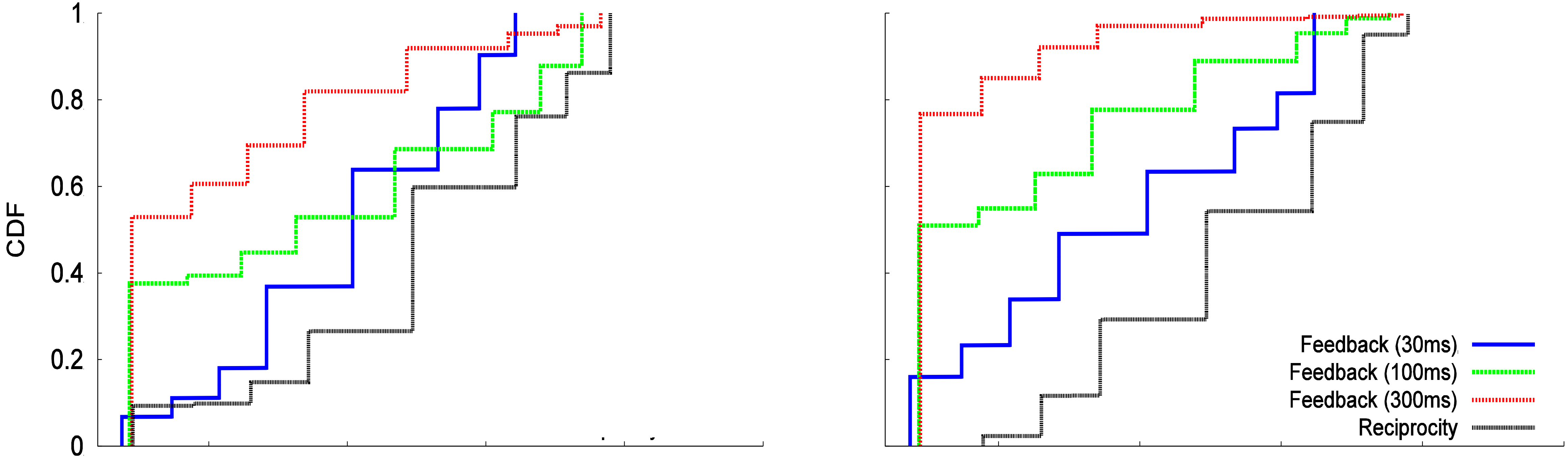
Reciprocity Throughput Gain with Mobility



Environmental Movement

Client Mobility

Reciprocity Throughput Gain with Mobility



No single feedback interval is optimal across all scenarios.

Reciprocity outperforms explicit feedback.

Conclusion

- MegaMIMO 2.0 is the first real-time distributed MIMO PHY layer operating across devices with independent clocks.
- Adapts to changing channel conditions in real-time with no overhead.
- Demonstrated with a hardware implementation.