UD-MIMO: Uplink Distributed MIMO for Wireless LANs

Hossein Pirayesh, Pedram Kheirkhah Sangdeh, Qiben Yan, and Huacheng Zeng
Department of Computer Science and Engineering, Michigan State University
Email: {pirayesh,sangdeh,qyan,hzeng}@msu.edu

***Abstract*—Wireless local area networks (WLANs) are a key component of the telecommunications infrastructure in our society. While many solutions have been produced to improve their downlink throughput, the techniques for enhancing their uplink throughput remain limited. The stagnation can be attributed to the lack of fine-grained inter-node synchronization due to the hardware limitation of most devices. In this paper, we present an uplink distributed multiple-input-and-multiple-output scheme (termed UD-MIMO) for WLANs to enable concurrent uplink transmission in the absence of fine-grained inter-node synchronization. The enabling technique behind UD-MIMO is a practical solution to decoding uplink packets from asynchronous users. UD-MIMO makes it possible for WLANs to significantly improve their uplink throughput while not requiring tight inter-node synchronization. We have built a prototype of UD-MIMO on a wireless testbed and demonstrate its compatibility with commercial off-the-shelf Atheros 802.11 client devices (with modified Linux driver). Our experimental results show that, for a WLAN with 8 APs in a conference room, UD-MIMO offers** $3.4\times$ **throughput compared to interference-avoidance approach.**

I. Introduction

The proliferation of wireless devices under the driving forces from emerging concepts such as smart cities, intelligent transportation systems, and the Internet of Things has led to unprecedented demands for wireless services. Cisco predicts that the wireless demands would double in the next two years and reach 120 exabytes per month by 2021 [1]. As a key component of the telecommunications infrastructure in our society, wireless local area networks (WLANs) carry even more data traffic for mobile devices than cellular networks. The predicament facing WLANs is that the increase of their capacity cannot catch up the growth of wireless demands. Such a predicament becomes particularly daunting in dense wireless environments such as conference rooms, football stadiums, cinemas, and airports.

A straightforward idea to increase the capacity of WLANs is to deploy more access points (APs) to enrich the service resources for users. This approach, however, does not work in dense wireless environments. The capacity of existing WLANs does not scale with the number of APs. This is because the existing WLANs use carrier sense multiple access (CSMA) protocol to manage the interference. Such an interference-avoidance protocol only allows one AP to access the spectrum in a collision domain, no matter how many APs are deployed in this area. Another idea to increase the capacity of WLANs is to enhance AP's capability [2]. Given the advancement of multiple-input-multiple-output (MIMO) technology in the past

978-1-6654-4108-7/21/$31.00 ©2021 IEEE

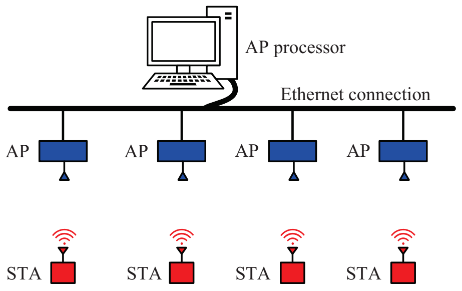


Fig. 1: Illustrating distributed MIMO in a WLAN.

decades [3], it is common nowadays that a commercial AP (Wi-Fi router) is equipped with multiple antennas. However, the advancement of individual AP cannot fundamentally solve the network capacity problem because the number of antennas on an AP is limited by its physical size.

Distributed MIMO has been widely regarded as a promising technique to improve the capacity of WLANs. Given the fact that APs are connected via high-speed Ethernet cables in some scenarios, all the APs can jointly process the signals from/to multiple users. With a proper design, the APs can serve many users simultaneously instead of being limited by their co-channel interference. Consider the WLAN in Fig. 1 for example. If the network uses CSMA-based interference-avoidance technique, only one AP can access the spectrum at a time. In contrast, if the APs are jointly processing the signals, then the WLAN resembles a 4×4 multi-user MIMO (MU-MIMO) system, making it possible for the APs to serve the four users simultaneously.

While the throughput gain of distributed MIMO is attractive, the realization of distributed MIMO in practical WLANs is challenging. The challenge lies in the clock/time synchronization among the network devices (AP and STAs). Despite the APs being connected via Ethernet cables, those connections are suitable for data packet transmission, but not suitable for clock/time synchronization. As such, synchronizing the network devices for distributed MIMO is not a trivial problem. Although some system papers have studied the synchronization issues to enable distributed MIMO for WLANs, most of them are focused on downlink transmission (see, e.g., [4, 5, 6, 7]). Very limited progress has been made so far in the design of a practical distributed MIMO scheme for concurrent uplink transmission. One may argue that most traffic in WLANs is carried by downlink, and thus the uplink capacity is not demanding. This may not be true in the next decades, given the increasing popularity of cloud-based applications that require frequent data transmission from user devices to cloud [8].

In this paper, we present UD-MIMO, an uplink distributed MIMO scheme for WLANs. We consider a WLAN as shown in Fig. 1 and focus on the scenario of busy wireless environments such as conference rooms, enterprises, hotels, shopping malls, and airports. We assume that multiple users send their packets to the APs simultaneously. Upon reception of the mixed signals, the APs send the received signals to an AP processor via Ethernet connection, which typically has high speed and low latency. At the AP processor, a signal detection technique is employed to decode the data packets. In such a network, if the devices are perfectly synchronized, then UD-MIMO would be identical to MU-MIMO, and conventional multi-user detection (MUD) methods such as zero-forcing (ZF) and minimum mean square error (MMSE) would be able to decode signals at the AP processor. But in reality, the network devices are driven by independent oscillators. Consequently, they are neither time-aligned nor frequency-synchronized, making the signal detection problem particularly challenging.

One natural approach to solving the signal detection problem is by designing a sophisticated protocol to synchronize the network devices. This approach, however, has two issues. First, the time synchronization among stations (STAs, a.k.a. user devices) is not easy to achieve. In order for the APs to decode the packets, the time misalignment of STAs' transmissions should be less than the cyclic prefix (CP) of an orthogonal frequency division multiplexing (OFDM) symbol, which is 800 ns in 802.11 networks. Given STAs' mobility, achieving such a fine-grained time synchronization among all the STAs will incur a large amount of airtime overhead. This issue was reflected by IEEE 802.11ac standard [9], which supports downlink MU-MIMO but does not supports uplink MU-MIMO. Second, the time and frequency synchronizations of STA-side transmissions require hardware modification of the user devices. Doing so will make UD-MIMO not compatible with already-existing 802.11 devices. For these two reasons, synchronizing the STAs for uplink transmissions is not a good approach to pursue.

We, therefore, explore an alternative approach: Instead of synchronizing the STAs, we live with their asynchrony and tackle the issue on the AP side. Specifically, we develop a new MUD method that can decode the asynchronous data packets from multiple STAs. Through sophisticated signal processing functions, the new MUD method can decode the data packet from each STA by treating the packets from other STAs as interference. As such, it does not require synchronization among the STAs. This new MUD method not only removes the need for hardware modification of user devices, it also eliminates the huge airtime overhead induced by synchronization protocols.

We have built a prototype of UD-MIMO and evaluated its performance on two wireless testbeds: (i) The APs are custom-built using USRP devices, and the STAs are commercial Atheros 802.11 dongles with modified drivers. (ii) Both APs and STAs are custom-built using USRP devices. Based on our experimental results, we have the following observations: (i) UD-MIMO is compatible with commercial off-the-shelf Atheros 802.11 devices (with modified Linux driver). (ii) For a WLAN with 8 APs deployed in a conference room, UD-MIMO offers 3.4$\times$ uplink throughput compared to CSMA-based interference-avoidance approach. Meanwhile, UD-MIMO achieves more than 82% throughput of MU-MIMO, where all the APs and STAs are perfectly synchronized via external clocks.

II. Related Work

Synchronization in Distributed MIMO: [4, 5, 6] are the most relevant papers to this work. In [4], a scheme called JMB (or MegaMIMO) was proposed to enable downlink distributed MIMO in WLANs. Its main efforts focus on realizing phase and time synchronizations among independent APs so that a joint beamforming technique can be used to enable downlink MU-MIMO transmission. A similar idea called Airsync was proposed in [5] to address timing and carrier phase synchronizations for distributed downlink MU-MIMO transmission. One may wonder if the schemes proposed in [4] and [5] can be used to enable UD-MIMO as well. Actually, it cannot. Because doing so will require hardware modification of 802.11 client devices. In contrast, UD-MIMO not only maintains compatibility with 802.11 devices but also reduces the overhead (see Fig. 2 in this paper and Fig. 3 in [4]).

In [6], a layering protocol called Chorus was proposed to achieve network-wide clock and time synchronization for LTE systems. However, Chorus relies on extra radio resource blocks and new hardware to update frequency shift and phase errors. It is therefore considered an expensive solution. Apparently, UD-MIMO takes a completely different approach.

Synchronization in Wireless Networks: A large body of work (see, e.g., [10, 11, 12, 13, 14, 15]) studied time and frequency synchronizations in wireless networks. For example, [10] proposed a distributed architecture called SourceSync to exploit the diversity of transmitters. Particularly, a specific protocol was proposed to meet the requirements of time synchronization on the transmitter side. Since SourceSync was dedicatedly devised for exploiting diversity, it cannot apply to distributed MIMO for spatial multiplexing. [11] analyzed the time and frequency synchronizations in large-sized dense wireless networks. However, these results cannot directly be applied to distributed MIMO systems, either because they are limited to theoretical analysis or because they entail an overwhelmingly large amount of overhead.

Performance of Distributed MIMO: [16] presented Signpost, a scalable MU-MIMO scheme without CSI feedback. [7] presented NEMOx, a hierarchical network architecture to achieve the scalability of distributed MIMO. [17] studied the performance of different precoding techniques in downlink distributed MIMO systems. However, these efforts focused on the practical realization of distributed MIMO but did not take into account the synchronization issues. Our work is orthogonal to this research line and complements these efforts.

III. UD-MIMO: An Uplink Distributed MIMO Scheme

We consider a dense WLAN as shown in Fig. 1, which comprises M single-antenna APs, N single-antenna STAs, and an AP processor. Such a network could be a Wi-Fi network deployed in conference rooms, shopping malls, or airports. For this network, we have the following assumptions: (i) The APs are connected via a high-speed wired connection, which is only good for exchange data packets but not suitable for clock synchronization. This is true in reality. (ii) The STAs in the network could be incumbent 802.11a/g/n/ac user devices. While their software (firmware and driver) can be upgraded, their hardware (e.g., PLL circuit and baseband signal processing at the PHY layer) cannot be upgraded.

The objective of our design is to enable concurrent uplink transmissions in such a WLAN while preserving its compatibility with incumbent 802.11 client devices (STAs). As the performance of conventional WLANs is limited by co-channel interference, the success of UD-MIMO will significantly improve the network uplink throughput. For ease of exposition, we consider the network where each device has a single antenna. In the end, we shall see that UD-MIMO can also apply to the networks where devices have multiple antennas.

A. Our Approach

Given that the difference between UD-MIMO and point-to-point MIMO lies in the synchronizations, an intuitive approach to enable UD-MIMO is by designing a sophisticated mechanism to achieve the necessary synchronizations on both AP and STA sides. This approach, however, cannot maintain backward compatibility with existing 802.11 devices (STAs). This is because synchronizing the STAs requires the modification of their hardware. The estimation and compensation of carrier frequency offsets can only be done through baseband signal processing modules, which are hard-coded in ASIC chips and cannot be modified by upgrading the firmware or driver. Hence, synchronization operations cannot be conducted by existing 802.11 devices, and such an approach cannot maintain the backward compatibility of the network infrastructure. We propose a new approach for UD-MIMO. In our approach, the APs take full responsibility for addressing the synchronization issues, and the STAs do not need to perform any synchronization operations.

B. UD-MIMO Protocol

Fig. 2 shows the protocol for UD-MIMO transmission. It comprises the following three steps:

- **Step 1: Trigger frame broadcast.** The lead AP, which is designated by the AP processor, broadcasts a trigger frame (packet). This packet serves the following two purposes. i) *Announcing UD-MIMO transmission:* The trigger packet includes the addresses of the slave APs and STAs that are involved in this UD-MIMO. Upon reception of this packet, the slave APs and the STAs are notified of their participation in the UD-MIMO transmission. ii) *Providing reference packet for the slave APs to estimate their carrier frequency offsets:* Based on the received trigger frame from the lead AP, each slave AP estimates the carrier frequency offset between itself and the lead AP. The estimated carrier frequency offset is recorded at the slave AP and will be used in Step 2.
- **Step 2: Uplink data transmission.** Upon reception of the trigger frame, the STAs prepare their data packets and send radio signals into the air simultaneously. More specifically, each STA uses an aggregate frame format in Fig. 3 for the uplink data transmission. Note that, since the STAs operate independently, their transmissions will not exactly start at the same time. A time misalignment may exist, as illustrated in Fig. 2. On the AP side, each AP receives mixed radio signals from the STAs. Each slave AP compensates the carrier frequency offset between itself and the lead AP using the frequency offset value estimated in Step 1. Then, all the APs send their signal streams to the AP processor.
- **Step 3: Acknowledgment (ACK).** Upon the decoding results, the lead AP broadcasts an ACK/NACK packet to the STAs. The ACK/NACK packet has the information of which packets from which STAs were not successfully decoded. Based on this information, each STA prepares a retransmission, if necessary, in the next round.

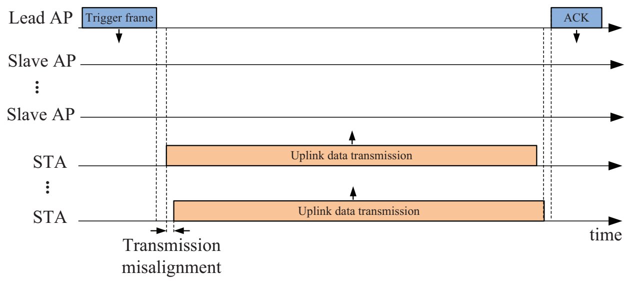


Fig. 2: Proposed protocol for UD-MIMO.

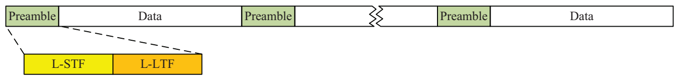
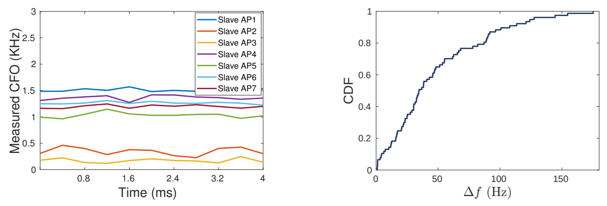


Fig. 3: An aggregate frame for uplink transmission.

A practical consideration is whether the trigger frame from the lead AP is sufficient for the slave APs to synchronize their carrier frequency in the time period of uplink data transmission. To address this concern, we study the stability of frequency synchronization among the lead and slave APs. Fig. 4 shows the measured carrier frequency offsets at seven slave APs and the residual carrier frequency offsets after the compensation of frequency offsets. It is evident that the residual carrier frequency offsets are less than 180 Hz in 4 ms. This accuracy is sufficient for concurrent data transmission.

It is evident that the proposed protocol is simple and has low airtime overhead. But a big question is yet to be answered: how can the AP processor decode the data packets from the STAs? We focus on this question in the next section.



(a) (b)

Fig. 4: Measured carrier frequency offsets at 7 slave APs with respect to a lead AP. (a) carrier frequency offsets over time; (b) distribution of residual carrier frequency offsets after frequency offset compensation.

IV. Packet Detection

At the AP processor, decoding the data packets faces the following two challenges. First, since the STAs are driven by independent clocks, their carrier frequencies are not exactly the same. Consequently, from the APs' perspective, the received signals from different STAs have different carrier frequency offsets, which must be compensated for signal detection. However, it remains unknown how to handle such heterogeneous carrier frequency offsets at a MIMO receiver. Second, since the STAs operate independently, their uplink data packets are unlikely to be aligned in the time domain. Moreover, for 802.11 devices, the misalignment of data packets is hardly confined within the duration of OFDM's CP (800 ns). Such a time misalignment makes it hard for the AP processor to decode the data packets.

A. Overview

To address the STA-side asynchrony issue, we propose a new packet detection method for the AP processor. This new detection method lives with the STA-side asynchrony and tackles the asynchrony issue through baseband signal processing on the AP side (i.e., at the AP processor). Fig. 5 shows the schematic diagram. The AP processor continuously receives the signal streams from the APs. From the signal streams, it extracts N signal frames, each of which corresponds to a packet from one STA. Then, it decodes each of the N signal frames separately, as illustrated in Fig. 5.

Consider one of the N signal frames, for example. Suppose that it corresponds to the data packet from STA i. We note that this signal frame includes not only the desired signal from STA i but also the undesired signals (interference) from other STAs. To decode this signal frame, the AP processor first performs carrier frequency correction. This module will estimate and compensate the carrier frequency offset between STA i and the APs (assuming the APs have been perfectly synchronized in Step 1 of our protocol). Then, the AP processor converts the signal to the frequency domain for signal detection. When performing signal detection, the AP processor treats the signals from other STAs (all the STAs except STA i) as unknown interference and constructs spatial filters to cancel the interference and equalize the channel distortion.

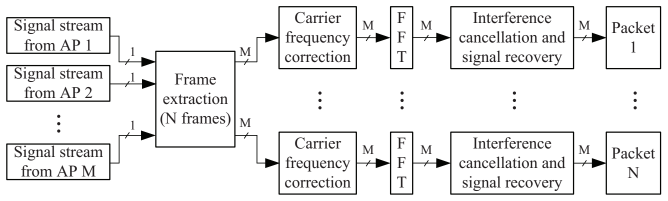

Signal stream from AP 1
Signal stream from AP 2
Signal stream from AP M
1
1
1
Frame extraction (N frames)
M
M
Carrier frequency correction
Carrier frequency correction
M
FFT
FFT
Interference cancellation and signal recovery
Interference cancellation and signal recovery
Packet 1
Packet N

Fig. 5: The schematic diagram of our packet detection method.

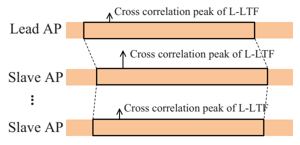

Lead AP
Slave AP
Slave AP
Cross correlation peak of L-LTF
Cross correlation peak of L-LTF
Cross correlation peak of L-LTF

Fig. 6: Extracting the signal frame for one STA at the AP processor via correlating signal streams with L-LTF.

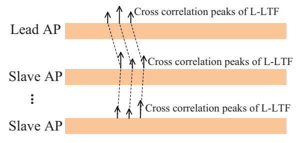

Lead AP
Slave AP
Slave AP
Cross correlation peaks of L-LTF
Cross correlation peaks of L-LTF
Cross correlation peaks of L-LTF

Fig. 7: Extracting the signal frames for three STAs at the AP processor via correlating signal streams with L-LTF.

B. Frame Extraction

To extract signal frames from the signal streams, the AP processor employs cross-correlation. Specifically, the AP processor correlates each signal stream with a local copy of L-LTF. If the normalized correlation value is greater than a predefined correlation threshold (e.g., 0.4), then it is regarded as the start of a signal frame. Fig. 6 illustrates the extraction procedure when the network has one STA, and Fig. 7 illustrates the correlation peaks when the network has three STAs. When there are multiple correlation peaks, their corresponding signal frames are matched in order at each AP (see Fig. 7 as an example).

It is worth pointing out that the cross-correlation is conducted in the presence of interference and noise. The correlation value is therefore dependent on the strength of interference and the noise power. Hence, the correlation threshold should be meticulously chosen. A small threshold may lead to a false positive, and a large threshold may lead to a false negative. In our experiments, we set the threshold to 0.4 and find that it works well for the following two reasons: (i) Cross-correlation itself is resilient to interference. L-LTF has 160 samples and appears to be robust against interference. (ii) False negatives are acceptable for packet detection. A small correlation value (below the threshold) indicates that the desired signal is weak and the interference is strong. The exclusion of this stream will actually improve the performance of packet detection, provided that the spatial DoF is sufficient for packet detection.

C. Carrier Frequency Correction

Referring to Fig. 5, let us consider one of the N signal frames. Suppose it corresponds to the data packet from STA i. To decode this signal frame, the AP processor first performs carrier frequency correction. In the presence of inter-user (inter-STA) interference, conventional methods do not work because they are susceptible to interference. To tackle this issue, we propose a new method, which comprises two steps:

(i) signal projection, and (ii) CP-based correlation. Denote $\mathbf{y}(n) \in \mathbb{C}^{M\times 1}$, $1 \leq n \leq N_s$, as the time-domain signal frame, where N_s is the number of samples in a frame. We first compute the eigenvectors as follows:

$$[\mathbf{u} \;\; \mathbf{d}] = \mathrm{eig}\big(\sum_{n=1}^{N_s} \mathbf{y}(n)\mathbf{y}(n)^{\mathsf{H}}\big), \tag{1}$$

where $\mathrm{eig}(\cdot)$ is eigendecomposition operator, $(\cdot)^{\mathsf{H}}$ is the Hermitian transpose operator, $\mathbf{u} \in \mathbb{C}^{M\times M}$ is the eigenvectors, and $\mathbf{d} \in \mathbb{C}^{M\times M}$ is diagonal matrix of eigenvalues. Then, we project the signals into the eigenvector space by letting $\tilde{\mathbf{y}}(n) = \mathbf{u}^{\mathsf{H}}\mathbf{y}(n)$, where $\tilde{\mathbf{y}}(n) \in \mathbb{C}^{M\times 1}$ and $1 \leq n \leq N_s$. Each row of $\tilde{\mathbf{y}}(n)$ can be used to estimate the carrier frequency offset, and we should pick up the best one (the one with highest signal-to-interference ratio). To do so, we perform the cross correlation again on each row of $\tilde{\mathbf{y}}(n)$ and choose the one with the maximum cross correlation value for carrier frequency offset estimation. Denote $\tilde{y}_m(n)$ as the row of $\tilde{\mathbf{y}}(n)$ that has the maximum cross correlation value. Denote $\hat{\theta}_i$ as the estimated phase offset per sample between the APs and STA i. Then we have $\hat{\theta}_i = (1/64)\arg\big(\sum_{n\in\mathcal{C}} \tilde{y}_m(n)\tilde{y}_m(n+64)^*\big)$, where $\arg(\cdot)$ is the angle of a complex number, $(\cdot)^*$ is complex conjugate operator, $\mathcal{C}$ is the set of samples in the CP of all OFDM symbols, and 64 is the distance between CP and its original copy in OFDM symbol.

After obtaining $\hat{\theta}_i$, we then compensate the carrier frequency offset by letting $\bar{\mathbf{y}}(n) = \mathbf{y}(n)\cdot e^{jn\hat{\theta}_i}$, $1 \leq n \leq N_s$. The resultant signal frame $\bar{\mathbf{y}}(n)$ is then sent to the FFT module, as shown in Fig. 5.

D. Interference Cancellation and Signal Recovery

Problem Formulation: After correcting the carrier frequency offset, the FFT module in Fig. 5 converts the signal frame from the time domain to the frequency domain. We let $Y_j(l,k)$ denote the output signals from the FFT module, where $j \in \{1,2,\cdots,M\}$ is the index of received signal streams (APs), $l \in \{1,2,\cdots,L\}$ is the index of OFDM symbols, and $k \in \{1,2,\cdots,K\}$ is the index of OFDM subcarriers. Assume that the signals in a frame experience block channel fading. Then, the signal transfer function can be written as:

$$Y_j(l,k) = \underbrace{H_{ji}(k)X_i(l,k)}_{\text{desired signal}} + \underbrace{\sum_{i'\in\mathcal{N}, i'\neq i} H_{ji'}(k)\bar{X}_{i'}(l,k)}_{\text{unknown interference}} + \underbrace{W_j(l,k)}_{\text{noise}}, \tag{2}$$

where $H_{ji}(k)$ is the channel between AP j and STA i, $X_i(l,k)$ is the original signal from STA i, $\bar{X}_{i'}(l,k)$ is the interfering signal from STA i', and $\mathcal{N}$ is the set of STAs.

For the transfer function in (2), we have the following three remarks. First, this transfer function requires carrier frequency synchronization between STA i and the APs. But it does not require phase synchronization between STAs and APs. Actually, the phase offset between STA i and AP j is considered as a part of $H_{ji}(k)$. Second, $X_i(l,k)$ in (2) is the original signal transmitted by STA i. But $\bar{X}_{i'}(l,k)$ is not the original signal transmitted by STA $i' \in \mathcal{N}/\{i\}$. This is because $\bar{X}_{i'}(l,k)$ is completely distorted by the carrier frequency offset and time misalignment between STA i' and the APs. It is considered unknown interference in this transfer function. Third, since $\bar{X}_{i'}(l,k)$ is an unknown interfering signal, it is hard to estimate the channel $H_{ji'}(k)$. This is the core challenge in signal recovery.

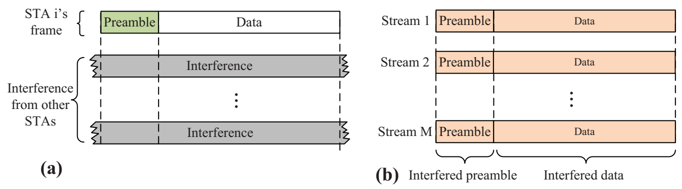


Fig. 8: (a) Transmitted signal and interference at STAs. (b) Received signal and interference at the APs.

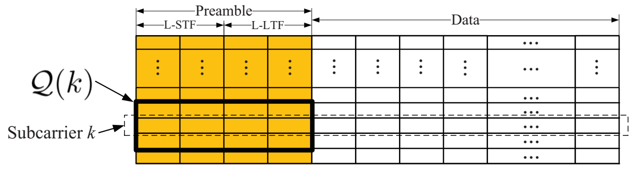


Fig. 9: Illustrating set $\mathcal{Q}(k)$ for filter construction.

Our Detection Method: Based on (2), if we know all the channels, then we can construct a spatial filter $\mathbf{G}(k) = [G_1(k), G_2(k), \cdots, G_M(k)]$ so that $\sum_{j=1}^{M} G_j(k)H_{ji}(k) = 1$ and $\sum_{j=1}^{M} G_j(k)H_{ji'}(k) = 0$, $i' \in \mathcal{N}/\{i\}$. Such a spatial filter can cancel the interference and recover the desired signal. This method is actually the well-known zero-forcing MIMO detector. By taking into account the effect of noise, the zero-forcing detector can be elevated to an MMSE detector.

Now the question is how to construct the spatial filter $\mathbf{G}(k)$ in the absence of channel knowledge. To address this question, we propose a training-based method. Consider the signal frame transmission from STA i to the APs. As illustrated in Fig. 8(a), we assume that (i) STA i's preamble is interfered by unknown signals from other STAs; and (ii) STA i's preamble is independent of its interference. Then, we focus on the received signal frame at the AP processor, which is illustrated in Fig. 8(b). The signal frame is composed of two parts: *interfered preamble* and *interfered data*. Since the preamble is known at the AP processor *a priori*, we use the interfered preamble in Fig. 8(b) as the training sequence to construct the spatial filter for data detection. Specifically, we construct the spatial filter as follows:

$$\mathbf{G}(k) = \Big[\sum_{(l,k')\in\mathcal{Q}(k)} \mathbf{Y}(l,k')\mathbf{Y}(l,k')^{\mathsf{H}}\Big]^{+}\Big[\sum_{(l,k')\in\mathcal{Q}(k)} \mathbf{Y}(l,k')X_i(l,k')^{\mathsf{H}}\Big], \tag{3}$$

where $[\cdot]^{+}$ is Moore–Penrose inverse (pseudo-inverse). $\mathbf{Y}(l,k) \in \mathbb{C}^{M\times 1}$ is the frequency-domain signal vector, i.e., $\mathbf{Y}(l,k) = [Y_1(l,k), Y_2(l,k), \cdots, Y_M(l,k)]^{\mathsf{T}}$. $\mathcal{Q}(k)$ is a set of reference symbols in the preamble. $\mathcal{Q}(k)$ can be empirically set. In our experiments, we let $\mathcal{Q}(k) = \{1\leq l \leq 4, k-1\leq k' \leq k+1\}$, as shown in Fig. 9.

After constructing the spatial filter, we then use it to estimate

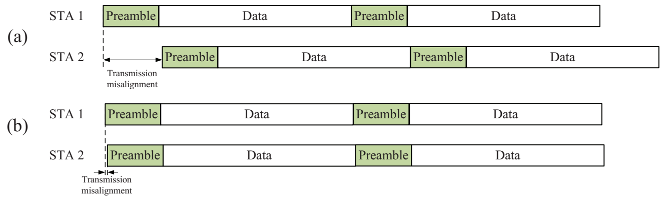


Fig. 10: (a) Transmission misalignment is greater than the time duration of preamble. (b) Transmission misalignment is less than the time duration of CP (guard interval).

the original signal by:

$$\hat{X}_i(l,k) = \sum_{j=1}^{M} G_j(k)^* Y_j(l,k), \qquad (4)$$

where $\hat{X}_i(l,k)$ is the estimated signal from STA i and $G_j(k)$ is the jth entry in vector (filter) $\mathbf{G}(k)$.

Discussions: We have the following two remarks on the proposed packet detection method. First, the proposed method does not require channel knowledge to decode the packet, as evidenced by (3) and (4). Instead, it uses the interfered preamble as the training sequence to construct a filter, which is then used to cancel the interference and equalize the channel distortion for signal recovery. Second, the proposed detection method is a heuristic. We will resort to experiments to evaluate its performance. As we will see in Section VI-A, this detection method yields surprisingly superior performance. With this detection method, the performance of UD-MIMO is close to that of MU-MIMO in all tested scenarios.

V. Compatibility with 802.11 Client Devices

In this section, we first point out the practical issues when UD-MIMO works with incumbent off-the-shelf Wi-Fi client devices and then propose a solution to these issues.

Practical Issues: UD-MIMO heavily relies on the new packet detection method to tame the asynchrony among the STAs. However, the packet detection method was proposed under the following two assumptions: (i) Referring to Fig. 8(a), STA i's preamble is interfered by the signals from other STAs. (ii) Referring to Fig. 8(a) again, STA i's preamble is linearly independent of the interfering signals from other STAs. To see why these two assumptions are mandatory, let us consider the examples in Fig. 10.

In Fig. 10(a), the transmission misalignment of the two STAs is greater than the time duration of the preamble. In this case, STA 1's first frame cannot be decoded at the AP processor. This is because its preamble is not interfered by the signal from STA 2. As a result, the spatial filter constructed based on this preamble cannot cancel the interference from STA 2. For STA 1's second frame, it can be decoded at the AP processor because its preamble is interfered by the signal from STA 2. In contrast, for STA 2's two frames, both of them can be decoded at the AP processor because their preambles are interfered by the signal from STA 1.

In Fig. 10(b), the transmission misalignment of the two STAs is less than the time duration of CP (guard interval).

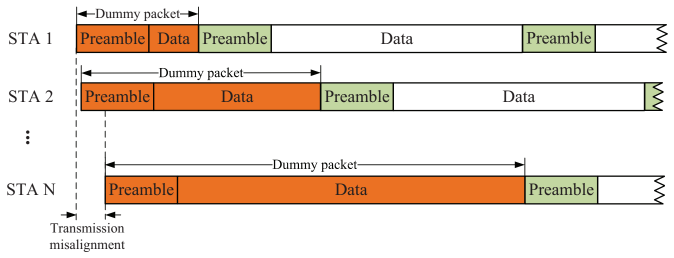


Fig. 11: Insert a different-length dummy packet at each STA to enable UD-MIMO transmission.

In this case, all of the four frames (two for STA 1 and two for STA 2) cannot be decoded at the AP processor. This is because their preambles are interfered by the same interference. From the AP processor's perspective, it has no way to differentiate the signal from STA 1 and that from STA 2. The spatial filter constructed based on the interfered preamble can neither cancel interference nor equalize the channel distortion. Therefore, all four frames cannot be decoded.

Such cases are likely to occur in practice. In Wi-Fi networks, the preamble is 16 microseconds, and the CP is 0.8 microseconds for long guard interval option and 0.4 microseconds for short guard interval option. Suppose that the synchronization error achieved by the timing synchronization function (TSF) specified in IEEE 802.11 is uniformly distributed in [0, 20] microseconds [18]. Then, the probability of case 1 is about 20%, and the probability of case 2 is 4% (or 2% for short guard interval option). Collectively, the probability of these two cases is 24%. Therefore, properly handling these two cases is imperative towards the real application of UD-MIMO.

Our Solution: To fulfill those two assumptions in the presence of STAs' transmission misalignment, our solution is simple. We insert a dummy packet at the beginning of uplink transmission at each STA, as illustrated in Fig. 11. By properly setting the length of the dummy packet, the preamble of each data packet will meet those two requirements. To determine the length of the dummy packet at each STA, let us again assume that the transmission misalignment is within 20 microseconds [18]. Then, we set the length of the dummy packet at STA i to $7i$ OFDM symbols ($1 \le i \le N$), including both preamble and data.

For the proposed solution, three remarks are in order.

Remark 1: To fulfill those two assumptions, the preambles from different STAs can partially overlap with each other. For example, the L-STF from one STA can overlap with the L-STF from another STA. In such a case, the AP processor can still decode the packets. Taking this fact into consideration may help reduce the length of dummy packets at the STAs.

Remark 2: Apparently, the proposed solution entails additional airtime overhead to enable UD-MIMO transmission. Further, the overhead slightly increases with the number of STAs. This issue can be alleviated by aggregating more packets (signal frames) in the uplink transmission. Since the channel coherence time is long enough in WLANs, an aggregate frame can accommodate hundreds of OFDM symbols. Then, the amortized overhead is acceptable in practice.

Remark 3: Since the dummy packet is a normal packet, no hardware modification is needed to insert the dummy packet for a commercial Wi-Fi client device. Rather, it can be implemented through modifying a Wi-Fi device's driver. The length of the dummy packet can be specified in the trigger frame by the lead AP. On the AP side, the AP processor will automatically drop the dummy packet, either because it cannot be decoded or it does not have necessary MAC information. In either case, the dummy packet will not affect the upper-layer applications.

VI. Experimental Evaluation

In this section, we conduct experiments to evaluate the performance of UD-MIMO on the two wireless testbeds.

A. Implementation and Experimental Setup

We have built two testbeds to evaluate the performance of UD-MIMO in real wireless environments.

802.11 Wi-Fi Dongle Testbed: The purpose of this testbed is to validate the practicality of UD-MIMO as well as its compatibility with commercial off-the-shelf Wi-Fi devices.

For the STAs, we use Wi-Fi dongles (Alfa AWUS036NHA Wireless USB Adapters), which are built on Qualcomm Atheros AR9271 chipset [19] and support IEEE 802.11b/g/n. We modify its firmware (`modwifi-ath9k-htc` in [20]) to disable carrier sense, RTS/CTS, ACK, set SIFS/AIFS to zero, and insert a dummy packet for UD-MIMO. For simplicity, we fix the MCS index to 2, which corresponds to QPSK modulation, 3/4 coding rate, and 18 Mbps data rate. While we use this specific modulation and coding scheme (MCS), UD-MIMO works with other MCS as well. We set channel bandwidth to 20 MHz and guard interval (OFDM CP) to 800 ns. The transmit power is fixed to 17 dBm, and the carrier frequency is set to 2.427 GHz (channel 4).

For the APs, we implement them using a set of USRP N210 devices [21]. Each USRP N210 device is connected to a D-Link SmartPro Switch via 1Gbps Ethernet RJ45 Cord, and the switch is connected to a computer via 10Gbps SFP+ DAC Cable. A software suite is developed using C++ and deployed at the computer to implement the protocol (see Fig. 2) and process the baseband signals. The output of our software suite is estimated signals from the STAs. Post-processing modules (e.g., deinterleaving, channel decoding, descrambling, and decryption) are not implemented.

USRP Testbed: The purpose of this testbed is to quantify the performance gap between UD-MIMO and MU-MIMO in the same scenarios. In this testbed, both APs and STAs are custom-built using USRP N210 devices. As such, we have full control for both APs and STAs. To measure the performance of MU-MIMO, we synchronize both APs and STAs using external clocks (Ettus' Octoclock CDA-2990G [22]). For ease of experimentation, we set the sampling rate to 5 Msps. Other parameters are the same as the 802.11 testbed.

Experimental Setup: Fig. 12 shows our experimental setup in a large conference room, where 8 APs and N STAs are deployed. The 8 APs are placed at the locations marked by solid boxes, and one of them is selected as lead AP. The 8 APs are connected to a computer via a 1/10 Gbps Ethernet switch. A set of spots (small circles in the figure) are marked out for the possible locations for the N STAs.

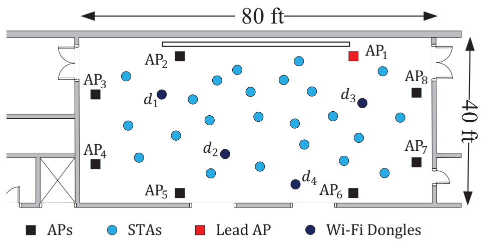


Fig. 12: A conference room for UD-MIMO evaluation.

TABLE I: EVM specification in IEEE 802.11ac standards [9].

EVM (dB)	(inf -5)	[-5 -10)	[-10 -13)	[-13 -16)	[-16 -19)	[-19 -22)	[-22 -25)	[-25 -27)	[-27 -30)	[-30 -32)	[-32 -inf)
Modulation	N/A	BPSK	QPSK	QPSK	16QAM	16QAM	64QAM	64QAM	64QAM	256QAM	256QAM
Coding rate	N/A	1/2	1/2	3/4	1/2	3/4	2/3	3/4	5/6	3/4	5/6
γ(EVM)	0	0.5	1	1.5	2	3	4	4.5	5	6	20/3

B. Comparison Baseline and Performance Metrics

Comparison Baseline: We use MU-MIMO as the comparison baseline to evaluate the performance of UD-MIMO. MU-MIMO serves as an upper bound for UD-MIMO. We quantify the performance gap between UD-MIMO and MU-MIMO. In MU-MIMO, all APs are synchronized via external clocks, and all STAs are synchronized via external clocks.

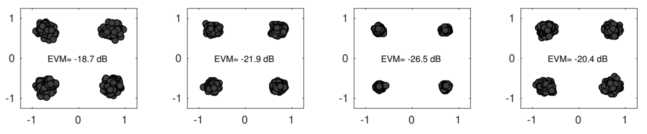
Performance Metrics: We consider two metrics. The first one is error vector magnitude (EVM), which is defined by: EVM (dB) $= 10\log_{10}\left(\frac{\mathbb{E}(|X-\hat{X}|^2)}{\mathbb{E}(|X|^2)}\right)$, where X is the original signal at STA and $\hat{X}$ is the estimated signal at AP. The second one is data rate, which is calculated by $r = \frac{48}{80} \times b \times \gamma(\text{EVM})$ Mbps, where 48 is the number of subcarriers used for payload in an OFDM symbol, 80 is the length of one OFDM symbol (including CP), b is the signal sampling rate (in Msps), and $\gamma(\text{EVM})$ is the average number of bits carried by one symbol and its values are given in Table I. In our experiments, we use $b = 20$ for the 802.11 testbed and $b = 5$ for the USRP testbed.

C. 802.11 Wi-Fi Dongle Testbed

On this testbed, we measure the uplink data rate per STA in two schemes: CSMA (interference avoidance) and UD-MIMO.

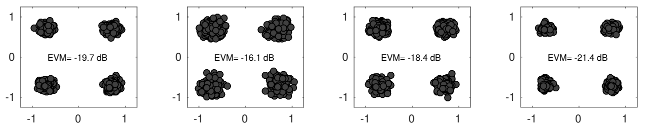
CSMA: In conventional Wi-Fi networks, only one Wi-Fi dongle can be allowed to communicate with one AP in a time slot. As such, we consider four dongles in four different time slots. In the ith time slot, dongle i placed at location d_i sends data packet to the lead AP, $1 \leq i \leq 4$.

Since each time slot has only one active dongle, there is no interference in this case. Fig. 13 plots the demodulated signal at the lead AP in the four time slots. Specifically, the measured EVMs for the four dongles are -18.7 dB, -21.9 dB, -26.5 dB, -20.4 dB, respectively. We then extrapolate the data rate based on the measured EVM values. Since each dongle uses one-fourth of time resources for data transmission, the data rate should be divided by four in the calculation. Therefore, the calculated data rate is 6.0 Mbps for dongle 1,



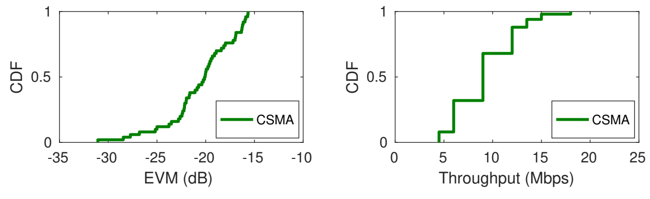
(a) Dongle 1 (b) Dongle 2 (c) Dongle 3 (d) Dongle 4

Fig. 13: The demodulated signals from the four dongles when CSMA is used (four dongles in four different time slots).



(a) Dongle 1 (b) Dongle 2 (c) Dongle 3 (d) Dongle 4

Fig. 14: The demodulated signals from the four dongles when UD-MIMO is used (four dongles in the same time slot).



(a) Measured EVM. (b) Throughput.

Fig. 15: Performance of the CSMA (interference-free) scheme.

9.0 Mbps for dongle 2, 13.5 Mbps for dongle 3, 9.0 Mbps for dongle 4. Collectively, the total throughput achieved by CSMA is 37.5 Mbps.

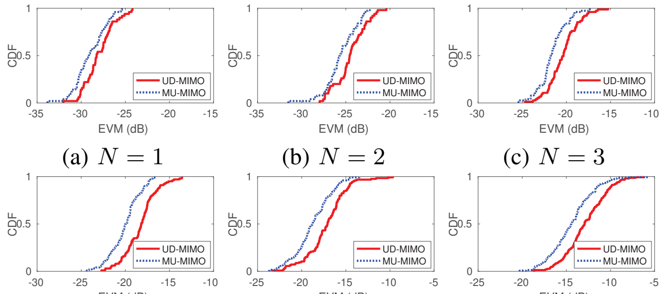
UD-MIMO: Using UD-MIMO, we let the 8 APs serve 4 Wi-Fi dongles simultaneously in the uplink. The 4 dongles are placed at d_1, d_2, d_3, and d_4 in Fig. 12. The received signals at the 8 APs are jointly decoded at the computer. Fig. 14 shows the constellation of the decoded signals. As shown in the figure, the measured EVMs are -19.7 dB, -16.1 dB, -18.4 dB, and-21.4 dB, respectively. The calculated data rates are 36 Mbps, 24 Mbps, 24 Mbps, and 36 Mbps, respectively. Collectively, the total uplink throughput achieved by UD-MIMO is 120 Mbps.

Observations: Based on the above experimental results, we have the following observations. First, UD-MIMO can serve multiple commercial off-the-shelf Wi-Fi client devices simultaneously in real wireless environments. This indicates that UD-MIMO is compatible with incumbent Wi-Fi client devices. Second, compared to conventional CSMA-based Wi-Fi networks, UD-MIMO can improve uplink throughput significantly. For the above case (8 APs and 4 dongles), UD-MIMO offers 3.2× throughput gain compared to CSMA.

D. USRP Testbed

On the USRP testbed, we measure the performance of UD-MIMO and MU-MIMO in the same scenarios and quantify their performance gap.

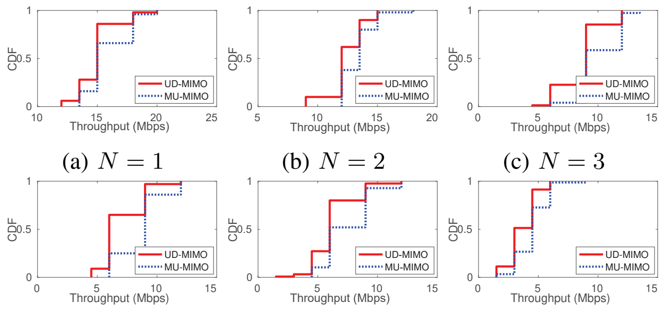
CSMA: This is an interference-free case. The interference is avoided in the time domain by assigning different STAs into different time slots. The STAs send their packets to the lead AP using a round-robin scheduler. We measure the EVM of the decoded signal for each STA at the lead AP. Fig. 15(a) plots the distribution of our measured EVM when the STA is placed throughout all the locations (small circles) in Fig. 12. Fig. 15(b) plots the distribution of our calculated data rate. Since the AP serves a single STA in each time slot, the average uplink throughput achieved by CSMA is 9.3 Mbps.



(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

Fig. 16: Measured EVM of the demodulated uplink signal from each of the N STAs.



(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

(d) $N = 4$ (e) $N = 5$ (f) $N = 6$

Fig. 17: The data rate achieved by each of the N STAs.

UD-MIMO versus MU-MIMO: In UD-MIMO, we let the 8 APs serve N STAs simultaneously in the uplink, where $1 \leq N \leq 6$. In each instance, we place the N STAs at N different locations (the small circles) in Fig. 12. At the computer, we measure the EVM of demodulated uplink signal from each of the N STAs. We then repeat the same measurements for MU-MIMO, for which we synchronize the 8 APs using an external clock (10 MHz reference signal and 1 PPS) and synchronize the N STAs using another external clock.

Fig. 16 plots the distribution of our measured EVM when UD-MIMO and MU-MIMO are used. In UD-MIMO, the average EVM of the demodulated uplink signals is -28.1 dB when the APs serve one STA, -24.5 dB when the APs serve two STAs, -20.4 dB when the APs serve three STAs, -18.3 dB when the APs serve four STAs, -17.2 dB when the APs serve five STAs, and -14.3 dB when the APs serve six STAs. Moreover, as shown in the figure, the EVM gap of UD-MIMO and MU-MIMO is only about 2.0 dB. This means that UD-MIMO successfully resolves the synchronization issues in distributed WLANs.

We extrapolate the measured EVM to each STA's data rate (with $b = 5$ Msps). Fig. 17 presents the calculated

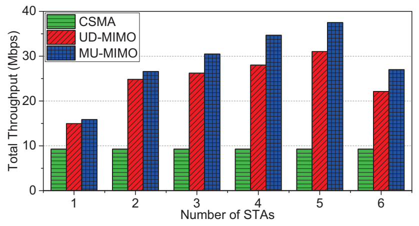


Fig. 18: Throughput comparison of CSMA, UD-MIMO, and MU-MIMO.

data rate. The staircase shape of the curves is caused by the modulation and coding scheme (MCS), which yields a discrete data rate in nature. When UD-MIMO is used, the average data rate per STA is 15.0 Mbps when the APs serve one STA, 12.4 Mbps when the APs serve two STAs, 8.7 Mbps when the APs serve three STAs, 7.0 Mbps when the APs serve four STAs, 6.2 Mbps when the APs serve five STAs, and 3.7 Mbps when the APs serve six STAs. Accordingly, the total uplink throughput achieved by UD-MIMO is 15.0 Mbps when $N = 1$, 24.8 Mbps when $N = 2$, 26.2 Mbps when $N = 3$, 28.0 Mbps when $N = 4$, 31.0 Mbps when $N = 5$, and 22.1 Mbps when $N = 6$.

Throughput Comparison: Finally, we compare the total uplink throughput achieved by CSMA, UD-MIMO, and MU-MIMO. For CSMA, since it serves one STA at a time, the total uplink throughput is the average of all STAs' data rates. For UD-MIMO and MU-MIMO, since it serves N STAs simultaneously, the total uplink throughput is the multiplication of N and the average of per-STA data rate. Fig. 18 presents the comparison of the total uplink throughput achieved by the three techniques. It is evident that, in each case, the throughput of UD-MIMO is much higher than that of CSMA and close to that of MU-MIMO. On average of the six cases, UD-MIMO achieves 3.4× throughput compared to CSMA and achieves 82% throughput of MU-MIMO. One may notice that the throughput of all three techniques decreases when the number of STAs increases from 5 to 6. This is because the sixth STA brings significant co-channel interference to the existing 5 STAs. The significant increase of co-channel interference can be attributed to the ill-conditioned MIMO channel between the 6 STAs and the 8 APs.

VII. Conclusion

In this paper, we presented UD-MIMO, a practical uplink distributed MIMO scheme for WLANs. UD-MIMO enables concurrent data transmissions from multiple STAs to multiple APs. UD-MIMO is compatible with commercial off-the-shelf 802.11 devices (with modified driver). The enabler behind UD-MIMO is a new signal detection method, which can decode concurrent data packets from asynchronous STAs. We have built a prototype of UD-MIMO on two wireless testbeds and demonstrated its compatibility with Qualcomm Atheros 802.11 devices. Our experimental results show that UD-MIMO offers 3.4× throughput compared to the CSMA-based interference-avoidance approach. Our experimental results also show that UD-MIMO achieves 82% throughput of MU-MIMO.

Acknowledgement

This work was supported in part by the NSF grants CNS-2100112, CNS-2113618, CNS-1950171, and CNS-1949753.

References

[1] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2016–2021 white paper," *Cisco, San Jose, CA, USA*, 2017.

[2] E. Charfi, L. Chaari, and L. Kamoun, "PHY/MAC enhancements and QoS mechanisms for very high throughput WLANs: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1714–1735, 2013.

[3] J. Yao, J. Xu, S. Luo, L. Wang, C. Yang, K. Wu, and W. Lou, "Comprehensive study on MIMO-related interference management in WLANs," *IEEE Communications Surveys & Tutorials*, 2019.

[4] H. S. Rahul, S. Kumar, and D. Katabi, "JMB: Scaling wireless capacity with user demands," in *Proc. of ACM SIGCOMM*, pp. 235–246, 2012.

[5] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire, "Airsync: Enabling distributed multiuser MIMO with full spatial multiplexing," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 6, pp. 1681–1695, 2013.

[6] E. Hamed, H. Rahul, and B. Partov, "Chorus: Truly distributed distributed-MIMO," in *Proc. of ACM SIGCOMM*, pp. 461–475, 2018.

[7] X. Zhang, K. Sundaresan, M. A. A. Khojastepour, S. Rangarajan, and K. G. Shin, "NEMOx: Scalable network MIMO for wireless networks," in *Proc. of ACM MobiCom*, pp. 453–464, 2013.

[8] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[9] IEEE 802.11ac, "IEEE standard for information technology local and metropolitan area networks part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput," *IEEE Standards 802.11ac*, 2014.

[10] H. Rahul, H. Hassanieh, and D. Katabi, "Sourcesync: A distributed wireless architecture for exploiting sender diversity," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 171–182, 2011.

[11] M. A. Alvarez and U. Spagnolini, "Distributed time and carrier frequency synchronization for dense wireless networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 683–696, 2018.

[12] Y.-W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 5, pp. 1085–1099, 2005.

[13] W.-Q. Wang, "Carrier frequency synchronization in distributed wireless sensor networks," *IEEE Systems Journal*, vol. 9, no. 3, pp. 703–713, 2014.

[14] A. A. Nasir, H. Mehrpouyan, S. D. Blostein, S. Durrani, and R. A. Kennedy, "Timing and carrier synchronization with channel estimation in multi-relay cooperative networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 793–811, 2011.

[15] M. M. U. Gul, X. Ma, and S. Lee, "Timing and frequency synchronization for OFDM downlink transmissions using Zadoff-Chu sequences," *IEEE Transactions on Wireless Communications*, vol. 14, no. 3, pp. 1716–1729, 2014.

[16] A. Zhou, T. Wei, X. Zhang, M. Liu, and Z. Li, "Signpost: Scalable MU-MIMO signaling with zero CSI feedback," in *Proc. of ACM MobiHoc*, pp. 327–336, 2015.

[17] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire, "Achieving high data rates in a distributed MIMO system," in *Proc. of ACM MobiCom*, pp. 41–52, 2012.

[18] A. Mahmood, R. Exel, and T. Bigler, "On clock synchronization over wireless LAN using timing advertisement mechanism and TSF timers," in *Proc. of IEEE ISPCS*.

[19] Qualcomm Atheros, "AR9271 Highly integrated single-chip USB with 802.11n support," *www.ath-drivers.eu/qualcomm-atheros-datasheets-for-AR9271.html [Online; accessed 28-Jul-2019]*.

[20] Qualcomm Atheros, "open-ath9k-htc-firmware," *https://github.com/vanhoefm/modwifi-ath9k-htc [Online; accessed 28-Jul-2019]*.

[21] Ettus Research, "USRP N210," *www.ettus.com/product/details/UN210-KIT [Online; accessed 30-Jul-2019]*, 2007.

[22] Ettus Research, "Octoclock CDA," *https://www.ettus.com/all-products/octoclock-g/ [Online; accessed 28-Jul-2019]*.