acPad: Enhancing Channel Utilization for 802.11ac Using Packet Padding

Chi-Han Lin[†], Yi-Ting Chen[†], Kate Ching-Ju Lin[‡], and Wen-Tsuen Chen[†♯]
[†]Department of Computer Science, National Tsing Hua University, Hsin-Chu, 300, Taiwan
[‡]Department of Computer Science, National Chiao Tung University, Hsin-Chu, 300, Taiwan
[♯]Institute of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan
finalspaceman@gmail.com, nancydermailbox@gmail.com, katelin@cs.nctu.edu.tw, chenwt@iis.sinica.edu.tw

***Abstract*—Multi-User Multiple Input Multiple Output (MU-MIMO) enables a multi-antenna access point (AP) to serve multiple users simultaneously, and has been adopted as the IEEE 802.11ac standard. While several PHY-MAC designs have recently been proposed to improve the throughput performance of a MU-MIMO WLAN, they, however, usually assume that all the concurrent streams are of roughly equal length. In reality, users usually have frames with heterogeneous lengths even after aggregation, leading to different lengths of transmission time. Hence, the concurrent transmission opportunities might not always be fully utilized when some streams finish earlier than the others in a transmission opportunity (TXOP). To resolve this inefficiency, this paper presents acPad, a PHY-MAC design that adds additional frames to fill up the idle channel time and better utilize the spatial multiplexing gain. Our acPad identifies proper users as the padding so as to improve the padding gain, while preventing this padding from harming all the ongoing streams. Our evaluation via large-scale trace-driven simulations demonstrates that acPad improves the throughput by up to $2.83\times$, or by $1.36\times$ on average, as compared to the conventional 802.11ac.**

I. Introduction

Multi-User Multiple Input Multiple Output (MU-MIMO) achieves substantial capacity gains by transmitting multiple data streams to a group of clients simultaneously and has been adopted as the IEEE 802.11ac standard [1] [2]. The main technique to accomplish concurrent transmissions is called *zero-forcing beamforming* (ZFBF) [3]–[5], which suppresses inter-stream interference and allows each user to only receive its intended signal. To amortize the overhead of *channel sounding* required for ZFBF precoding, 802.11ac supports frame aggregation and extends the maximum frame size to over 10,000 bytes. This implies that the lengths, and thereby the occupied channel time, of concurrent streams could be very different.

Existing literatures have proposed several signaling protocols [6] or user selection algorithms [7]–[11] to improve the sum rate of a MU-MIMO network. Most of them assume that all the concurrent streams are of roughly equal length and

This work is partially supported by the Ministry of Science and Technology of R.O.C. under contract No. 104-2628-E-009-014-MY2 and 104-2221-E-001-001-, and by Academia Sinica Thematic Research Program under contract No. AS-104-TP-A05.

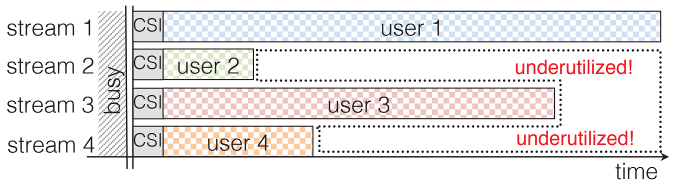


Fig. 1: Motivating example showing the need of packet padding.

can fully extract the multiplexing gains. However, in reality, some evidences have demonstrated that a large fraction of Internet packets is very small ($<$ 200 bytes), while, for some applications, such as streaming video, most of the packet sizes are close to the network's maximum transmission unit (MTU) [12]–[14]. Moreover, frame aggregation might not always be possible when a user does not have burst of packets arrived at about the same time. As a result, some streams might occupy much shorter channel time than the others, like the 4-antenna example shown in Fig. 1. Furthermore, ZFBF usually reduce the SNR of each stream if the channels of concurrent clients are not perfectly orthogonal to each other [8]–[10]. Hence, the problem of channel under-utilization could become even worse when the number of streams scales up since SNR reduction caused by ZFBF gets more critical when more streams are involved. A lower SNR leads to a lower transmission bit-rate and thereby a longer transmission time of a certain stream, which could even lower the utilization of concurrent transmission opportunities.

In this paper, we propose acPad, a PHY-MAC design that enables an AP to add additional frames as the padding to fill up the channel time of each transmission opportunity (TXOP).[1] While this idea is simple, there are, however, some fundamental challenges to be addressed. First, if the AP keeps beamforming using the ZFBF precoder selected based on the channels of the original users, the padding frame would be interfered by other ongoing original streams. How can the AP efficiently identify proper users that receive a smaller inter-stream interference and achieve a higher SINR as the padding? Second, to avoid introducing interference to a padding user, we can, alternatively, update the ZFBF precoder according to

[1]Our padding design is very different from 802.11ac's VHT PHY padding, which simply appends dummy bits to a frame such that the frame can end on a physical-level symbol boundary.

the channel of this padding user. This new precoder, however, might change the achievable SNR of the ongoing streams. How can we prevent such re-precoding from harming the decodability at the original users?

Our goal is to improve the throughput gain of padding, while ensuring the reliability of the ongoing streams. To achieve this goal, we investigate two padding schemes: *SINR-based padding* and *padding with re-precoding*. The former beamforms the padding streams using the original precoder, while the later re-calculates the precoder according to the channels of users to be added. *SINR-based padding* sorts the candidate users based on their SINR, and appends their frames in order after an original stream. To reduce the cost of information collection, we propose an efficient feedback protocol that uses *bloom filtering* [15] to allow all the candidates to report their estimated achievable rates simultaneously. This concurrent feedback needs only 3–5 OFDM symbols, which are negligible overhead. On the other hand, *padding with re-precoding* avoids inter-stream interference by finding the new ZFBF precoder for each padding user. It then exploits a novel power re-allocation algorithm to maintain the achievable SNR at any ongoing stream. Finally, we explore the advantages of both schemes, and combine them into acPad's MAC protocol to merge their strengths.

We evaluate acPad via experiments on the WARP board and large-scale trace-driven simulations. Our main findings are:

- The ratio of idle channel time in a TXOP increases as the number of concurrent streams increases. It ranges from ∼20% to ∼75% in the 4-antenna scenario.
- *SINR-based padding* selects proper users to join the padding, and, hence, achieves a higher throughput gain when there exist more candidates available for selection.
- It is harder for *padding with re-precoding* to select the best padding users due to the need of learning the channel state information (CSI) of the candidates, which is an expensive overhead. *Padding with re-precoding*, however, is more reliable when the number of antennas scales up.
- By combining the advantages of the two schemes, acPad achieves the average throughput of 1.36× over 802.11ac. The gain is related to the number of concurrent streams and the variation of frame lengths, and can be up to 2.83×.

The rest of this paper is organized as follows. We give the background of multiuser MIMO and zeroforcing beamforming in Section II. The details of the proposed padding protocol are described in Section III. We show the evaluation results in Section IV. Finally, Section V summarizes related works and Section VI concludes this work.

II. Background

The 802.11ac standard exploits zero-forcing beamforming (ZFBF) to support concurrent transmissions for multiple users. ZFBF nullifies the undesired data streams at a user, empowering it to use the standard decoder to recover its desired stream [3]–[5]. Say an M-antenna AP wants to simultaneously serve a set of N single-antenna users (denoted by $\mathcal{N}$), where $N \leq M$, and each user i receives a symbol x_i from the AP.

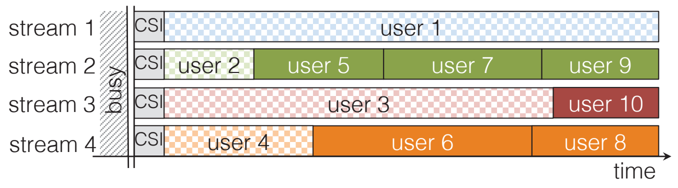


Fig. 2: Example of acPad's frame padding.

The AP allocates a transmit power P_i to the symbol x_i, and precodes x_i using the ZFBF weight vector $\mathbf{w}_i$. Then, the signal received at user i can be expressed as

$$y_i = \sqrt{P_i}\mathbf{h}_i\mathbf{w}_i x_i + \sum_{j=1,j\neq i}^{N} \sqrt{P_j}\mathbf{h}_i\mathbf{w}_j x_j + n_i, \qquad (1)$$

where $\mathbf{h}_i$ is the $1 \times M$ channel vector of user i and n_i is white Gaussian noise at user i. By finding $\mathbf{w}_j$ that satisfies $\mathbf{h}_i\mathbf{w}_j = 0$ as the precoding vector of the interfering signal $x_j, \forall j \neq i$, we can nullify the interference (i.e., the second term in Eq. (1)), enabling user i to only receive its desired symbol x_i. The ZFBF precoding vectors can also be represented as the matrix form $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_N]$, which can be found by the pseudo-inverse of the channel matrix, i.e., $\mathbf{W} = \mathbf{H}^\dagger = \mathbf{H}^*(\mathbf{H}\mathbf{H}^*)^{-1}$, where $\mathbf{H} = [\mathbf{h}_1^T \cdots \mathbf{h}_N^T]^T$ [3]–[5]. Each user i now achieves the following SNR:

$$\mathsf{SNR}_i^{\mathsf{ZFBF}} = \frac{P_i|\mathbf{h}_i\mathbf{w}_i|^2}{N_0}, \qquad (2)$$

where N_0 is the mean noise level.

However, this ZFBF precoder $\mathbf{W}$ can only ensure interference free for the target users in $\mathcal{N}$. If we replace a user $i \in \mathcal{N}$ with any other user $i' \notin \mathcal{N}$ but still use the original precoder $\mathbf{W}$, this new user i' will receive the following signal

$$y_{i'} = \sqrt{P_{i'}}\mathbf{h}_{i'}\mathbf{w}_i x_{i'} + \sum_{j=1,j\neq i}^{N} \sqrt{P_j}\mathbf{h}_{i'}\mathbf{w}_j x_j + n_{i'}. \qquad (3)$$

We can see that the interfering symbols x_j, $j \neq i$, now cannot be canceled since the channels of various users are usually different, i.e., $\mathbf{h}_i \neq \mathbf{h}_{i'}$, and, thus, $\mathbf{h}_{i'}\mathbf{w}_j \neq 0, \forall j \neq i$. That is, user i' would be interfered by the concurrent streams and obtain an SINR as follows:

$$\mathsf{SINR}_{i'}^{\mathsf{BF}}(\mathbf{W}) = \frac{P_{i'}|\mathbf{h}_{i'}\mathbf{w}_i|^2}{\sum_{j=1,j\neq i}^{N} P_j|\mathbf{h}_{i'}\mathbf{w}_j|^2 + N_0}, \qquad (4)$$

where $P_{i'} = P_i$ if we do not re-allocate the power for user i'. One important thing worth noting is that this user replacement does not affect the other users $k \in \mathcal{N}\backslash\{i\}$. This is because the original precoding vectors $\mathbf{w}_j$ still cancel the interfering signals $\mathbf{h}_k\mathbf{w}_j x_j$, $\forall j \neq k$, at user k.

III. acPad Design

acPad is a PHY-MAC design that allows the AP to efficiently utilize the idle channel time by appending additional frames as the padding, while avoiding harming the decodability of the initial selected users. The AP can use any existing user selection algorithm, such as those in [7]–[11], to pick initial users. Let $\mathcal{S}$ and $\mathcal{N}$ denote the set of users with downlink packets buffered in the AP and the set of initial served users, respectively. We assume that the AP can use aggregation/fragmentation [1] [2] to combine/divide the frames of a user. The AP then calculates the transmission time of each frame according to its frame length and the selected bit-rate, which can be found based on the SNR-based bit-rate adaptation algorithms [16] [17]. For simplicity, we use the term, *dimension*, to denote the index of a stream, i.e., the i-th stream occupying the i-th dimension of MIMO transmission, and designate the one spanning the longest transmission time as the master dimension. Hence, the rest of dimensions can be filled with other downlink frames queued in the AP, as the solid blocks shown in Fig. 2. For the sake of compact representation, hereafter, we simply use the term, *user* i, to denote the initial user occupying the i-th dimension, and use i' to represent any padding user appended in the i-th dimension. We call those remaining in the other dimensions the *ongoing users*.

We propose two padding schemes to beamform padding frames. One reuses the original precoder $\mathbf{W}$ selected according to the channels of the initial users (see Section III-A), and the other updates the precoder $\mathbf{W}'$ based on the channels of the padding users and the ongoing users (see Section III-B). Both schemes guarantee that the ongoing users achieve an SNR equal to the SNR before a padding user joins. Hence, the ongoing users can decode their desired streams as usual, without worrying the presence of the padding. We will describe in Section III-C our final MAC protocol that combines the two schemes to further improve channel utilization.

A. SINR-based Padding without Re-Precoding

In this padding scheme, when the transmission of any stream terminates, the AP picks a frame queued in the buffer and appends it immediately after the original stream, as shown in Fig. 2. Any non-initial user can overhear the transmission from the AP. Hence, the intended user of the appended frame can learn its channel from the overheard preamble and decode its frame. To avoid harming the ongoing initial users, the AP still uses the original precoder $\mathbf{W}$ to beamform not only the initial streams but also the padding streams. The advantage of this padding strategy is that the AP does not need to learn the channels of the padding users, and can save the overhead of channel sounding for padding. Therefore, we can append as many frames as possible in each dimension. However, as we mentioned in Section II, without re-precoding, the padding users would be interfered by the ongoing concurrent streams. To improve the throughput gain of padding, we should identify those candidates that achieve a higher SINR as the original precoder $\mathbf{W}$ is applied. The goal is to find the best user i' as the padding for the i-th dimension, which can be formulated as follows:

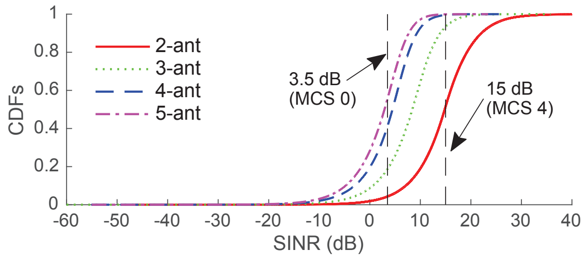
$$\begin{aligned} i' &= \arg\max_{u \in \mathcal{S} \setminus \mathcal{N}} \mathsf{SINR}^{\mathsf{BF}}_{i \leftarrow u}(\mathbf{W}) \\ &= \arg\max_{u \in \mathcal{S} \setminus \mathcal{N}} \frac{P_i |\mathbf{h}_u \mathbf{w}_i|^2}{\sum_{j=1, j\neq i}^{N} P_j |\mathbf{h}_u \mathbf{w}_j|^2 + N_0}. \end{aligned} \quad (5)$$

Unfortunately, to solve the above optimization problem, the AP still needs to know the channels of all the candidates. We, hence, propose a novel selection protocol that identifies the proper users as the padding without requiring expensive channel feedback overhead. At a high level, we ask the AP to broadcast some training symbols precoded by the precoding vectors $\mathbf{w}_i, i = 1, \cdots, N$, for all the candidate users to learn its SINR when its frame is sent in the i-th dimension. We then design a *bloom-filter based feedback protocol* that allows the AP to efficiently learn the optimal bit-rates of those candidates using a very small signaling overhead (2–5 OFDM symbols). The AP then picks the candidate producing the highest bit-rate and sends its frame when a dimension becomes idle.

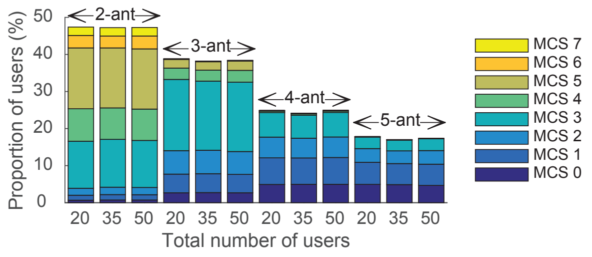
SINR estimation: A candidate user u can learn its achievable SINR based on Eq. (4) as the precoder $\mathbf{W}$ is applied. To learn SINR, user u needs to know not only its channel $\mathbf{h}_u$ but also all the precoding vectors $\mathbf{w}_i, i = 1, \cdots, N$. However, announcing the precoding matrix $\mathbf{W}$ will introduce a significant overhead. An interesting observation we made from Eq. (4) is that a user u can actually estimate its SINR by only using the information $\mathbf{h}_u\mathbf{w}_i, i = 1, \cdots, N$, without exactly knowing the precoder $\mathbf{W}$. To do so, acPad lets the AP send additional N preambles (i.e., Legacy Long Training Field, L-LTF) and precode each preamble s by $\mathbf{w}_i, i = 1, \cdots, N$, as a precoded training sequence (called C-LTF for short). Each user then receives N precoded preambles $\mathbf{h}_u\mathbf{w}_i s$ and can easily learn $\mathbf{h}_u\mathbf{w}_i$ for estimating its achievable SINR as joining the i-th dimension, i.e., $\mathsf{SINR}^{\mathsf{BF}}_{i \leftarrow u}(\mathbf{W})$ in Eq. (5). The benefit of such a learning scheme is that the required overhead is only N preambles, which is a much smaller cost as compared to precoder announcement. Note that, given a precoder $\mathbf{W}$, a user usually sees very different SINR values as its frame is sent in different dimensions. Therefore, each user should estimate its SINRs for all the dimensions $i = 1, \cdots, N$.

Bloom-filter based feedback: Since the AP appends as many users as possible in a dimension, a feasible user selection method is to sort the users in descending order of their $\mathsf{SINR}^{\mathsf{BF}}_{i \leftarrow u}$ and add the users to the i-the dimension in order. However, the cost of collecting SINR from all the candidates is extremely high, especially when a user now needs to feedback N SINR values corresponding to the N dimensions, respectively. To simplify the design, we alternatively let each user report its optimal bit-rate, which is usually very close to the achievable throughput, for every dimension. We then propose a novel *bloom-filter based feedback* mechanism that allows all the users to feedback *simultaneously* while enabling the AP to extract the list of users having the optimal bit-rate r, where each $r \in \mathcal{R}$ corresponds to a unique MCS (modulation and coding scheme) and $\mathcal{R}$ is the set of 802.11ac's available bit-rates.

We notice that most of non-initial users would receive strong inter-stream interference if the AP serves them using the original ZFBF precoder. To verify this point, we plot in Fig. 3(a) the CDFs of the SINR of the non-initial users measured in our trace-driven simulations (see detailed settings



(a) Distributions of the achievable SINR of the padding users



(b) Proportion of users in groups

Fig. 3: Properties of the achievable SINR in *SINR-based padding*.

in Section IV) for various numbers of antennas at the AP. The figure shows that only a part of users can achieve an SINR within the operational SNR range of 802.11 (i.e., above $\sim$3.5 dB as reported in [17] [18]). Also, in general, the SINR decreases as the number of antennas grows because a padding stream is allocated less power but interfered by more undesired streams. This implies that, among all the users, only a few are feasible candidates that can produce a positive padding gain. With this property, we find that *bloom filter* [15] is a very suitable technique that can efficiently detect whether an element is a member of a set when the number of members belonging to the set is much smaller than the total number of elements. Let $\mathcal{G}_r$ denote the set of users with the optimal bit-rate r. The goal of acPad's AP is to test whether a user belongs to group $\mathcal{G}_r$ and find all the users in $\mathcal{G}_r$.

The basic idea of bloom filtering is that each member uses f different hash functions, each of which maps its ID to one of the m array positions (typically $f < m$), and tags those mapped positions with "1". The server then receives the array positions tagged by all the members. To check the presence of a member, the server uses its ID to apply the same set of hash functions and detects this member if the f mapped positions are all tagged with "1". To enable bloom-filter based signaling over wireless, we adopt frequency-domain feedback. In particular, a set of m frequency-domain subcarriers is used to represent the m array positions. If a user hashes to a position, it then transmits a pulse, i.e., "1", on that mapped subcarrier and "0" on the others. If the AP detects an energy burst in a subcarrier, it knows that some users tag this position with "1". We use $|\mathcal{R}|$ sets of subcarriers and let each set of subcarriers be a bloom filter querying the users with the optimal bit-rate r. Since each user needs to report N optimal bit-rates, each for one dimension, and each 801.11ac symbol contains 52 data subcarriers, the total overhead is $\lceil\frac{N\sum_{r\in\mathcal{R}} m_r}{52}\rceil$ symbols, where m_r is the number of subcarriers allocated to group $\mathcal{G}_r$.

A bloom filter ensures that no false negative occurs, but false positives, i.e., falsely detecting a non-member as the member, might be possible. The more elements belonging to the set, the larger probability of false positives. Fortunately, the false negative probability of group $\mathcal{G}_r$ can be controlled by picking a proper number of hash functions f_r and a suitable array size m_r. According to [19], to maintain a fixed false positive probability p for group $\mathcal{G}_r$, e.g., 10%, we can set the number of hash functions f_r and the array size m_r, respectively, to

$$m_r = -\frac{|\mathcal{G}_r| \ln p}{(\ln 2)^2} \text{ and } f_r = \frac{m_r}{|\mathcal{G}_r|} \ln 2. \tag{6}$$

The remaining problem is: How can the AP estimate $|\mathcal{G}_r|$, namely the number of users having the optimal bit-rate r? To answer this question, we plot the proportion of users belonging to group r as a stacked bar graph shown in Fig. 3(b). We can see that the proportion of users in each group depends mainly on the number of AP's antennas, but is almost independent of the total number of users. Moreover, many groups, e.g., bit-rates corresponding to MCS 4–7, contain nearly no users in the 4- and 5-antenna scenarios due to the lower SINR of users. To reduce the overhead, the AP does not need to assign any subcarriers for those empty groups, and can merge those empty groups with the one of the next lower rate, e.g., clustering MCS 3–7 as one group in the 5-antenna case. Our acPad, hence, lets each AP offline learn the proportion of users choosing the optimal bit-rate r from historical frames and updates the parameters m_r and f_r every T beacon intervals. The updated information about m_r and f_r can then be piggybacked to the beacon message.

B. *Padding with Re-Precoding*

To avoid introducing interference to a padding user i' in the i-th dimension, a naïve solution is to re-calculate the new ZFBF precoder according to the channels of all the updated concurrent users, i.e., $\mathcal{N}' = \mathcal{N}\backslash\{i\} \cup \{i'\}$, by $\mathbf{W}' = \mathbf{H}_{i'}^{\dagger} = \mathbf{H}_{i'}^{*}(\mathbf{H}_{i'}\mathbf{H}_{i'}^{*})^{-1}$, where $\mathbf{H}_{i'} = [\mathbf{h}_1^T \cdots \mathbf{h}_{i'}^T \cdots \mathbf{h}_N^T]^T$. To do so, the AP needs to learn the CSI of the appended user, which incurs a relatively higher overhead as compared to the overhead of *SINR-based padding*. Hence, in *padding with re-precoding*, to control the overhead, we only add one padding frame in each non-master dimension.

While updating the precoder can eliminate the inter-stream interference for a padding user, the new precoder $\mathbf{W}'$ might change the achievable SNR of an ongoing users. The main reason is that, with ZFBF, the achievable SNR of each user closely depends on channel correlation among concurrent users. If the channels of concurrent users are not perfectly orthogonal to each other, the ZFBF precoder might reduce the SNR of a user, as compared to the SNR achieved by SISO transmission. In theory, the higher channel correlation results in more SNR reduction [7]–[11]. Since channel correlation among the updated concurrent users $\mathcal{N}'$ is usually different from channel correlation among the original concurrent users $\mathcal{N}$, the achievable SNR of any ongoing user is very likely to change as the AP applies the new precoder $\mathbf{W}'$. This SNR variation would hinder an ongoing user from reliably decoding its subsequent symbols sent at the optimal bit-rate selected based on its original SNR. Fortunately, this issue can be prevented by proper power re-allocation as we state below.

SNR-guaranteed power reallocation: An ongoing user can reliably decode its stream if its achievable SNR becomes no worse when the padding user joins. We note that, as shown in Eq. (2), the achievable SNR of an ongoing user j depends on not only the assigned precoding vector $\mathbf{w}_j$ but also the allocated power P_j. Since the ZFBF precoding vector $\mathbf{w}'_j$ should be updated based on the updated channel matrix $\mathbf{H}_{i'}$, we can only control the power of the ongoing stream to ensure SNR stability of an ongoing user j. Specifically, to avoid reducing the achievable SNR of an ongoing user j, we should re-allocate a power P'_j to user j so as to satisfy the following constraint:

$$P'_j|\mathbf{h}_j\mathbf{w}'_j|^2 = P_j|\mathbf{h}_j\mathbf{w}_j|^2, \forall j \in \mathcal{N}'\backslash\{i'\} \tag{7}$$

Hence, the minimum power required by the stream of user j, after a new user i' joins as the padding, can be found by

$$P'_j = \frac{P_j|\mathbf{h}_j\mathbf{w}_j|^2}{|\mathbf{h}_j\mathbf{w}'_j|^2}, \forall j \in \mathcal{N}'\backslash\{i'\}. \tag{8}$$

In addition, the transmit power allocated to all the streams should still be subject to the total power constraint $P_{\max}$. Then, the remaining power available for the padding stream i' becomes

$$P_{i'} = \max(P_{\max} - \sum_{j=1, j\neq i'}^{N} P'_j, 0). \tag{9}$$

Note that, if the channel of the padding user is highly correlated with the channels of the ongoing users, we might need to allocate a much higher power to the ongoing streams for maintaining their SNR, leaving no power available for the padding user. If this is the case, adding this user may not contribute any gain, and we give up this padding opportunity.

User selection: To identify the best padding user producing the maximal throughput gain, the AP should learn the CSI of all the candidates, which incurs an unacceptable large overhead and would offset the gain of padding. To avoid this, we adopt a more practical strategy to select the padding user without knowing its CSI. In particular, instead of finding the padding user maximizing the throughput improvement, we, alternatively, select the user *with the longest frame buffered in the AP* for each dimension. The rationale of this design is that, since each dimension allows at most one padding user, to compensate the cost of channel sounding, we select the user with a frame that could occupy the channel as long as possible.[2] The AP then triggers channel sounding to learn the CSI of the selected padding users immediately after 802.11ac's channel sounding, and updates the precoder and power whenever the frame of a padding user is sent.

[2]Note that the transmission time of a stream also depends on its bit-rate. However, we select the padding user without using the CSI, as a result impossible to infer its optimal bit-rate. We, hence, simplify the design and pick the padding user only according to the length of its frame.

Pilot-assisted decoder re-estimation: Another thing worth noting is that, with ZFBF, an initial user j receives the signal $y_j = \sqrt{P_j}\mathbf{h}_j\mathbf{w}_j x_j$ and uses the coefficient $v_j = \sqrt{P_j}\mathbf{h}_j\mathbf{w}_j$ as the decoder to recover its symbol x_j. In 802.11ac, the AP sends a preamble precoded by $\mathbf{w}_j$ for user j to easily learn its decoder $\sqrt{P_j}\mathbf{h}_j\mathbf{w}_j$. However, if we modify the precoder and the power of an ongoing stream j after padding, the user j now receives the signal $y_j = \sqrt{P'_j}\mathbf{h}_j\mathbf{w}'_j x_j$, which should be decoded using the updated decoder $v'_j = \sqrt{P'_j}\mathbf{h}_j\mathbf{w}'_j$, instead of v_j. A simple solution is to allow the AP to send another preamble precoded by the new precoder $\mathbf{w}'_j$ before padding such that the ongoing user j can learn its new decoder. However, this will interrupt the decoding procedure of the ongoing users and also cause extra overhead.

Fortunately, we found that such channel re-estimation is actually not necessary in our design because our power allocation ensures that, for any ongoing user j, the amplitude of its original decoder is the same with that of the updated decoder after padding, i.e., $|v_j|^2 = P_j|\mathbf{h}_j\mathbf{w}_j|^2 = P'_j|\mathbf{h}_j\mathbf{w}'_j|^2 = |v'_j|^2$, as shown in Eq. (7). That is, the decoders before and after padding only differ in their phase. We, interestingly, observe that this phase change can be naturally calibrated by 802.11's phase tracking, which is essentially designed to correct the phase rotation caused by frequency offsets between a transmitter and a receiver [20]. Simply put, phase tracking allows a receiver to keep learning the phase change θ, which, in our case, is the overall phase rotation caused by frequency offsets and our re-precoding, from the pilot subcarriers of every symbol. The user then updates the decoder by $v'_j = v_j e^{-2j\pi\theta}$ and decodes the data subcarriers using the calibrated decoder v'_j. In other words, with phase tracking, the user can still decode its symbols after re-precoding as usual without the need of channel re-estimation. We will experimentally verify this observation in Section IV-A.

C. MAC for the Joint Padding Scheme

So far, we have described our two padding schemes. *SINR-based padding* needs a small signaling overhead and can fill the idle channel time with as many padding frames as possible. However, the padding users might receive strong inter-stream interference since the AP does not re-calculate the ZFBF precoder according to their channels. On the contrary, *padding with re-precoding* can completely eliminate inter-stream interference. However, re-precoding requires the AP to learn the CSI of the padding users. Due to this relatively higher overhead, the AP adds at most one padding frame in each dimension, which might not fully utilize the idle channel time, and also does not select the optimal padding users to maximize the throughput improvement. We, hence, further propose a joint padding scheme to combine their strengths.

In short, the joint scheme starts by *padding with re-precoding*, which is then followed by *SINR-based padding* if the idle channel time has not been fully utilized. Fig. 4

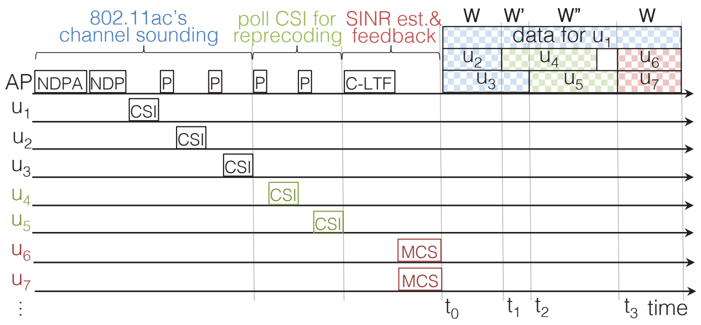


Fig. 4: acPad's MAC for joint padding (3-antenna scenario).

illustrates a 3-antenna example of joint padding, where the frames of users u_4 and u_5 are added by *padding with re-precoding* while the frames of users u_6 and u_7 are sent using the original precoder $\mathbf{W}$ of the initial users. To support this joint padding, after 802.11ac's channel sounding, acPad's AP further sends $(N-1)$ additional polling frames, each of which triggers the user appended in a non-master dimension to report its CSI. After that, the AP broadcasts N coded preambles (C-LTF), each precoded by the original precoding vector $\mathbf{w}_i, i=1,\cdots,N$, and collects the optimal bit-rates of the candidates for *SINR-based padding* using the bloom-filter based feedback mechanism mentioned in Section III-A. Whenever a user selected by *padding with re-precoding* joins (e.g., time t_1 when user u_4 joins and time t_2 when user u_5 joins as in Fig. 4), the AP calculates the new ZFBF precoder (e.g., $\mathbf{W}'$ at time t_1 and $\mathbf{W}''$ at time t_2, respectively) and re-allocates the power of all the streams according to the CSI of the concurrent users. The AP finally applies *SINR-based padding*, which adds as many users as possible in descending order of their reported optimal bit-rates so as to fill up the remaining idle time. One thing worth noting is that *SINR-based padding* should start when *all the streams sent with re-precoding end*, e.g., at time t_3 in Fig. 4. This is because the SINR of each candidate user is estimated based on the original precoder $\mathbf{W}$ and transmit power. Therefore, the AP should switch back to use the original precoder and power allocation to serve those selected users. To reduce the overhead, we ask each padding user to send acknowledgment when it is selected as the initial user of a future TXOP. In particular, it only sends ACK if its normal 802.11ac frame and the previous padding frames are all received correctly.

IV. Results

We check the effectiveness of acPad's key components via testbed experiments over the WARP board [21] and evaluate the performance of acPad via large-scale trace-driven simulations. Building the prototype of a large-scale network incurs an expensive cost of software radios. Therefore, we alternatively implement a Matlab-based simulator combined with empirical channel traces to check the performance of acPad in a network of reasonable size. To collect traces, we fix the location of a multi-antenna WARP board as the AP, while deploying another single-antenna WARP in 50 randomly-selected user locations in our testbed. We then measure the channel state information of a 2-second trace from each of the AP's antennas to the antenna in any user location, and synthesize the MIMO channels using the collected channel traces.

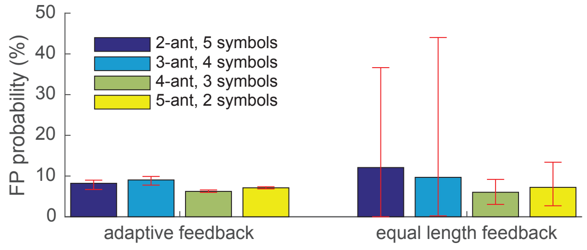


Fig. 5: False positive probability of bloom-filter based feedback.

A. Micro Benchmark

False positive of bloom-filter based feedback: We conduct a trace-driven simulation to verify the effectiveness of bloom-filter based feedback. Each of 50 users is assigned a channel trace collected from our measurements. For each TXOP, we pick N users as the initial users, where N is the number of antennas at the AP varying from 2 to 5, and calculate their ZFBF precoder. The rest of users use this precoder to estimate its SINR in each dimension and find the corresponding optimal bit-rate. In our simulation, the AP measures the proportion of users belonging to a group of bit-rate $r \in \mathcal{R}$ from the feedback collected in the previous beacon interval, i.e., 100ms, and uses the statistics to update the parameters of the number of hash functions f_r and the array size m_r at the beginning of the current beacon interval in order to ensure a false positive probability of 10%. We compare our adaptive parameter configuration with the fixed configuration, in which every group $\mathcal{G}_r$ is assigned a fixed number of subcarriers (i.e., fixed array size). For fair comparison, we let the compared scheme use the same number of subcarriers, but divide them into arrays of equal length, each assigned to a group.

Fig. 5 plots the average false positive as well as the maximum/minimum false positive in the proposed bloom-filter based feedback. The figure verifies that our adaptive configuration can always ensure a false positive below the predefined threshold 10%. This confirms that the parameters selected based on historical statistics operate properly for future frames. Note that we do not update the parameters based on the precoder of each TXOP but can still achieve a limited false positive ratio. This implies that the SINR distribution, and thereby the proportion of users in each group, are to some extent independent of the precoder of the initial users. This is because the precoder is determined by the channels of the initial users, which are typically a Gaussian complex random variables. By contrast, equal length arrays, i.e., fixed setting of f_r and m_r, fail to always guarantee a false positive of 10%. To maintain the same level of false positive, it might need more subcarriers and incur a much larger overhead. We hence conclude that our adaptive configuration achieves the required performance using the minimum overhead.

Pilot-assisted decoding: In *padding with re-precoding*, we argue that, with 802.11's phase tracking, an ongoing user can

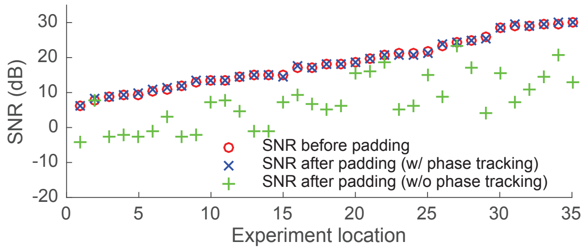


Fig. 6: SNR of the ongoing streams before and after padding.

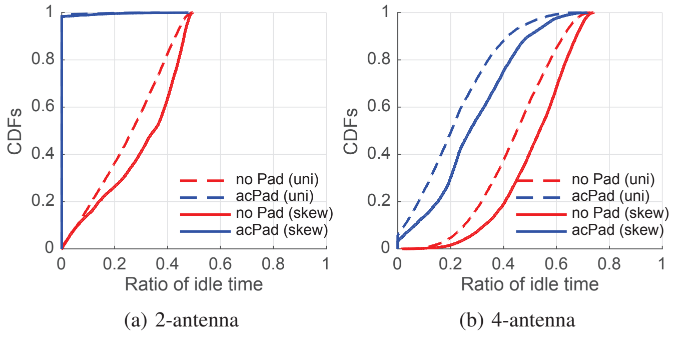


Fig. 7: CDFs of the ratio of idle channel time in TXOPs.

automatically learn its updated decoder after padding. We now experimentally verify this argument using the WARP software radio. In this experiment, we set up a 2-antenna AP serving two users simultaneously. Each user receives 50 symbols from the AP. The AP precodes the first 25 symbols of a stream using the ZFBF precoding vector $\mathbf{w}$ found based on the channels of the two users, while precoding the rest of 25 symbols using a randomly selected vector $\mathbf{w}'$, which emulates the procedure of re-precoding. We allocate equal power to the two streams for the first 25 symbols, and then adjust the power of the rest of 25 symbols such that $P|\mathbf{hw}|^2 = P'|\mathbf{hw}'|^2$. The AP sends the preamble precoded by $\mathbf{w}$ for each user to learn its initial decoder $\sqrt{P}\mathbf{hw}$ and decode all its symbols with phase tracking *enabled* and *disabled*, respectively. We deploy the users randomly in our testbed, and repeat the experiment with 35 random deployments.

Fig. 6 compares the average decoding SNR of the first 25 symbols (called the original SNR) to the average decoding SNR of the rest of symbols (called the SNR after padding) when phase tracking is enabled and disabled, respectively, in each random location. The reported results are sorted by the original SNRs in ascending order. The results show that, without phase tracking, the user will use its original decoder to decode the symbols after padding (i.e., applying a new precoder), and fail to reliably recover its symbols, leading to a significant SNR drop after padding. Fortunately, since our power allocation maintains the receiving power of each stream, with phase tracking, the user can exploit pilot subcarriers to automatically calibrate the additional phase rotation and learn the updated decoder for the new precoder. The achievable SNR after padding is, hence, almost the same with the original SNR. This allows a user to reliably decode the symbols even when any padding user joins during its ongoing transmission.

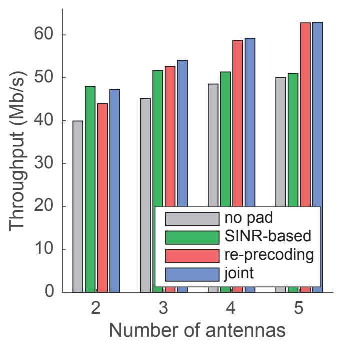
B. Trace-driven Simulations

We compare acPad with the traditional 802.11ac standard, and apply the algorithm proposed in [8] to select initial concurrent users. Our trace-driven simulation implements all the protocol overhead. We check the performance of acPad when the number of the AP's antennas varies from 2 to 5 and when the number of users varies from 10 to 50. We assume that each user always has a downlink *aggregated* frame. However, unlike existing works, which usually assume that all the users have extremely long fixed-length frames, e.g., 15,000 bytes aggregated frames used in [9], we consider a more practical frame length setting. In particular, since the idle channel time is closely related to the distribution of frame lengths, we test two different frame length distributions: uniform and skew. For uniform distribution, the frame length is randomly uniformly picked between 200 bytes and 11,454 bytes, which is the maximum MPDU size defined in 802.11ac [1] [2]. For skew distribution, we assume that all the frames are either small (200–400 bytes), e.g., HTTP packets, or large (8,000–10,000 bytes), e.g., file delivery, and the length of each frame is randomly picked from these two intervals. Each simulation outputs the average results of 10,000 TXOPs.

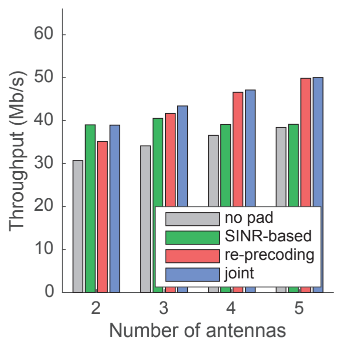
Impact of number of antennas: In this simulation, we fix the number of users to 50, and assign a channel trace to each user. Before demonstrating the throughput performance, we first show in Fig. 7 the CDFs of the ratio of idle channel time, before and after applying acPad's joint padding, of all the frames in the 2-antenna and 4-antenna scenarios. The ratio of idle channel time is formally defined as $\sum_i T_i^{\text{idle}}/(NT_{\max})$, where T_i^{idle} is the idle time of the i-th dimension in a TXOP and $T_{\max}$ is the transmission time of the stream in the master dimension. We can see that, without padding, the ratio of idle channel time in the 4-antenna scenario is much higher than that in the 2-antenna scenario. This is because when there are more streams, it is more likely that any stream in a non-master dimension could not fully utilize the channel. The figures also show that the ratio of idle time is higher when the frame length distribution is skew as more small frames would be sent concurrently with a long frame. After applying acPad, we can almost fully utilize the channel in the 2-antenna scenario and significantly reduce the idle time in the 4-antenna scenario. The idle time cannot be reduced to nearly 0 in the 4-antenna case since our joint padding design can only start *SINR-based padding* until all the frames sent with re-precoding finish. As a result, there are still some idle time if the frames sent with re-precoding occupy different lengths of channel time.

Figs. 8(a) and (b) illustrate the average throughput of the comparison schemes for uniform and skew distribution, respectively, while Figs. 8(c) and (d) plot the throughput gain of different padding schemes over the conventional 802.11ac. Our findings are as follows:

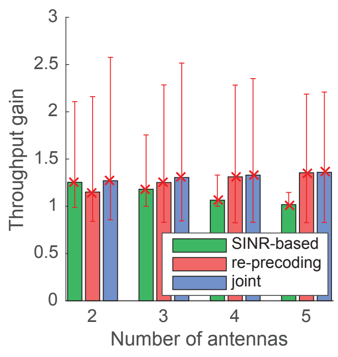
- In 802.11ac, the total average throughput grows as more users are involved in concurrent transmissions. However, the improvement gets slower when the number of antennas at



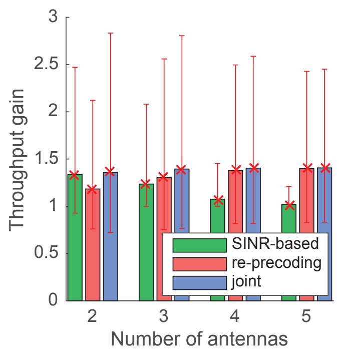
(a) Throughput (uniform)



(b) Throughput (skew)

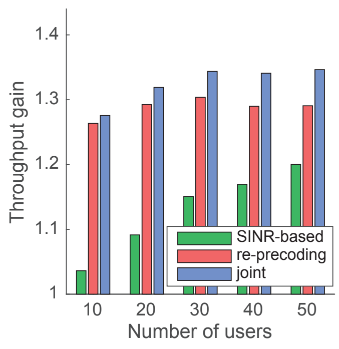


(c) Throughput gain (uniform)

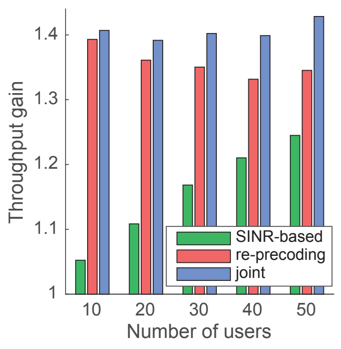


(d) Throughput gain (skew)

Fig. 8: Impact of number of antennas.



(a) Throughput gain (uniform)



(b) Throughput gain (skew)

Fig. 9: Impact of number of users.

the AP increases since, as we have shown in Fig. 7, the channel utilization actually decreases due to variable lengths of concurrent streams. This also explains why the average throughput for skew distribution is slightly lower than that for uniform distribution.

- Figs. 8(c) and (d) show that *SINR-based padding* achieves a higher gain in the 2-antenna scenario, but has a decreasing gain when the number of antennas scales up. The main reason is that, without updating the ZFBF precoder, a padding stream is allocated less power, but would be interfered by more interfering streams. As a result, the achievable SINR of a padding user usually decreases as more streams are served simultaneously.
- *Padding by re-precoding*, in general, achieves a higher gain because re-precoding ensures interference free for the padding users. It performs worse than SINR-based padding in the 2-antenna scenario due to its higher overhead for additional channel sounding. However, the gain achieved by re-precoding increases as the number of antennas increases and more idle channel time is utilized, amortizing the additional cost of channel sounding.
- acPad's joint design combines the advantages of the two schemes and hence outperforms both in most of cases.
- We also plot in Figs. 8(c) and (d) the maximum gain and minimum gain among all TXOPs. The figures show that the minimum gain might be lower than 1 because, in a very few cases, we might have spent the overhead to learn the CSI of the padding users (for *padding with re-precoding*) or collect the optimal bit-rates of the candidates (for *SINR-based padding*), but fail to find any feasible padding user, i.e., a user producing a positive throughput gain. However, except for those rare cases, our acPad produces the throughput gain of $1.36\times$ on average, while the gain can be up to $2.83\times$.

Impact of number of users: We further check the impact of number of users in Fig. 9 when the number of antennas is set to 3. The figures again verify that the throughput gain of padding can be higher when the frame length distribution is skew, which leads to longer idle channel time. The results also show that the throughput of *padding with re-precoding* is less correlated to the number of users since it selects the padding users more implicitly based only on the frame length. However, the gain of *SINR-based padding* obviously increases as the number of users scales up because the AP can identify a user achieving a higher SINR if more candidates join selection.

V. Related Work

In the last few years, MU-MIMO networks have attracted much attention from research and industrial communities [3]–[5]. Several recent works further increase the capacity of a MU-MIMO system by scaling up the number of antennas at an AP [22], [23] or combining distributed antennas as a large network MIMO system [24]–[26]. In a MU-MIMO, the performance of ZFBF is closely related to how users are grouped to join concurrent transmissions. Therefore, several user selection algorithms [7]–[11] have been proposed to identify a set of concurrent users with less channel correlation and thereby producing a higher sum-rate for either downlink or uplink MU-MIMO. acPad's user selection differs from the above proposals in that it seeks for proper padding users that can coexist with the ongoing users without affecting their decoding process.

The performance of a MU-MIMO network also depends on how much power each stream is allocated subject to the total power constraint. The simplest but suboptimal way is equal power allocation, which transmits all the concurrent streams using the same level of power. Other power allocation strategies have been proposed to improve the capacity [27] [28] or guarantee fairness [29] [30]. Unlike the above approaches that try to improve the performance for a given set of concurrent users, acPad's power allocation for *padding with re-precoding* aims at keeping the receive power of all the ongoing streams as any user is replaced by a new padding user and the ZFBF precoder has to be changed accordingly.

Our bloom-filter based feedback is related to the previous efforts on reducing the overhead of channel feedback via quantization [31], compression [32], analog feedback [33] or selective feedback [34] [35] The first three types of solutions reduce the size of each feedback message, while the last type asks only a subset of users qualified to be served to feedback. However, all those approaches still trigger users to report their information *sequentially*. On the contrary, acPad lets all the valid candidates feedback *simultaneously* using bloom-filter based feedback over frequency-domain subcarriers. The AP can hence collect all the required information using only a few symbols and reduce the overhead significantly.

VI. Conclusion

This paper presents acPad, a frame padding protocol that appends additional frames as the padding to utilize the idle channel time in 802.11ac. Our acPad combines two padding schemes, one using the original ZFBF precoder but properly selecting the users achieving a high SINR and the other re-calculating the ZFBF precoder but reallocating power to protect the ongoing users. We propose several PHY-MAC designs that collect the required information using a very small overhead, while preventing the padding frames from harming the decodability of 802.11ac's original frames. We demonstrate via trace-driven simulations that the ratio of idle channel time can be up to about 75%; by efficiently improving channel utilization, acPad produces a higher throughput than the conventional 802.11ac in most of scenarios.

References

[1] "IEEE Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks– Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications– Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz," *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)*, pp. 1–425, Dec. 2013.

[2] M. S. Gast, *802.11ac: A Survival Guide*. O'Reilly Media, Inc., 2013.

[3] T. Yoo and A. Goldsmith, "On the Optimality of Multiantenna Broadcast Scheduling Using Zero-Forcing Beamforming," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 528–541, Mar. 2006.

[4] E. Aryafar, N. Anand, T. Salonidis, and E. W. Knightly, "Design and Experimental Evaluation of Multi-User Beamforming in Wireless LANs," in *ACM MobiCom*, 2010.

[5] H. Yu, L. Zhong, A. Sabharwal, and D. Kao, "Beamforming on Mobile Devices: a First Study," in *ACM MobiCom*, 2011.

[6] T. Wei and X. Zhang, "Random Access Signaling for Network MIMO Uplink," in *IEEE INFOCOM*, 2016.

[7] N. J. A. G. Taesang Yoo, Student Member, "Multi-Antenna Downlink Channels with Limited Feedback and User Selection," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 7, pp. 1478–1491, Sep. 2007.

[8] X. Xie and X. Zhang, "Scalable User Selection for MU-MIMO Networks," in *IEEE INFOCOM*, 2014.

[9] N. Anand, J. Lee, S. J. Lee, and E. W. Knightly, "Mode and User Selection for Multi-User MIMO WLANs without CSI," in *IEEE INFOCOM*, 2015.

[10] A. Zhou, T. Wei, X. Zhang, M. Liu, and Z. Li, "Signpost: Scalable MU-MIMO Signaling with Zero CSI Feedback," in *ACM MobiHoc*, 2015.

[11] W.-L. Shen, K. C.-J. Lin, M.-S. Chen, and K. Tan, "Sieve: Scalable User Grouping for Large MU-MIMO Systems," in *IEEE INFOCOM*, 2015.

[12] R. Sinha, C. Papadopoulos, and J. Heidemann, "Internet Packet Size Distributions: Some Observations," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643, May 2007.

[13] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig, *Investigating IPv6 Traffic*. Springer Berlin Heidelberg, March 2012, pp. 11–20.

[14] S. Sengupta, H. Gupta, P. De, B. Mitra, S. Chakraborty, and N. Ganguly, "Understanding data traffic behaviour for smartphone video and audio apps," in *The International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1–2.

[15] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[16] W.-L. Shen, Y.-C. Tung, K.-C. Lee, K. C.-J. Lin, S. Gollakota, D. Katabi, and M.-S. Chen, "Rate Adaptation for 802.11 Multiuser MIMO Networks," in *ACM MobiCom*, 2012.

[17] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 Packet Delivery from Wireless Channel Measurements," in *ACM SIGCOMM*, 2010.

[18] H. Rahul, F. Edalat, D. Katabi, and C. Sodini, "Frequency-Aware Rate Adaptation and MAC Protocols," in *ACM MobiCom*, 2009.

[19] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and Practice of Bloom Filters for Distributed Systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.

[20] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams Publishing, 2001.

[21] "WARP: Wireless Open Access Research Platform," http://warpproject.org/trac/.

[22] Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, "BigStation: Enabling Scalable Real-time Signal Processing in Large MU-MIMO Systems," in *ACM SIGCOMM*, 2013.

[23] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical Many-Antenna Base Stations," in *ACM MobiCom*, 2012.

[24] H. S. Rahul, S. Kumar, and D. Katabi, "JMB: Scaling Wireless Capacity with User Demands," in *ACM SIGCOMM*, 2012.

[25] X. Zhang, K. Sundaresan, M. A. A. Khojastepour, S. Rangarajan, and K. G. Shin, "NEMOx: Scalable Network MIMO for Wireless Networks," in *ACM MobiCom*, 2013.

[26] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi, "Real-time Distributed MIMO Systems," in *ACM SIGCOMM*, 2016.

[27] D. N. Tse, "Optimal Power Allocation over Parallel Gaussian Broadcast Channels," in *IEEE ISIT*, 1997.

[28] J. Wang, D. J. Love, and M. D. Zoltowski, "User Selection With Zero-Forcing Beamforming Achieves the Asymptotically Optimal Sum Rate," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3713–3726, Aug. 2008.

[29] Z. Shen, J. G. Andrews, and B. L. Evans, "Optimal Power Allocation in Multiuser OFDM Systems," in *IEEE GLOBECOM*, 2003.

[30] I. C. Wong, Z. Shen, B. L. Evans, and J. G. Andrews, "A Low Complexity Algorithm for Proportional Resource Allocation in OFDMA Systems," in *IEEE Workshop on Signal Processing Systems (SIPS)*, 2004.

[31] K. K. Mukkavilli, A. Sabharwal, E. Erkip, and B. Aazhang, "On Beamforming With Finite Rate Feedback in Multiple-Antenna Systems," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2562–2579, Oct. 2003.

[32] T. Eriksson and T. Ottosson, "Compression of Feedback for Adaptive Transmission and Scheduling," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2314 – 2321, Dec. 2007.

[33] O. E. Ayach and R. W. Heath, "Interference Alignment with Analog CSI Feedback," in *IEEE MILCOM*, 2010.

[34] V. Hassel, D. Gesbert, M.-S. Alouini, and G. E. Øien, "A Threshold-Based Channel State Feedback Algorithm for Modern Cellular Systems," *IEEE Transactions on Wireless Communications*, vol. 6, no. 7, pp. 2422 – 2426, Jul. 2007.

[35] S. Sanayei and A. Nosratinia, "Opportunistic Downlink Transmission With Limited Feedback," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4363 – 4372, Nov. 2007.