KPad: Maximizing Channel Utilization for MU-MIMO Systems using Knapsack Padding

Song Wang*, Jingqi Huang* and Anfu Zhou
Beijing Key Lab of Intell. Telecomm. Software and Multimedia, Beijing University of Posts and Telecomm
{wangsong17, huangjingqi, zhouanfu}@bupt.edu.cn

***Abstract*— In a Multi-User Multiple Input Multiple Output (MU-MIMO) system, an access point (AP) equipped with multiple antennas can serve multiple users simultaneously (*i.e.*, support concurrent multi-streams) and thus achieves multi-fold throughput gain. In practice, however, the gain is significantly comprised by frame-size diversity, *i.e.*, shorter frames need to wait for the finish of the longest frame, which leads to low channel utilization and thus throughput degradation. Frame padding (*i.e.*, more than one short frames are grouped together to fill in the idle channel) has been proposed to solve the problem, but existing approaches are based on heuristic and cannot fully exploit the potential of padding. In this paper, we propose Knapsack Padding (KPad), a novel model-driven frame padding design to maximize MU-MIMO channel utilization. We first formally formulate the frame padding problem as a *multi-stream knapsack* model, and then design a stream decoupling mechanism to handle the unique and complicated inter-stream interference underlying the model, so as to derive the optimal padding schedule efficiently. We evaluate KPad using trace-driven emulation. Extensive evaluation results demonstrate remarkable throughput gain (up to 42%) compared with the state-of-the-art.**

I. Introduction

Multi-User Multiple Input Multiple Output (MU-MIMO) is the key technology that enables high-speed wireless access [2], [4]–[7], and has been adopted by mainstream IEEE 802.11ac [3], [19], 4G LTE [20] and also the coming 802.11ax [1] and 5G wireless networks. In contrast to the Single Input Single Output (SISO), an Access Point (AP) or Based station (BS) in MU-MIMO is usually equipped with multiple antennas, which provide multiple concurrent transmission opportunities (TXOP). MU-MIMO exploits the TXOP to enable multiple concurrent transmissions between the AP and client users, as if there were multiple parallel links. Theoretically, the throughput gain increases linearly with the number of antennas on the AP [21].

In practice, however, throughput gain of a MU-MIMO system is severely compromised by under-utilized TXOP, due to frame-size diversity issue [8]. We illustrate this in Fig. 1(a), where three frames with different sizes are transmitted concurrently in a round of MU-MIMO transmission. After the shortest frame finishes, it needs to wait until the longest frame finishes. In consequence, much channel time is wasted, which leads to poor network performance. Note that frame-size diversity is a common phenomenon in Internet traffic (*i.e.*, frame sizes are widely-distributed from dozens of bytes to maximum transmission unit, or MTU usually of 1500 bytes [13]–[15]), which renders remarkable channel time waste. One straightforward solution is *frame aggregation* (as regulated in 802.11ac [3]), in which short frames from a user are aggregated together in a TXOP (also referred to as a stream in literature), so the total length of each TXOP can be aligned as close as possible. However, frame aggregation might not always be possible when a user does not have a burst of frames. Moreover, the frame aggregation may hurt performance of real-time applications (*e.g.*, VoIP or video telephony) where frames need to be sent out immediately after being generated.

* Co-primary authors.

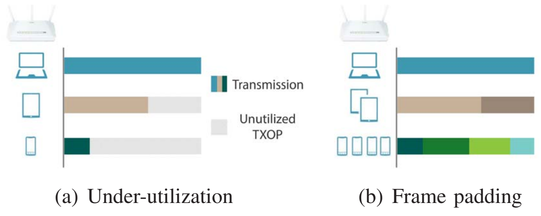


Fig. 1. No padding vs. frame padding.

Alternatively, one can break the limit of frame aggregation (*i.e.*, requiring that all frames in a stream should come from the same user), and append frames from different users in one stream (referred to as *frame padding*), as shown in Fig. 1(b). While frame padding can combat the inefficiency of frame aggregation, it is confronted with the extra challenge of handling dynamic inter-steam interference. In particular, channel correlation keeps evolving dynamically as frames from different users join a stream. The dynamic needs to be deliberately handled since the overall performance of a MU-MIMO system is largely determined by the inter-stream interference [21]. The state-of-the-art work, referred to as acPad [8], proposes a heuristic-based MAC-PHY design for IEEE 802.11ac frame padding. acPad selects initial users based on their channel correlation, and then incrementally pads users afterward, without taking into account the effect of padding users from the beginning. As a result, acPad is unable to achieve maximum performance, as validated in Sec. IV.

In this paper, we propose *KPad*, a novel user padding design to maximize the channel utilization and thus to optimize the MU-MIMO system performance. Instead of heuristic-based approach, we formally formulate the user padding as a *multi-stream knapsack* model, *i.e.*, a variant of the classical knapsack model [22]. To address complicated dynamic inter-stream interference in the model, we propose a 'step-by-step' greedy stream decoupling mechanisim to decouple the inter-stream interference, so as to derive an optimal padding users efficiently. Accordingly, we design the KPad algorithm that schedules padding users optimally to fully utilize the channel resource.

TABLE I
SUMMARY OF SYMBOLS

Symbol	Definition
M	Number of antennas at AP
N	Number of users
Ψ	User set
ψ_i	The i-th user in Ψ
$\mathbf{H}$	Channel state matrix between AP and users
$\mathbf{W}$	Precoding vectors matrix
T_{max}	Length of the longest frame in Ψ
l_i	Length of transmission time of ψ_i
λ_i	Throughput brought by ψ_i
$\boldsymbol{U}_m$	The set of selected user at stream m
r_i	Bit rate of ψ_i
γ_i	Equivalent rate of r_i
$\epsilon_{i,j}$	Throughput loss of ψ_i caused by inter-stream interference
$\xi[\tau]$	Thoughput achieved by time limit τ

We evaluate KPad using an emulation approach. In particular, we collect channel trace of up to 50 users in a typical office using WARP SDR board [9], and then feed the trace into a simulator to conduct extensive simulations. We compare KPad against the benchmark baseline of No Pad [3], and the state-of-the-art acPad [8]. The main results are as follows:

- KPad can significantly reduce idle channel time under various scenarios. KPad fills up to 99% channel time in average, which is $1.42\times$ over No Pad and $1.12\times$ over acPad.
- KPad achieves significant throughput gain. In average, KPad's throughput is $1.52\times$ and $1.42\times$ (for 2-antenna AP), $1.32\times$ and $1.21\times$ (for 3-antenna AP) over No Pad and acPad, respectively.
- KPad exhibits a desirable advantage of good scalability, *i.e.*, its performance gain becomes more remarkable as the number of client users increases. For example, the throughput gain of KPad increases from $1.03\times$ to $1.42\times$ over acPad, when the number of client users increases from 7 to 45 in a MU-MIMO system with 2-antenna AP.

The rest of paper is organized as follows. We present system model in Sec. II, and then describe system design of KPad in Sec. III. We validate the performance of KPad in Sec. IV and conclude the paper in Sec. V.

II. System Model

Here we briefly introduce MU-MIMO system, in particular the coding/decoding process which is relevant to frame padding design. We consider a M-antenna AP and a collection of N users $\Psi = \{\psi_1, \psi_2, ..., \psi_N\}$. For simplicity, we assume that each user has one antenna, but it is straightforward to extend to the case of users with multi-antenna. Let x_i be the desired symbol for user ψ_i and $\mathbf{h}_i$ be the $1 \times M$ channel vector between M antennas of AP and user ψ_i. In MU-MIMO systems, x_i is precoded by AP before the transmission, using the $1 \times M$ ZFBF precoding weight vector $\mathbf{w}_i$. After channel distortion, the received signal at user ψ_i will be as follows:

$$y_i = \mathbf{h}_i\mathbf{w}_i x_i + \sum_{\psi_j\in\Psi, j\neq i} \mathbf{h}_i\mathbf{w}_j x_j + n_i \tag{1}$$

where $\sum_{j\in\Psi,j\neq i}\mathbf{h}_i\mathbf{w}_j x_j$ is the interference from concurrent users, and n_i is the noise. We can remove the interference by ensuring: $\mathbf{h}_i\mathbf{w}_j = 0, \forall j \neq i$. As a result, ψ_i can receive its own desired data symbol without interference caused by other users. Generally, we denote channel state matrix between all users and all transmit antennas of AP as $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]^T$, and say $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N]$ as the matrix form of precoding vectors. $\mathbf{W}$ can be computed from the pseudo inverse as $\mathbf{W} = \mathbf{H}^\dagger = \mathbf{H}^*\left(\mathbf{H}\mathbf{H}^*\right)^{-1}$ [10].

The pseudo inverse indicates that $\mathbf{h}_i\mathbf{w}_i = 1$. After such precoding and channel distortion, SNR of user ψ_i can be stated as follows:

$$SNR_i = \frac{|\mathbf{h}_i\mathbf{w}_i|^2}{N_0} \tag{2}$$

Note that the precoder $\mathbf{W}$ can only nullify the interferences between users in Ψ. Consider MU-padding, when a user k ($\psi_k \in \Psi, k \neq i$) has finished its transmission, it may be substituted by another user k', $k' \in \Psi$. After the substitution, $\mathbf{w}_k$ changes to $\mathbf{w}_{k'}$. Now we denote the updated precoder with $\mathbf{W}'$. In consequence, the interferences, *i.e.*, the second RHS term in (1), does not equal to zero any longer. Now received signal of user ψ_i can be expressed as:

$$y_i' = \mathbf{h}_i\mathbf{w}_i x_i + \sum_{\psi_j\in\Psi, j\neq i,k} \mathbf{h}_i\mathbf{w}_j x_j + \mathbf{h}_i\mathbf{w}_{k'} x_{k'} + n_i \tag{3}$$

Now the precoder has changed, *i.e.*, $\mathbf{w}_k \neq \mathbf{w}_{k'}$, thus $\sum_{\psi_j\in\Psi,j\neq i,k}\mathbf{h}_i\mathbf{w}_j x_j + \mathbf{h}_i\mathbf{w}_{k'} \neq 0$. Clearly, the interference from other users cannot be nullified any longer. Influenced by the interference caused by concurrent users, user ψ_i will have the following SNR:

$$SNR_i' = \frac{|\mathbf{h}_i\mathbf{w}_i|^2}{\sum_{\psi_j\in\Psi,j\neq i}|\mathbf{h}_i\mathbf{w}_j|^2 + |\mathbf{h}_i\mathbf{w}_{k'}|^2 + N_0} \tag{4}$$

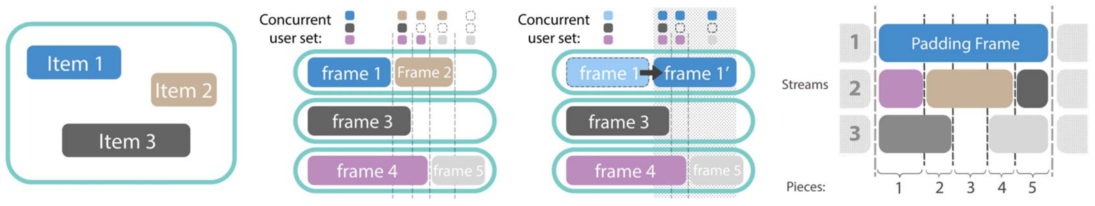
Interference that appears after the substitution leads to SNR degradation. Therefore, we should make an optimal tradeoff between throughput boost from extra padding transmissions and the additional interference caused by the appended users, which is key challenge underlying MU-MIMO padding.

III. KPad design

A. High-level Overview

In our design, we model MU-MIMO frame padding as a variation of knapsack problem. Knapsack problem is a classic combinational optimization problem. In particular, given some *items* with certain values and weights, the objective is to pack the *knapsack* to get the maximum total value under the constraint of a total weight limit [18]. We find that MU-MIMO padding problem shares a great deal of similarities with the knapsack problem. In particular, the frames in MU-MIMO padding are analogous to *items* which is to be filled into length-limited streams, to achieve maximum throughput. Despite the similarity, there are two major differences worth noting between conventional knapsack problem and MU-MIMO padding:

- *From single-stream packing to multi-stream padding*: A stream is analogous to a knapsack since they both possess limited ability to accommodate items. MU-MIMO exploits multiplexity by accommodating more than one concurrent streams in one round of concurrent transmission, which is equivalent to multiple knapsacks to be filled at the same time. Note that multiple streams are not independent (as detailed below), which introduces fundamentally new challenges.



(a) Conventional Knapsack (b) Multistream Knapsack (c) Influence of frame position (d) Entanglement of frames

Fig. 2. Comparison of MU-MIMO padding problem and knapsack problem.

- *Inter-stream inference*: According to (4), temporal overlapping of frames from concurrent streams causes SNR degradation and in consequence throughput loss. Furthermore, different concurrent user sets lead to different SNR degradation and temporal positions of frames in streams complicate the overlapping relationship. As an example, Fig. 2(b) shows 2 frames in the first stream overlapping with multiple items in other streams. Due to the diversity of frame lengths, the overlappings are not strictly aligned, which results in complicated inter-stream interference. Moreover, a slight change of frame position can lead to very different inter-stream interference. As shown in Fig. 2(c), if we move frame 1 to position $1^{'}$, the inter-stream interference and the overall MU-MIMO throughput will be significantly different.

The inter-stream inference has not been addressed in previous variation of knapsack problem. *First*, temporal positions of frames in a stream plays critical role in determining overall throughput. *Second*, complicated entanglement of inter-stream overlapping makes it hard to compute exact SNR degradation (and thus throughput loss), let alone to select and schedule frames based on it. To address above challenges, we first design an algorithm that computes throughput loss caused by a frame efficiently given transmission schedule. Then we take a greedy 'step-by-step' approximation method to decouple inter-stream interference, and design a viable procedure that schedules padding frames to achieve maximum throughput. In the following, we detail these design modules one by one.

B. Multi-stream knapsack formulation

For MU-MIMO padding problem, frame collection Ψ is analogous to the *item* set: a transmitted frame ψ_i occupies a proportion of limited TXOP. The occupied time is equivalent to item *weight* and can be represented by the length l_i of the frame. Meanwhile, the frame brings certain throughput λ_i to MU-MIMO system as its value. λ_i can be represented in the form of transmission length multiplying by bit rate: $\lambda_i = l_i r_i$. According to ShannonHartley theorem [11]:

$$r_i = \boldsymbol{B} log_2(1 + SNR_i) \tag{5}$$

where $\boldsymbol{B}$ is the bandwidth of the channel between i and its AP. For most MU-MIMO system transmission streams share a common bandwidth, we can thus use equivalent rate $\gamma_i = \frac{r_i}{\boldsymbol{B}}$ to represent bit rate. Accordingly the equivalent frame value can be written as $\gamma_i l_i$. Similar to frame, we regard a stream as a knapsack which imposes a certain limit ϕ_i on total length of frames it accommodates. Essentially our goal is to select a user group $\boldsymbol{U}_m$ for every stream m that maximizes overall throughput while meeting the limits required by each stream:

$$\begin{aligned} &\max \sum_{m=1}^{M} \sum_{i \in \boldsymbol{U}_m} \gamma_i l_i - \epsilon_{i,m} \\ &s.t. \sum_{i \in \boldsymbol{U}_m} l_i \le \Phi \end{aligned} \tag{6}$$

where $\epsilon_{i,m}$ is the throughput loss ψ_i suffers due to inter-stream interference. Here we can see the differences more clearly. In (6), the overall throughput depends on not only on sum throughput of frames, but also on the throughput loss $\epsilon_{i,m}$ which is, as mentioned earlier in this section, caused by inter-stream inferences.

Another major difference is that the temporal positions of frames play a role in overall throughput since they determine transmission overlaps. Since the objective of frame selection is maximizing overall throughput, position information of frame is inevitably involved in selection and scheduling of frames. We create a structure st in which two fields are defined: p array records the frames added to the stream in order; s array stores starting time of each frame in p. We form a structure matrix $sts[M, \Phi]$ with each element $sts[m, \phi]$ a structure st indicating the selection and scheduling of the substream of m up until ϕ.

C. Step-by-step interference decoupling

Earlier in this section, we illustrate how throughput loss caused by inter-stream interference differentiates MU-MIMO padding from knapsack problem, and its influence on overall throughput. Thus an algorithm that computes throughput loss efficiently is crucial to the following user selection algorithm. Similar to frame value, throughput loss can be represented as bit rate drop $\Delta\gamma_i$ multiplying by corresponding length l_o. The bit rate drop part is pretty straightforward. According to (2) and (4), $\Delta\gamma_i$ can be written as:

$$\Delta\gamma_i = \gamma_i - \gamma_i' = log_2 \frac{1 + SNR_i}{1 + SNR_i'} \tag{7}$$

The difficulties of computing throughput loss lie in finding the lengths for corresponding bit rate drop. The entanglement of frames can be complicated when the number of streams increases. Fig. 2(d) shows an example of convoluted overlap relationship among 3 streams. The padding frame is aligned to (part of) 3 frames in stream 2, which are again aligned to another two frames and a fragment of idle time in stream 3. The complicated entanglement makes it difficult to compute throughput loss of the frame.

To address this problem, instead of treating the inference period as a whole, we decouple it and accumulate throughput loss in ”step-by-step” manner. The dotted lines in Fig. 2(d) indicate start/end time of steps. One can see that they separate the overlap time of padding frame into 5 smaller ”steps” that contain at most one frame (or fragment of frame) for one stream. Consequently, the bit rate drop within a ”step” is constant and can be calculated with (4). The overlap relationships are thus simplified by ”steps” and the overall throughput loss can be derived by adding up throughput loss of all ”steps”.

We designed an efficient algorithm to calculate throughput loss given ψ_p and schedule of other streams. Algorithm 1 shows the main procedures. The basic idea is to chop the occupied channel time of ψ_p into aforementioned ”steps”. To implement the procedure, we define two M sized arrays: br stores the start and end times of frames in $\hat{\Psi}$; p is a list of frames in current ”step”. Instead of breaking occupied time of ψ_p into ”steps” all at once, we chop a piece at a time, calculate its throughput loss, and then move on to the next piece. By this way, we minimize the calculation of (7) and reduce the redundancy of storing length and temporal location of ”steps”.

Algorithm 1 Step-by-step throughput loss calculation

```
1: $s \leftarrow$ start time of $\psi_p$
2: $e \leftarrow$ end time of $\psi_p$
3: $c \leftarrow e$
4: for m = 1:M do
5:    Find the first frame $\hat{\psi}$ in $m$ : start time of $\hat{\psi} \geq s$
6:    $p[m] \leftarrow \hat{\psi}$
7:    $br \leftarrow$ end time of $\hat{\psi}$
8: end for
9: while $min(br) < e$ do
10:    $\tau_{min} \leftarrow min(br)$
11:    $m_{min} \leftarrow argmin(br)$
12:    $TP \leftarrow TP + \sum_{i \in p} \Delta\gamma_i(\tau_{min} - c)$
13:    $p[m_{min}] \leftarrow$ next packet in $m_{min}$
14:    $br[m_{min}] \leftarrow$ end time of $p[m_{min}]$
15:    $c \leftarrow \tau_{min}$
16: end while
```

D. Greedy approximation to a knapsack variation

The concept of knapsack problem is pretty straightforward: Given a knapsack with weight limit W_{max} and a set of items with each has some weight w_i and benefit value v_i, the goal is to choose a subset of items I to achieve maximal value sum within W_{max}. It can be formulated as:

$$\begin{aligned} &max \sum_{i \in I} v_i \\ &s.t. \sum_{i \in I} w_i \leq W_{max} \end{aligned} \tag{8}$$

Several algorithms have been proposed to address knapsack problem [16], [17]. The most commonly used one is based on dynamic programming. We define $m[\omega]$ as the maximum value achievable at weight limit $\omega, \omega \leq W_{max}$. Thus the goal is converted to finding $m[W_{max}]$. We can observe following properties of $m[\omega]$:

$$\begin{aligned} &m[0] = 0 \\ &m[\omega] = max_{w_i \leq \omega}(m[\omega - w_i] + v_i) \end{aligned} \tag{9}$$

Algorithm 2 KPad algorithm

```
1: Find the frame $\psi_L$ with the largest length
2: $T_{max} \leftarrow$ length of $\psi_L$
3: $sts[1, T_{max}].p \leftarrow \{\psi_L\}$
4: $sts[1, T_{max}].s \leftarrow \{0\}$
5: Mark $\psi_L$ as used
6: for $i = 2 : M$ do
7:    for $j = 1 : T_{max}$ do
8:       $v_{min} \leftarrow 0$
9:       $p_{min} \leftarrow \{\}$
10:      for $p \in \Psi, p$ is not marked as used do
11:         if $p \notin sts[i, j - p.len].p$ then
12:            $l \leftarrow Loss\,(p, j)$
13:            if $l \leq v_{min}$ then
14:               $v_{min} \leftarrow l$
15:               $p_{min} \leftarrow p$
16:            end if
17:         end if
18:      end for
19:      if $p_{min}$ is empty then
20:         $sts[i, j] \leftarrow sts[i, j - 1]$
21:      else
22:         $sts[i, j].p \leftarrow sts[i, j - p_{min}.len].p \cup p_{min}$
23:         $sts[i, j].s \leftarrow sts[i, j - p_{min}.len].s \cup j - p_{min}.len$
24:      end if
25:   end for
26:   Mark all elements in $sts[i, T_{max}].p$ as used
27: end for
```

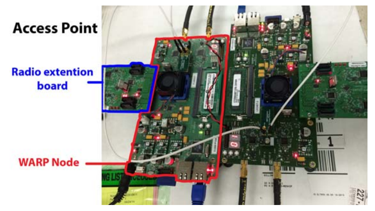
To get $m[W_{max}]$, we can iterate from $m[0]$, adding unit weight at each step and store $m[\omega]$, until we reach W_{max}.

As illustrated before, such simplistic solution does not fit for MU-MIMO padding problem. The aforementioned critical factor of inter-stream interference is not taken it into account. To address the problem, we re-design the above property functions. In particular, we define $\lambda_m[\tau]$ as maximum achievable throughput of stream m at time limit τ. We define $\xi[\tau]$ as maximum achievable throughput of all stream at time limit τ. Note that $\lambda_m[\tau]$ is the aggregation of values of frames in it while in $\xi[\tau]$ inter-stream interference is considered. Thus the $\xi[\tau]$ is the sum of $\lambda_m[\tau]$ in all M streams minus inter-stream interference among them before τ:

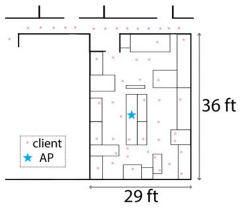
$$\xi[\tau] = \sum_{m=1}^{M} \lambda_m[\tau] - \sum_{m=1}^{M} \sum_{\psi_i \in \Delta_m} Loss_m\,(\psi_i, \tau) \tag{10}$$

where t_i is start time of ψ_i. In order to maximize $\xi[T_{max}]$, we need not only maximize the throughput of each stream, *i.e.*, the first term in (10), but also take interference, *i.e.*, the second term, into account so that it would not offset throughput of frames. The maximization of the first term can be addressed by dynamic programming solution for conventional knapsack problem. As for the second term, we need to add consideration to the maximization function in property functions so that when a frame is selected for $\lambda_m[\tau]$, instead of throughput of stream m alone, the combination of throughput of streams and throughput loss caused by the frame is maximized, *i.e.*,

$$\begin{aligned} \lambda_m[\tau] = max_{l_p \leq \tau, \psi_p \notin m}(\lambda_m[\tau - l_p] + \gamma_i l_i \\ - \sum_{i=1, i \neq m}^{M} Loss_i(\psi_p, \tau - l_i)) \end{aligned} \tag{11}$$

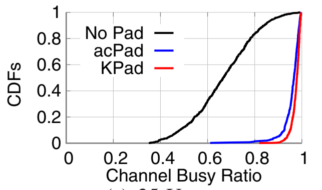


(a) KPad prototype with WARP

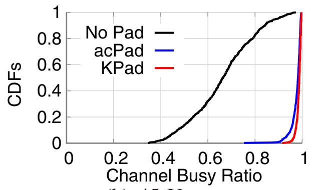


(b) Testbed Overview

Fig. 3. Experimental Setup



(a) 25 Users



(b) 45 Users

Fig. 4. CDF of CBR with Different Number of Users

(11) describes the principle for frame selection. However unlike (9) one cannot derive executable algorithm to compute $\xi[T_{max}]$ from (11). The reason lies in that, to compute throughput loss for streams other than i, we have to acquire the combinations and schedule of those streams, which in turn are partially determined on stream i.

Due to this, a proper approximation is required to achieve sub-optimal result. We noticed that in (4), the computing of SNR loss for all streams are cumulative. Thus we can correctly derive throughput loss of padding frame in last stream without knowing throughput loss of frames in previous streams. Inspired by this, we use a greedy approximation of selecting and scheduling frames for one stream at a time, while leaving unreached streams temporarily empty when computing throughput loss. Formally, we can rewrite (9) as follow:

$$\begin{aligned}\lambda_m[\tau] = max_{l_p \le \tau, \psi_p \notin m}(\lambda_m[\tau - l_p] + \gamma_i l_i \\ - \sum_{i=1}^{m-1} Loss_i(\psi_p, \tau - l_i))\end{aligned} \quad (12)$$

where the limit of sum is modified to $[1, m-1]$. Now we can iterate from $\xi_1[0]$ like conventional knapsack solution, since selection of frames determines on their influences only on those that are padded beforehand.

We design KPad algorithm based on aforementioned property functions. KPad algorithm (as given in Algorithm 2) starts with finding the longest frame in Ψ, and then adds it to the first stream and sets its length as weight limit T_{max}. In the following iterations, KPad selects the frame that maximizes $\lambda_m[\tau]$ according to (12) for $m \le M$ and every $\tau \le T_{max}$. Note that for the calculation of throughput loss, we store frame selection and scheduling corresponding to each frame that is selected for $\lambda_m[\tau]$. The final frame selection and scheduling that achieves maximum $\xi[T_{max}]$ can be found in $sts[m, T_{max}]$ for $m \le M$. We note that such greedy approximation still outperforms conventional No Pad the state-of-the-art AcPad, as validated in Sec. IV.

E. *Computational complexity*

KPad algorithm possesses a time complexity of $O(MN^2T_{max})$, where T_{max} stands for the maximum frame length. Note that such polynomial complexity can be handled by modern AP or BS with enough computational resource [23]. Moreover, the running time can be further reduced by dividing lengths of frames by their greatest common divisor, but it is out of the scope of this paper.

IV. Evaluation

We have built a prototype of KPad based on WARP [9], a fully programmable software-radio platform, and adopt a trace-driven emulation approach to evaluate Kpad. In particular, we first collect CSI at 50 nodes, by moving a WARP node (shown in Fig. 3(a)) to 50 different locations in a typical office in Fig. 3(b). In each location, we measure CSI between the client user to every antenna of AP. All traces are collected on a 2.4GHz channel unused and non-overlapping with ambient wireless devices. Then we feed the trace into a simulator, which implements all modules of KPad, including SNR, loss computation, and frame padding schedule.

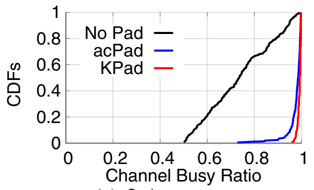
For performance comparison, we have also implemented the following two designs: *(i) No Pad:* As the benchmark baseline, No pad allows only one user for each stream in each round MU-MIMO transmission, which is the de-facto in existing MU-MIMO system. To ensure users with the strongest orthogonality to be selected, we implement the scalable Signpost algorithm proposed recently [12]. *(ii) AcPad:* As the state-of-the-art, Acpad designs a hybrid padding mechanisms. It first employs *padding with re-precoding* to protect the ongoing frames by re-allocating power, and then uses *SINR-based padding* to select users that can obtain a high SINR.

A. *Channel Busy Ratio*

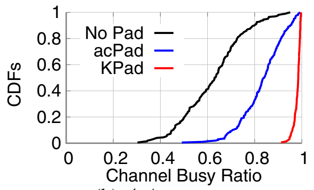
We first evaluate whether KPad can stuff all channel idle time when confronting packet-size diversity, by investigating channel busy ratio (CBR), which is defined as $\sum_{i=1}^{M} T_i^{busy} / (NT_{max})$, where T_i^{busy} is the length of channel busy time in i-th stream and T_{max} is the length of longest stream, *i.e.*, master stream. For experiment comprehensiveness, we vary the number of antennas on the AP from 2 to 4, and the number of client users from 5 to 45. We plot the CDF of CBR in Fig. 5 and Fig. 4, from which we have two main observations: *(i)* KPad can achieve very high CBR in both scenarios, *i.e.*, over 99% in average, irrespective of antenna settings on AP. CBR of AcPad, however, decreases from 97% to 80% while the number of antennas increases from 2 to 4. We also notice that CBR of No Pad is only 76% under 2-antenna-AP case, and becomes much worse for 4-antenna AP, 67% in particular. This corroborates the finding in [8]: For No Pad and acPad, the more streams exist, the higher possibility that any stream other than master stream cannot fully utilize the channel. *(ii)* CBR of KPad remains high as the number of users increases. In particular, the CBR is over 99% for both 25-users and 45-users cases.

B. *Throughput vs. number of antennas on the AP*

In this experiment, we fix the number of users to 45 and vary the number of antennas on the AP from 2 to 4. Fig. 6(a) plots the average throughput comparisons. We can observe that KPad performs better when the number of antennas is smaller. For instance, the throughput gain of KPad increases from $1.21\times$ to $1.42\times$ over acPad and $1.32\times$ to $1.55\times$ over No

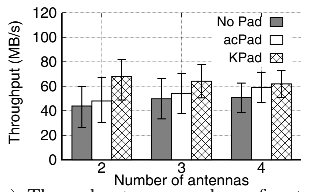


(a) 2 Antennas

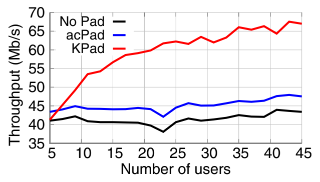


(b) 4 Antennas

Fig. 5. CDF of CBR with Different Number of Antennas



(a) Throughput vs. number of antennas



(b) Throughput vs. number of users

Fig. 6. Throughput vs. different scenarios

Pad, respectively, when the number of antennas decreases from 3 to 2. The reason is that when number of users is fixed, fewer antennas indicate more opportunities to pad high-throughput frames, which contribute more throughput to the system.

C. Throughput vs. number of client users

In this part, we fix the number of antennas to 2 and vary the number of users (*i.e.* N) from 5 to 45. Fig. 6(b) describes the average throughput of three comparing methods, respectively, from which we have two main observations: *(i)* KPad has the desirable feature of high scalability, *i.e.*, its advantage becomes more profound as the number of users increases. In particular, while KPad's throughput is a little bit lower when $N \leq 7$, KPad's throughput gain grow rapidly. When N reaches 45, KPad's throughput is $1.42\times$ and $1.55\times$ over acPad and No Pad, respectively. The observation shows that KPad is highly beneficial in crowd scenarios with large-scale client users. *(ii)* Not surprisingly, even the initial concurrent users selected by No pad are with the strongest orthogonality, No Pad has the least throughput, which again demonstrates the critical role of frame padding in MU-MIMO systems.

V. Conclusion

In this paper, we propose KPad, a model-driven frame padding method that can maximize the channel utilization and lead to optimal throughput. In contrast with previous heuristic approach, we formulate the frame padding problem as a multi-stream knapsack model, and derive the optimal padding schedule using novel designs. Trace-driven evaluation shows that KPad's throughput gain can be up to 42% compared with the state-of-the-art. In our future work, we plan to extend KPad to more general scenarios such as distributed net-MIMO networks spanning across multiple contention domains.

VI. Acknowledgment

The work reported was supported in part by the National Natural Science Foundation of China (No.61772084, No.61720106007, No. 61332005 and No. 61532012).

References

[1] M. S. Afaqui, E. Garcia-Villegas and E. Lopez-Aguilera, "IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi," *IEEE Wireless Communications*, vol.3, no. 24, pp. 130-137, 2017.
[2] A. B. Flores, S. Quadri and E. W. Knightly, "A Scalable Multi-User Uplink for Wi-Fi," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.
[3] E. H. Ong, J. Kneckt, O. Alanen, Z. Chang, T. Huovinen and T. Nihtil, "IEEE 802.11ac: Enhancements for very high throughput WLANs," in *IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2011.
[4] S. Sur, L. Pefkianakis, X. Zhang and K. Kim, "Practical MU-MIMO User Selection on 802.11ac Commodity Networks," in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (Mobicom)*, 2016.
[5] T.-W. Kuo, K.-C. Lee, K. C.-J. Lin and M.-J. Tsai, "Leader-Contention-Based User Matching for 802.11 Multiuser MIMO Networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 8, pp. 4389-4400, 2014.
[6] A. Narendra, J. Lee, S.-J. Lee, and E. W. Knightly,"Mode and User Selection for Multi-User MIMO WLANs without CSI," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.
[7] 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 *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom)*, 2012.
[8] C. H. Lin, Y. T. Chen, K. C. J. Lin and W. T. Chen, "acPad: Enhancing channel utilization for 802.11ac using packet padding," in *IEEE Conference on Computer Communications (INFOCOM)*, 2017.
[9] Rice University, "Wireless Open-Access Research Platform," http://warp.rice.edu/trac/wiki, Jul. 2014.
[10] T. Yoo, N. Jindal and A. Goldsmith, "Multi-Antenna Downlink Channels with Limited Feedback and User Selection," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 7, pp. 1478-1491, 2007.
[11] H. Taub and D. L. Schilling, *Principles of communication systems*, McGraw-Hill Higher Education, 1986.
[12] A. Zhou, T. Wei, X. Zhang, M. Liu and Z. Li, "Signpost: Scalable MU- MIMO Signaling with Zero CSI Feedback," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2015.
[13] R. Sinha, C. Papadopoulos and J. Heidemann, "Internet Packet Size Distributions: Some Observations," *USC/Information Sciences Institute, Tech. Rep.*, ISI-TR-2007-643, 2007.
[14] N. Sarrar, G. Maier, B. Ager, R. Sommer and S. Uhlig, "Investigating IPv6 Traffic," in *Proceedings of the 13th International Conference on Passive and Active Measurement (PAM)*, pp. 11-20, 2012.
[15] 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.
[16] S. Martello, D. Pisinger and P. Toth, "Dynamic Programming and Strong Bounds for the 01 Knapsack Problem," *Management Science* vol. 45, no. 3, pp. 414-424, 1999.
[17] D. Bertsimas and R. Demir, "An approximate dynamic programming approach to multidimensional knapsack problems," *Management Science*, vol. 48, no. 4, pp. 550-565, 2002.
[18] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM Journal on Computing*, vol. 35, no. 3, pp. 713-728, 2005.
[19] E. Perahia and M. X. Gong, "Gigabit wireless LANs: an overview of IEEE 802.11 ac and 802.11 ad," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 3, pp. 23-33, 2011.
[20] J. Ketonen, J. Markku and J.R. Cavallaro, "PerformanceComplexity comparison of receivers for a LTE MIMOOFDM system," *IEEE transactions on signal processing*, vol. 58, no. 6, pp. 3360-3372, 2010.
[21] E. Telatar, "Capacity of Multiantenna Gaussian Channels," *Transactions on Emerging Telecommunications Technologies*, vol. 10, no. 6, pp. 585-595, 1999.
[22] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: a survey," *Naval Research Logistics (NRL)*, vol. 22, no. 1, pp. 127-144, 1975.
[23] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang and L. Zhong, "Argos: Practical many-antenna base stations," in *Proceedings of the 18th annual international conference on Mobile computing and networking. (Mombicom)*, 2012.