PNOFA: Practical, Near-Optimal Frame Aggregation for Modern 802.11 Networks

Ali Abedi
Cheriton School of Computer Science
University of Waterloo
ali.abedi@uwaterloo.ca

Tim Brecht
Cheriton School of Computer Science
University of Waterloo
brecht@cs.uwaterloo.ca

Omid Abari
Computer Science Department
UCLA
omid@cs.ucla.edu

ABSTRACT

MAC-layer frame aggregation has significantly improved the efficiency of IEEE 802.11n/ac networks by placing multiple MAC-layer data units in a large PHY-layer frame. In this paper, we focus on finding the optimal length of an *Aggregated MAC Protocol Data Unit (A-MPDU)* in order to maximize throughput. This problem has proved to be extremely challenging because of the chain of dependencies between consecutive A-MPDUs due to software retransmissions and because error rates can be higher in the later part of the A-MPDU.

In this paper we develop a model of A-MPDU frame aggregation and use it to design a *statistically optimal* algorithm. We then develop a standard compliant, Practical, Near-Optimal Frame Aggregation algorithm (PNOFA). Our trace-based evaluation shows that across a variety of devices and scenarios PNOFA outperforms existing state-of-the-art algorithms and obtains throughputs that are within 97% of those obtained using the statistically optimal algorithm. Furthermore, we implement PNOFA on an 802.11ac Google Wifi access point. We find that when compared with the proprietary frame aggregation algorithm in the Qualcomm IPQ 4019 chipset's firmware, PNOFA increases average throughput by 17% in the scenarios tested.

CCS CONCEPTS

• **Networks → Wireless local area networks**; **Wireless access points, base stations and infrastructure**; **Network performance evaluation**; • **Hardware → Wireless devices**.

KEYWORDS

WiFi; 802.11; frame aggregation; aggregated MAC protocol data unit; A-MPDU length; optimal algorithms; performance evaluation;

ACM Reference Format:
Ali Abedi, Tim Brecht, and Omid Abari. 2020. PNOFA: Practical, Near-Optimal Frame Aggregation for Modern 802.11 Networks. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3416010.3423215

1 INTRODUCTION

The 802.11n and 802.11ac standards are able to achieve high physical-layer bit rates by adding new features such as MIMO and channel bonding. However, without frame aggregation, the MAC-layer throughput does not surpass 50 Mbps regardless of the physical layer bit rate, due to overheads such as the inter-frame spacing and acknowledgments. To reduce this overhead, a MAC-layer frame aggregation mechanism has been introduced starting with the 802.11n standard that aggregates multiple MAC-protocol data units (MPDUs) into one larger aggregated MPDU (A-MPDU). As a result, rather than sending multiple small frames, which require their own backoff, inter-frame spacing, and acknowledgments, one larger frame containing multiple subframes is transmitted instead.

Maximizing throughput during transmission requires optimizing the number of subframes (MPDUs) for the current channel conditions. If only a small number of MPDUs are aggregated, throughput may suffer. However, if too many MPDUs are aggregated and many of them are not received successfully, throughput may also be lower than possible. One key factor that has to be considered by frame aggregation algorithms are channel compensation (or correction) limitations. Because channel estimation is done only once using the preamble of each A-MPDU, it may not be as accurate for MPDUs that are farther from the beginning of the A-MPDU. As a result, MPDUs near the end of an A-MPDU may be more likely to fail than those near the beginning especially in environments with mobility.

Figure 1 plots the *MPDU Delivery Ratio* (MDR) for each subframe at different positions in an A-MPDU for an instance in time. Index 1 is the first MPDU in the frame and index 32 is the last. The figure shows that as the MPDU index increases, the MDR generally decreases (i.e., there is a lower probability of successful delivery). In this example, peak throughput is obtained with a size of 7. Since the MPDU delivery ratios vary over time and change with several factors including speed of movement and transmission rate, the optimal A-MPDU size also changes over time. While algorithms

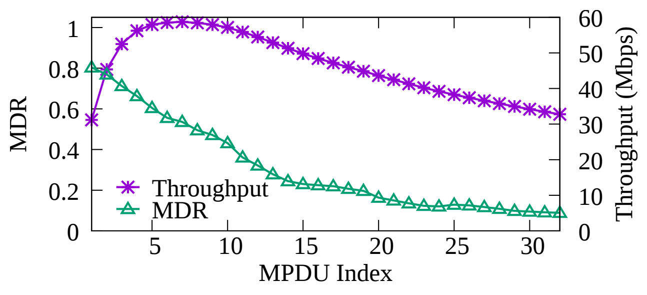


Figure 1: Impact of A-MPDU size on throughput

to adjust A-MPDU lengths to deal with this phenomenon have been proposed [3] [4], ours is the first work to derive the optimal algorithm.

In this paper, we develop an analytic model that can be used to determine the statistically optimal number of frames to aggregate to maximize throughput. Our model is the first to consider the impact of the combination of software retransmission and the block ACK mechanism, which creates dependencies between consecutive A-MPDUs. These dependencies make frame aggregation more challenging because determining the optimal A-MPDU length for a specific frame depends on the fate of each MPDU in the next frame. However, the optimal A-MPDU length of that frame depends on the fate of each MPDU in the frame after that. These interdependencies repeat for a chain of A-MPDUs.

We design a Practical, Near-Optimal Frame Aggregation algorithm (PNOFA) for modern 802.11 networks . Our trace-driven evaluations show that PNOFA outperforms state-of-the-art algorithms and achieves throughputs very close to the statistically optimal algorithm across a wide variety of scenarios and devices. We also implement PNOFA on a Google Wifi access point [10] that utilizes the modern Qualcomm IPQ 4019 [15] system-on-chip that supports Wave 2 802.11ac features. This chipset is used in at least 82 other access points from major manufacturers [5]. This and other WiFi chipsets used in modern 802.11ac devices typically implement frame aggregation in the chipset's closed-source firmware. However, because PNOFA requires relatively little information and support from the underlying device we are able to implement PNOFA as a user-space process. Such an implementation is not possible for existing algorithms because they would require modifications to closed-source firmware. Our results show that PNOFA can provide throughput improvements over the proprietary frame aggregation algorithm in the Qualcomm IPQ 4019 chipset. Our contributions in this paper are:

- We determine *statistically optimal* A-MPDU sizes by studying dependencies between A-MPDUs and modelling A-MPDU frame aggregation. Then, we develop a practical frame aggregation algorithm called Practical, Near-Optimal Frame Aggregation (PNOFA). In contrast with the statistically optimal algorithm PNOFA does not require a priori knowledge of the MPDU delivery ratios.
- Using trace-driven evaluation, we show that average throughput obtained using PNOFA is within 97% of that obtained using the statistically optimal algorithm across different scenarios and devices, and that state-of-the-art approaches do not perform nearly as well.
- The design of PNOFA enables user-space implementation on devices with closed-source firmware. Our experimental results show that for the scenarios examined, PNOFA improves UDP and TCP throughput when compared to the proprietary Qualcomm IPQ 4019 chipset's frame aggregation algorithm by 17% and 13%.

The insights obtained from designing the statistically optimal frame aggregation algorithm have helped us to develop a statistically optimal rate adaptation algorithm in subsequent research [11]. That work also uses PNOFA in conjunction with a neural network model-based rate adaptation algorithm to signicantly improve WiFi throughput.

2 AGGREGATION CHALLENGES

In this section, we explain why A-MPDU frame aggregation is an important, interesting, and challenging problem.

2.1 Channel Compensation Limitations

MAC-layer frame aggregation significantly improves the throughput of 802.11 networks, however it introduces some challenges in the operation of these networks. The first challenge caused by A-MPDU frame aggregation is due to limitations in channel compensation (correction). Although the MPDUs in an A-MPDU have their own MAC header and CRC, Byeon et al. [3] have shown that the frame error rate (FER) of MPDUs depend on their location in an A-MPDU, especially in mobile environments. Since timing and frequency calibration are done only once using the preamble of an A-MPDU, if channel conditions change during the reception of the A-MPDU, the initial channel estimations are no longer valid rendering channel correction ineffective. Consequently, subframes at the end of an A-MPDU are more likely to fail. A direct consequence of this finding is that the throughput is not necessarily maximized by adding as many MPDUs as possible in an A-MPDU.

2.2 Dependencies Between A-MPDUs

The second challenge in determining the best A-MPDU length is caused by the block acknowledgment (block ACK) mechanism and software retransmission. To our knowledge, this is the first work that considers dependencies between A-MPDUs in finding the optimal length of an A-MPDU. The 802.11n and 802.11ac standards support block acknowledgment in which the receiver selectively acknowledges receiving multiple subframes by transmitting a single acknowledgment. This mechanism improves efficiency since only one block ACK is transmitted instead of one acknowledgment per subframe. In addition, the block ACK determines which MPDUs were not received and have to be retransmitted. This process is called *software retransmission.*

In 802.11n and ac protocols, a block acknowledgment, consisting of a *starting sequence number* and a 64-bit bitmap, is transmitted back to the sender, where each bit acknowledges a subframe with the corresponding offset from the starting sequence number. With this block ACK mechanism, only MPDUs with sequence numbers that are within the 64-MPDU window can be transmitted. This can potentially limit the number of frames that the sender can aggregate in an A-MPDU [14]. For example, suppose that the sender transmits 64 frames with sequence numbers 1000 to 1063. If the MPDU with sequence number 1000 is lost, the sender cannot aggregate new MPDUs in the next A-MPDU because their sequence number will be outside the current block ACK window (BAW). Therefore, even if all other transmitted MPDUs are successfully received, the sender cannot transmit new subframes in the next A-MPDU(s) until the MPDU with sequence number 1000 is acknowledged or a software retry limit is reached. The chain of dependencies between consecutive A-MPDUs, caused by block ACK window advancements and software retransmission, significantly complicates how to determine the optimal number of frames to aggregate.

3 RELATED WORK

Frame aggregation in 802.11 networks has attracted a lot of attention due to the significant efficiency improvements promised by this feature. As a result, different categories of frame aggregation algorithms have been studied in recent years. Frame Aggregation Schedulers [18] [8] are concerned with optimizing the transmissions under unsaturated link conditions. The second category of frame aggregation algorithms [12] [17] [16] is concerned with finding the optimal length of an Aggregated MAC Service Data Unit (A-MSDU). Our work is complementary to frame aggregation schedulers and A-MSDU frame aggregation algorithms so we do not consider them in this work.

This paper focuses on finding the optimal number of subframes to aggregate in an A-MPDU. The most closely related studies to our work are MoFA [3] and STRALE [4] frame aggregation algorithms. As a result, we implement these algorithms and compare their performance to that of our proposed algorithm.

To overcome channel compensation limitations, Byeon et al. [3] propose a frame aggregation algorithm called MoFA that adjusts the length of A-MPDUs in a attempt to maximize throughput. This algorithm tries to distinguish stationary and mobile channel conditions and reduces the length of A-MPDUs dynamically under mobile conditions. This algorithm compares the average frame error rates (FER) of the first and second half of an A-MPDU. If the difference between the average FERs is higher than 20% the environment is considered mobile and the A-MPDU length is reduced. If this difference is less than 20% for multiple consecutive A-MPDUs, the algorithms assumes the channel is ready to support longer A-MPDUs. Therefore, MoFA increases the number of subframes in the next A-MPDUs.

Byeon et al. [4] propose another algorithm called STRALE that tries to jointly optimize frame aggregation and rate adaptation. The authors find that under certain conditions, in addition to limiting the length of an A-MPDU, reducing the transmission rate can also help to cope with the channel compensation limitations. They utilize this finding to implement a heuristic that considers two options to cope with channel compensation limitations. STRALE estimates the expected throughput for two transmission settings: (1) the same transmission rate with a smaller A-MPDU size (2) a lower transmission rate (i.e., one-level lower MCS index) with the same A-MPDU size. If option (1) results in higher expected throughput, the A-MPDU size is decreased based on the fate of the previous A-MPDUs. Otherwise, the rate adaptation algorithm is instructed to override its chosen transmission rate by one MCS index. Therefore, in some cases the frame aggregation algorithm impacts the rate adaptation algorithm.

Byeon et al. [3] [4] implement and evaluate MoFA and STRALE on an 802.11n platform. In addition, STRALE was also evaluated using an 802.11ac simulator, since it could not be implemented due to limitations in the ath10k driver [4]. We study these algorithms using trace-driven evaluation (802.11n) and find that PNOFA outperforms them in a variety of scenarios. We also implement PNOFA as a user-space process on an 802.11ac Google Wifi access point. Such an implementation is not possible for these algorithms because they would require modifications to closed-source firmware.

4 OPTIMIZING A-MPDU LENGTH

The goal of this section is to find the optimal length of an A-MPDU given the expected frame error rates of each subframe within an A-MPDU (for the given transmission rate). We model A-MPDU frame aggregation in 802.11 networks in order to optimize throughput, first without considering the dependencies between consecutive A-MPDUs. Then, we examine how these dependencies may change frame aggregation decisions.

4.1 Modeling Optimal A-MPDU Length

We model the transmission time of an A-MPDU with n subframes, $T_{\text{A-MPDU}}(n)$, as follows:

$$T_{\text{A-MPDU}}(n) = \theta + \lambda n \tag{1}$$

where θ includes all overheads associated with transmitting an A-MPDU, including PHY header, DIFS, SIFS, backoffs, and block ACK as illustrated in Figure 2. λ is the transmission time of one MPDU and is defined as $\lambda = \frac{B}{R}$, where B is the number of bits in an MPDU and R is the transmission rate in bits per second.

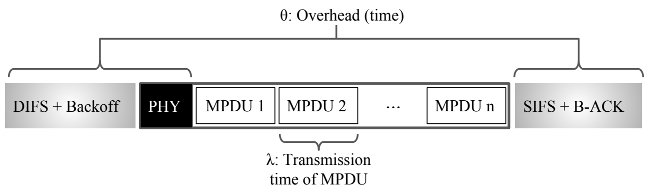


Figure 2: 802.11 A-MPDU and transmission overheads

The expected number of successful MPDUs, $N_s(n)$, in an A-MPDU with n subframes is:

$$N_s(n) = \sum_{i=1}^{n} MDR(i) \tag{2}$$

where $MDR(i)$ is the delivery ratio of MPDU index i (i.e., the probability of the successful transmission of i'th MPDU in an aggregated frame).

Finally, we compute the expected throughput when n subframes are aggregated in an A-MPDU. Since the transmission time of an A-MPDU, including all overheads such as receiving a block ACK, is $\theta + \lambda n$, we can transmit $1/(\theta + \lambda n)$ A-MPDUs of size n per second. The number of successful MPDUs in each aggregated frame is $N_s(n)$, therefore the expected throughput will be:

$$Tput(n) = \frac{BN_s(n)}{T_{\text{A-MPDU}}(n)} = \frac{B\sum_{i=1}^{n} MDR(i)}{\theta + \lambda . n} \tag{3}$$

To maximize throughout, we need to find the value of n that maximizes Equation 3. The derivative of Equation 3 with respect to n does not have a general closed-form and depends on the $MDR(i)$ function. However, the optimal length of an A-MPDU can be calculated numerically by computing the expected throughput for all possible lengths of an A-MPDU.

MoFA [3] and STRALE [4] use a similar formula to compute what the best length would have been for past A-MPDUs. Then they

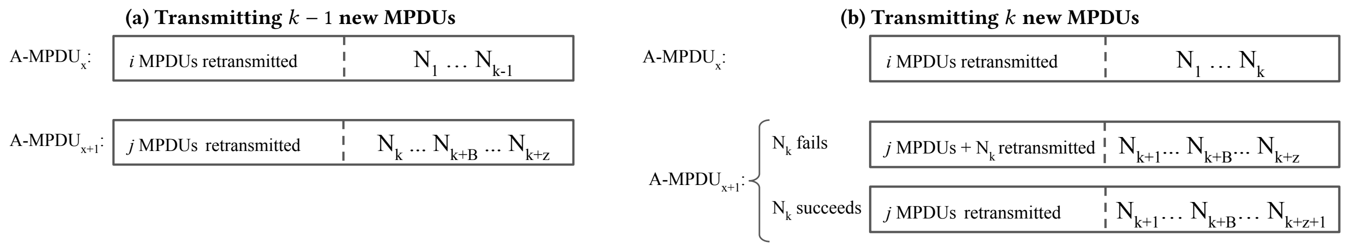


Figure 3: Transmitting $k-1$ versus k new subframes in an A-MPDU

utilize this metric in their proposed heuristic to decrease the length of subsequent A-MPDUs if necessary. They also propose a separate heuristic for increasing the length of subsequent A-MPDUs that is substantially different from the technique we employ. In contrast, we provide the expected delivery ratios (MDRs) to Equation 3 to compute the optimal length for the next A-MPDU.

4.2 Dependencies Between A-MPDUs

In Section 4.1, we computed the optimal length of an aggregated frame without considering the dependencies between consecutive A-MPDUs. As explained in Section 2.2, one might think that because of the limitations caused by block ACKs and software retransmissions, it might be beneficial to transmit shorter A-MPDUs than computed in the previous section. However, we now show that these dependencies do not affect the optimal length of A-MPDUs.

Theorem 1. *An algorithm that optimizes the length of an A-MPDU without considering the following A-MPDUs achieves the statistically optimal throughput.*

Before formally proving this theorem, we describe an example that provides some intuition behind the proof. Suppose that based on Equation 3 the optimal length of an A-MPDU is calculated to be 64. In this extreme example imagine that all MPDUs are delivered successfully except the first MPDU. In this case, the software retransmission mechanism reschedules the first MPDU in the next A-MPDU and the block ACK window can not be advanced. Therefore, no new subframes can be sent, and the length of the first and second A-MPDUs will be 64 and 1, respectively. Recall from Figure 1 that throughput of A-MPDUs with only one subframe will be very low. As a result, we now consider the question of whether or not limiting the first A-MPDU results in higher throughput. For example, does sending 32 and 33-subframe A-MPDUs (the second frame carries the retransmission of the first MPDU and 32 new MPDUs) result in higher throughput? To compare the two cases we calculate their total transmission times. Recall that θ denotes all overheads when transmitting an A-MPDU. In both cases, since two A-MPDUs are transmitted, the total overhead will be 2θ. The transmission time of all 65 (i.e., 64+1 or 32+33) MPDUs will be 65λ. Therefore, the total transmission time in both cases will be $2\theta + 65\lambda$. As a result, in this worst case there is no gain in throughput by limiting the first A-MPDU. We now provide a proof of Theorem 1.

Proof. We consider different possibilities when creating two consecutive A-MPDUs x and x+1. As illustrated in Figures 3, when creating A-MPDU$_x$, we need to schedule $i \geq 0$ MPDUs for retransmission, because they failed in previous A-MPDUs. Assume that the optimal length of A-MPDU$_x$ without considering the dependency between A-MPDUs is $i+k$, therefore, we can add k new MPDUs to A-MPDU$_x$, denoted $N_1 \ldots N_k$. To prove the theorem, we compare the possible outcomes of sending $i+k$ or $i+k-1$ subframes in an A-MPDU.

Aggregating $i+k-1$ MPDUs (Figure 3a):
Assume that we add $k-1$ new MPDUs to A-MPDU$_x$. When creating A-MPDU$_{x+1}$, we need to retransmit $j \geq 0$ MPDUs. The length of an A-MPDU can be limited by, the block ACK window, the length suggested by Equation 3 and possibly the maximum size of the frame. Since the last transmitted MPDU was N_{k-1}, the new MPDUs that can be transmitted in A-MPDU$_{x+1}$ start from N_k. The last MPDU that can be aggregated because of the block ACK window is denoted by N_{k+B}. Similarly, the last aggregated MPDU based on Equation 3 is denoted by N_{k+z}. To illustrate a sample A-MPDU, in Figure 3a, N_{k+B} happens before N_{k+z} although these two imaginary MPDUs can have any order. Therefore, the last MPDU that actually can be aggregated is $min(N_{k+B}, N_{k+z})$.

Aggregating $i+k$ MPDUs (Figure 3b):
If k new MPDUs are aggregated, two outcomes are conceivable for the next A-MPDU (i.e., A-MPDU$_{x+1}$). If N_k succeeds, j MPDUs have to be scheduled for retransmission, otherwise, N_k must also be retransmitted. In these cases, the new MPDUs start from N_{k+1} since N_k was attempted in the last A-MPDU. Similar to the $i+k-1$ case, A-MPDU$_{x+1}$ is limited by the block ACK window or the length from Equation 3. In both cases (i.e., N_k fails or succeeds), if $B < z$, the last MPDU will be N_{k+B}. Note that, N_{k+B} refers to the frame with the highest sequence number in the block ACK window. As a result, the fate of N_k does not impose further restrictions on the BAW in this case. However, if $z < B$ the last MPDU will be N_{k+z} or N_{k+z+1} if N_k fails or succeeds, respectively. In this case (with $i+k$ MPDUs), including N_k in A-MPDU$_x$ creates an opportunity to possibly transmit one more packet when compared to the case where N_k is not included in A-MPDU$_x$ (Figure 3a). As a result, there is no throughput gain from aggregating fewer MPDUs than what Equation 3 suggests. This proof can be recursively applied on smaller A-MPDUs to show that smaller MPDUs do not provide higher throughout either. Hence, the best throughout is achieved when the length from Equation 3 is used. □

Note that aggregating more frames than what Equation 3 suggests does not provide higher throughput either. This is because the first A-MPDU is sub-optimal and it creates no opportunity for improvement in the following A-MPDUs.

4.3 Statistically Optimal Algorithm (SO)

We now use our findings from Sections 4.1 and 4.2 to define the Statistically Optimal A-MPDU aggregation algorithm (SO). For each A-MPDU, the SO algorithm calculates the number of MPDUs to aggregate based on Equation 3 for the given PHY rate and MPDU delivery ratios. Note that SO uses knowledge of the MDRs for future A-MPDUs (i.e., it is an offline algorithm). To implement SO in our trace-driven evaluation, we calculate the expected throughput for all A-MPDU sizes from 1 to 32 subframes[1] using Equation 3 and choose the length that achieves the highest throughput.

To provide some evidence that the statistically optimal algorithm is operating as expected, we conduct a trace-driven evaluation. In this evaluation, we compare the throughput obtained from this algorithm with that of a variety of constant A-MPDU length settings. The idea is that as channel conditions change, each different fixed length setting may produce the best throughput at different points in time. *However, SO should meet or exceed the throughput of these different lengths over the entire experiment.* Details of our performance evaluation methodology can be found in Section 6.1. Each constant length setting algorithm chooses the same aggregation length throughout the experiment. In this 10 minute experiment, we gradually increase the speed of movement from 0.5 m/s to 1.5 m/s. As illustrated in Figure 4 (top), the throughput of SO is always the same or higher than all constant configurations. We show the performance of a subset of all possible settings due to the limited space on this plot.

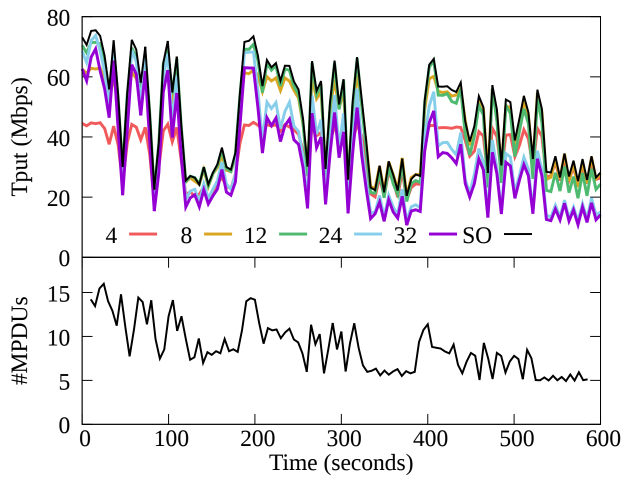


Figure 4: SO versus constant A-MPDU sizes

Figure 4 (bottom) shows the number subframes SO aggregates in an A-MPDU for the same experiment. In this experiment the optimal A-MPDU length constantly changes over relatively short periods of time. Changes in the optimal length are due to short term variations in speed and the movement of the device in a person's hand or pocket. In addition, we observe that the A-MPDU size generally decreases as the movement speed increases.

[1]Using the ath9k driver's optimization of creating a subsequent frame while the current A-MPDU is transmitted.

5 PNOFA

The statistically optimal algorithm (SO) utilizes Equation 3 and information about future transmissions to determine the optimal length of an A-MPDU. In order to derive a practical and standard compliant algorithm that approximates SO we had to overcome several practical challenges. We now describe how we address these challenges and present pseudocode for the implementation of PNOFA in Figure 5.

The first challenge is that MPDU delivery ratios are not known for the A-MPDU that is being aggregated for the next transmission. If the transmission rate is R, PNOFA estimates the MDRs using recent transmissions for that rate. Specifically, the average MDRs (for each rate) are computed over a specified window of time (i.e., AveragingWindow) [lines 5–6]. If rate R had not been used in the last window, the aggregation level is set to the maximum length [line 3]. We use the maximum because as shown in Section 4.1 if the failure probabilities for each MPDU are roughly equal this yields the best throughput. Alternatively, we could have considered rates that are similar to R (e.g., R-1 or R+1).

```
Input: Fate of MPDUs in the last AveragingWindow
Parameters: Transmission rate R
1  SAMPLES ← Fate of MPDUs transmitted at rate R in the last
     AveragingWindow;
2  if size(SAMPLES) == 0 then
3      A-MPDU-SIZE = MAX-MPDU(R);
4  else
5      for i in 1 to MAX-MPDU(R) do
6          MDR[i] ← Delivery ratio of MPDU i in SAMPLES;
7      end
8      OPT-SIZE ← Compute optimal A-MPDU size based on Eq. 3
         (using MDR and R as input);
9      A-MPDU-SIZE ← OPT-SIZE + $\frac{\text{ExtraMPDUsWindow}}{\lambda}$;
10     if A-MPDU-SIZE > MAX-MPDU(R) then
11         A-MPDU-SIZE = MAX-MPDU(R);
12     end
13 end
Output: A-MPDU-SIZE
14 Function MAX-MPDU(R) is
15     return max size of A-MPDU for rate R;
16 end
```

Figure 5: PNOFA algorithm

The second challenge is that if a maximum of n MPDUs have been recently aggregated, we do not have any information about the MDR of subframes $n+1$ and beyond. Therefore, if the length should be increased, the algorithm may not be able to properly adapt as no information is available about MPDU $n+1$ and later subframes. To address this challenge with very little overhead, PNOFA always adds a few additional MPDUs (beyond the value obtained from Equation 3). The number of additional MPDUs is determined by the ExtraMPDUsWindow and λ (the MPDU transmission time at rate R)[line 9]. This approach aggregates slightly more than the optimal number of MPDUs, in order to obtain a bit of information about the delivery ratios of slightly longer than optimal length A-MPDUs. If the optimal length should be increased the algorithm will have information about the MDRs of longer frames. In addition, aggregating only a few more MPDUs than the optimal decreases throughput only very slightly (if at all).

5.1 PNOFA Parameters

ExtraMPDUsWindow: We have empirically determined how many subframes to add to an A-MPDU beyond the value obtained from Equation 3. Because the time required to transmit an MPDU depends on the transmission rate, we use an approach where the number of additional subframes changes with the rate. The idea is that PNOFA adds x additional MPDUs for a rate R, where x is the number MPDUs that can be sent in time ExtraMPDUsWindow.

To find a good value to use for ExtraMPDUsWindow, we conduct several evaluations using different scenarios consisting of different devices and walking speeds (refer to Section 6.3 for details for each scenario). In the scenarios tested, a client device is carried with slow and normal walking speeds (approximately 1.0 and 1.4 meters per second, respectively) in a lab and office space. In S1, we carry a laptop equipped with an Intel AC 3160 802.11ac WiFi card at normal walking speed for 300 seconds. In S4 and S5, we carry a laptop equipped with a TL-WDN4200 802.11n and Archer T9UH 802.11ac WiFi cards, respectively. Both experiments are run for 400 seconds with a mix of slow and normal walking speeds.

Figure 6 shows results for three different scenarios, S1, S4 and S5 while plotting the average throughput obtained using different values of ExtraMPDUsWindow. This graph shows that if the window size is too small, not enough MPDUs are added and PNOFA does not have the necessary information to increase the A-MPDU size when channel conditions improve (e.g., movement speed decreases). The figure also shows that if the window size is too large, throughout decreases (due to the higher error rate of the later subframes in an A-MPDU). The best throughput is achieved when the window size is about 250 microseconds (which is used in our prototype implementation). To provide insight into the number of extra subframes that would be added with 250 microseconds, a physical rate of 144 Mbps allows for 3 extra MPDUs, while a rate of 72 Mbps allows for 1 extra MPDU.

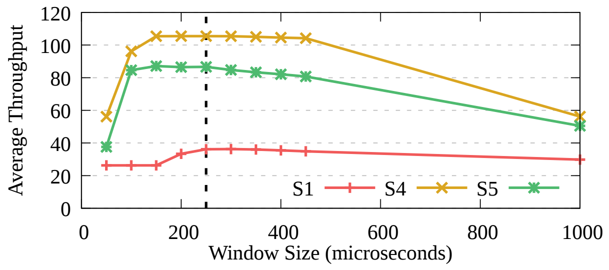


Figure 6: ExtraMPDUsWindow sizes and throughput

AveragingWindow: Similarly we have examined different values for the averaging window size under a variety of channel conditions (the results are not included here). We have found that a window size of 200 milliseconds works well in practice and that PNOFA was not very sensitive to this window size in the scenarios tested (obtaining peak throughput for values of up to one second). The reason is that the acceleration of a human body is quite limited when walking and therefore the speed of movement does not change significantly with windows of up to one second.

6 TRACE-DRIVEN EVALUATION

6.1 Methodology

As discussed previously, we could not implement existing frame aggregation algorithms such as MoFA and STRALE on modern 802.11ac platforms because of the closed-source firmware. In order to compare the performance of PNOFA with that of the state-of-the-art algorithms, we conduct trace-driven evaluations. We obtain and use T-SIMn, a trace-driven evaluation framework [1] that allows for highly accurate and fair comparisons under conditions that include mobility, WiFi and non-WiFi interference, and which use 2.4 and 5 GHz spectrums. Previous research [9] has demonstrated that T-SIMn is extremely realistic and highly accurate. Different algorithms can then be compared by replaying the same trace using each algorithm. In addition, using trace-driven evaluation allows us to implement the statistically optimal algorithm which requires a priori knowledge of the fate of future frames. We ported code from the publicly available STRALE implementation [4] to T-SIMn. We have also implemented MoFA as described by Byeon et al. [3]. Finally, the statistically optimal algorithm and PNOFA are implemented as described in Sections 4.3 and 5. We have also ported the Minstrel HT rate adaptation algorithm from Linux to T-SIMn. Minstrel HT is a sampling-based algorithm and is the default rate adaptation algorithm in the widely used Linux mac802.11 module [7].

In each scenario, we run an experiment to collect a trace of A-MPDU transmissions. The sending device (i.e., the access point) transmits every A-MPDU at a new rate. All rates are sampled in a round-robin fashion and this process continues for the duration of the experiment. The fate of all packets are recorded in the trace which allows the evaluation framework to determine the MPDU delivery ratios for all transmission rates over any window of time. During trace collection, A-MPDUs are transmitted using the maximum length to measure the MPDU delivery ratios of all indexes. T-SIMn uses these traces to evaluate frame aggregation algorithms.

6.2 Trace Collection Test Bed

We have created a small test bed for collecting traces. It is housed in lab and office spaces in a building on a university campus. Our access point is a desktop with a TP-Link TL-WDN4800 dual-band wireless N PCI-E adapter. We create an 802.11n AP using `Hostapd` [13] on this machine. This device uses a modified `ath9k` (Atheros) device driver that enables round-robin trace collection.

To diversify our experiments, we use a few different client (receiving) devices. In most experiments, we use a laptop configured to use a TP-Link TL-WDN4200 dual-band wireless 802.11n card or a TP-Link Archer T9UH dual-band wireless 802.11ac USB adapter We have also used a laptop equipped with 802.11ac Intel AC 3160 WiFi chipset as the client in some experiments. Since the goal of these frame aggregation algorithms is to maximize MAC-layer throughput we use `iperf` [6] to generate UDP traffic to determine the maximum throughput each algorithm can achieve.

Obtaining the maximum possible throughput is an important goal not only because it determines the maximum speed for one client device in the network, but it also impacts the network's overall throughput. For instance, if one or more clients are generating bursty traffic (as is the case for widely popular streaming video requests) it is important to maximize network throughput in order to minimize the transmission time of each chunk of bursty video traffic. Therefore, the performance of frame aggregation algorithms is critical in both saturated (e.g., large downloads) and bursty (e.g., video streaming) traffic scenarios.

6.3 Scenarios Studied

We conduct experiments using several scenarios to evaluate the performance of PNOFA and other competing frame aggregation algorithms. Each experiment is long enough for devices to experience a variety of channel conditions. Table 1 summaries all scenarios including MIMO configurations, channel bandwidths, frequency bands and the speed of mobility. In Scenario 1, a laptop equipped with Intel AC 3160 WiFi card is carried for 300 seconds at normal walking speed. Scenarios 2 through 6 start with 200 seconds at a low walking speed followed by 200 seconds of normal walking speed (i.e., mixed speed). Finally, in Scenario 7, a stationary experiment is conducted for 400 seconds.

S	Device	MIMO	BW	Band	Speed
1	Intel AC 3160	1x1	20	5	Normal
2	TL-WDN4200	1x1	20	5	Mix
3	TL-WDN4200	3x3	20	5	Mix
4	TL-WDN4200	3x3	40	5	Mix
5	Archer T9UH	3x3	20	5	Mix
6	Archer T9UH	3x3	20	2.4	Mix
7	TL-WDN4200	3x3	20	5	Zero

Table 1: Different scenarios (S). Mix: slow and normal

6.4 Performance Details for Scenario 1

We start by examining the throughput obtained using the different frame aggregation algorithms in Scenario 1 (S1). Figure 7 (top) shows the achieved throughput over time (averaged over 5 second intervals) for all algorithms and the statistically optimal algorithm (labeled SO). This graph shows that the throughput obtained using PNOFA is always close to SO despite continually changing channel conditions due to interference and mobility. Because `ath9k` is oblivious to the channel compensation problem it performs quite poorly. MoFA and STRALE provide higher throughput but nevertheless are still sometimes quite far from optimal.

To more easily see the difference between these algorithms and the statistically optimal algorithm we plot the CDF of the throughput relative to that of SO in Figure 7 (bottom). Specifically, This figure shows that the relative loss is bound by 8%, while the throughput of MoFA and STRALE deviates from optimal by up to 24% and 36%, respectively. Note that even occasional drops in throughput can significantly impact a user's quality of experience.

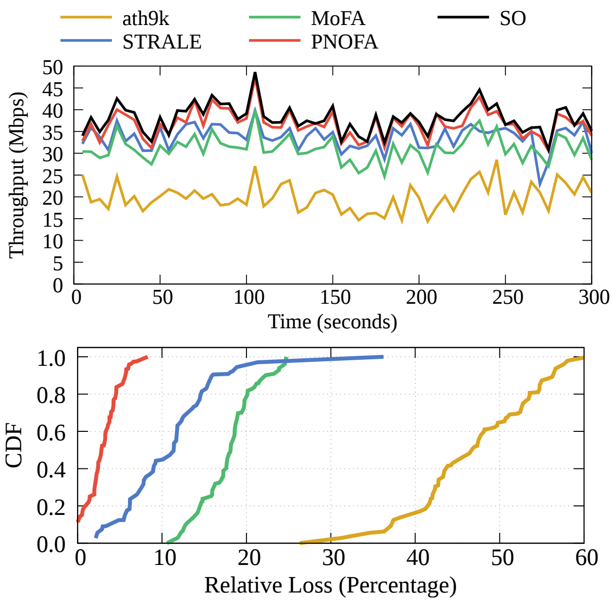


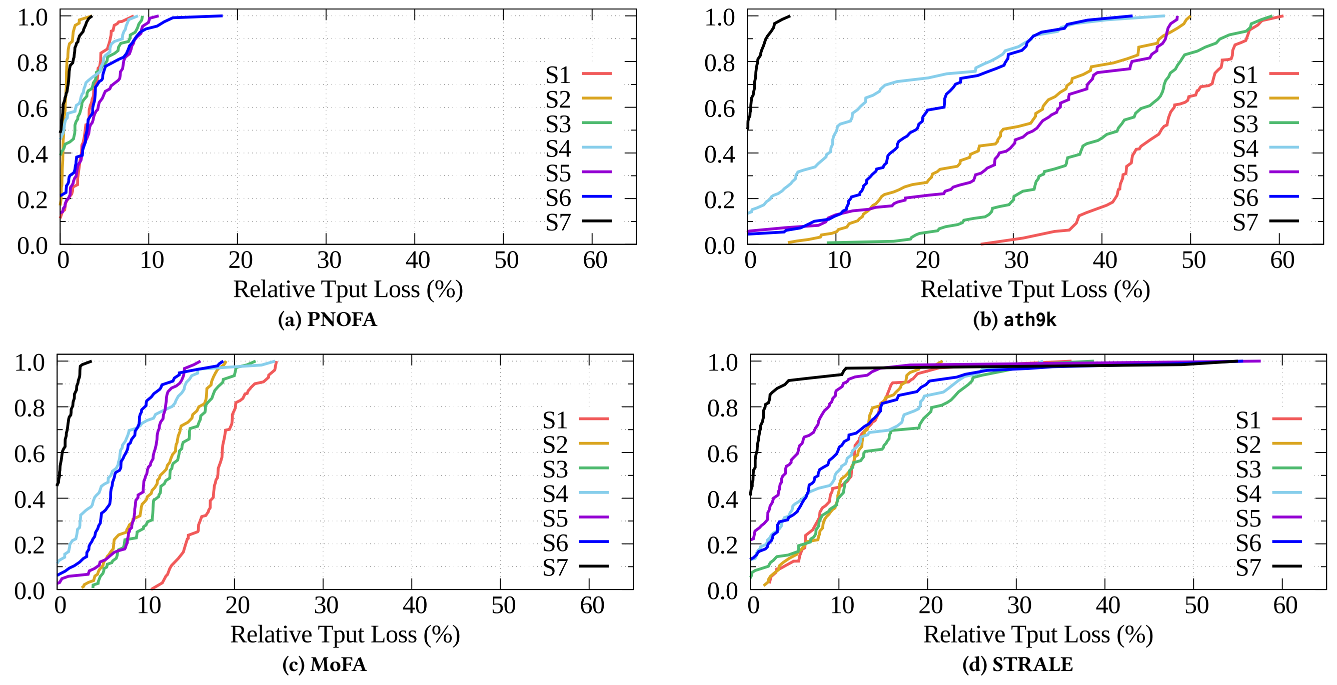
Figure 7: Throughput and CDF of relative loss

6.5 Performance with Different Scenarios

Figure 8 plots the performance of each algorithm using the CDF of the relative loss of throughput (when compared with the statistically optimal algorithm) for all 7 scenarios (described in Table 1). Figure 8a shows that the performance of PNOFA is very close to the statistically optimal algorithm across all scenarios. Specifically, the median and 90th percentile relative loss are less than 4% and 9% in all scenarios, respectively. These experiments demonstrate that despite the advantage the statistically optimal algorithm has in using a priori knowledge of actual frame error rates, PNOFA error rate estimations are sufficiently accurate to obtain excellent performance. In addition, PNOFA's mechanism of aggregating some extra MPDUs (beyond what is determined to be optimal) does not impose significant overhead.

Figure 8b shows results for the `ath9k` algorithm. It only performs well in scenario 7 in which the client device is always stationary, channel compensation is effective, and it does not impact the MPDU delivery ratios significantly. As a result, the aggregation level should not be limited and `ath9k` performs well. Figure 8b also nicely highlights the differences in the scenarios we have chosen because the `ath9k` algorithm is oblivious to the channel compensation problem. Consequently, throughputs for scenarios that are farther from optimal are more severely impacted by this problem.

Figures 8c and 8d show the performance of MoFA and STRALE, respectively. These graphs demonstrates that although their performance is reasonable, there is significant room for improvement relative to the statistically optimal algorithm. The median and 90th percentile throughput loss for MoFA is as high as 15% and 30%, respectively, and the median and 90th percentile loss for STRALE is as high as 15% and 25%, respectively. Figures 8c and 8d also reveal that the performance of MoFA and STRALE are not robust in term of their performance rankings. In some scenarios, such as S1 and S5, STRALE outperforms MoFA, while in other scenarios, such as S4, MoFA performs better. These differences can be explained by



(a) PNOFA

(b) `ath9k`

(c) MoFA

(d) STRALE

Figure 8: CDFs of throughput loss, relative to optimal

the fact that these algorithms are based on heuristics that are not necessarily accurate for all scenarios.

6.6 Analysis of Algorithm Differences

To better understand the differences between the behavior of PNOFA, STRALE and MoFA, Figure 9 shows several different metrics for each algorithm in S1. The comparison with STRALE is especially interesting because it is designed to change both the number of frames aggregated and the rate being used for transmission. Therefore, one might expect that STRALE could provide higher throughput than the other algorithms that do not modify the rate adaptation algorithm.

One can see that over the length of the experiment the mean throughput (Figure 9a) of PNOFA is consistently higher than STRALE which is almost always higher than MoFA. MoFA's throughput suffers because it is aggregating more frames (between about 5 and 7 frames) than the other algorithms (Figure 9b), which results in higher frame error rates (Figure 9c). On the other hand, STRALE's mean A-MPDU length is on average a bit lower than that of PNOFA and its frame error rate is relatively close to that of PNOFA. However, its mean PHY rate is slightly lower than that of PNOFA and MoFA (Figure 9d) because STRALE sometimes reduces the transmission rate to cope with the channel compensation problem. The shorter A-MPDUs combined with the lower mean PHY rate used by STRALE result in lower frame error rates and higher throughput than MoFA but the shorter frames and lower PHY rate result in lower throughput than PNOFA.

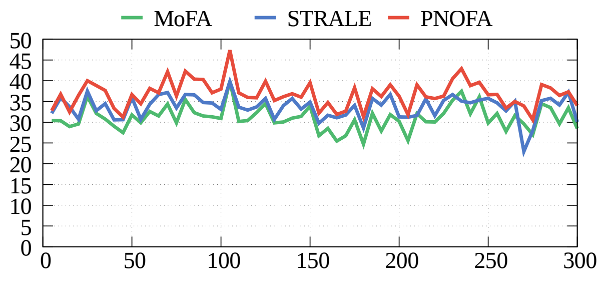
We have conducted a similar analysis for other scenarios (results not shown due to space). The results for MoFA suggest that it generally aggregates too many subframes, resulting in higher error rates and lower throughput than PNOFA. Our experiments also show that STRALE's mechanism of lowering the transmission rate appears to be too conservative and it cannot compensate for the loss of throughout caused by choosing lower transmission rates. Our evaluations in Sections 6.5 and 6.6 show that PNOFA consistently maintains effective A-MPDU sizes. Across all seven scenarios studied the average throughput of PNOFA is within 97% of the average throughput of the statistically optimal algorithm.

7 EXPERIMENTAL EVALUATION

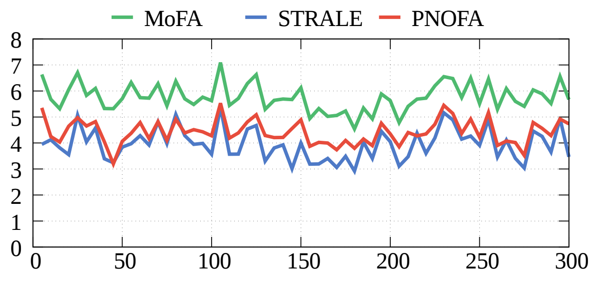
7.1 Methodology

We have implemented PNOFA on a commercial Google Wifi access point [10]. This device utilizes a Qualcomm IPQ 4019 [15] system-on-chip that supports Wave 2 802.11ac features. This chipset is widely used in over 82 WiFi access points including wireless mesh systems offered by Google, ASUS, D-Link, Linksys, Netgear, TP-LINK and Samsung [5].

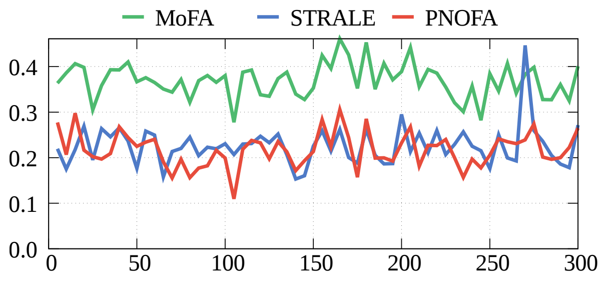
Tcpdump is used to capture the block ACK of the transmitted A-MPDUs to find out the fate of subframes. PNOFA utilizes this information to compute the MPDU Delivery Ratios (MDR) to determine the optimal A-MPDU length under the current channel conditions. Since the MDRs of different PHY-layer transmission rates are different, when PNOFA receives a block ACK it needs to know the rate at which the A-MPDU was transmitted (to record the data for the correct rate). However, this information is not available in the block ACK. Consequently, PNOFA retrieves the physical rate from the `iw` command for each connected station. In contrast with previous work, PNOFA can be implemented as a user-space process and does not require any modification to the firmware of WiFi chipsets.



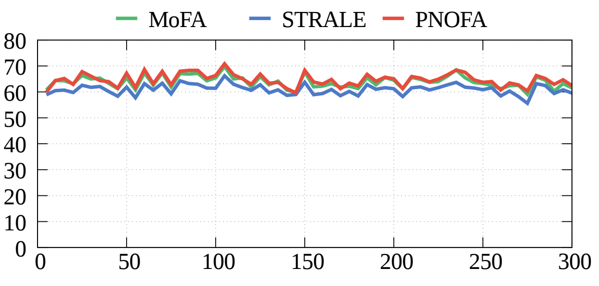
(a) Mean Throughput (Mbps)



(b) Mean A-MPDU length



(c) Frame Error Rate



(d) Mean PHY rate (Mbps)

Figure 9: Comparing metrics for different algorithms in Scenario S1

7.2 Performance Results

To evaluate the performance of PNOFA, we setup an experiment where a Google Wifi access point transmits UDP using `iperf` [6] to a mobile client. The client device is a Microsoft Surface Pro (5'th Gen) that is carried in an office environment (described in Section 6.2) at walking speed. We compare the performance of the frame aggregation algorithm in the Qualcomm IPQ4019 chipset and PNOFA in terms of the maximum achieved throughput. There is no line of sight between the access point and client for most of the trial. Each experiment consists of ten 30 seconds trials for each algorithm. We use a randomized interleaved trials [2] technique in which we switch between the two algorithms rather than running all trials of one algorithms and then the other one. This ensures that the two algorithms are exposed to similar channel conditions for a fair comparison.

Figure 10 shows the average UDP throughput obtained by each algorithm over the duration of the experiment. Each data point is the average throughput obtained during that one second window across 10 trials. As the figure shows PNOFA consistently outperforms the frame aggregation algorithm in the Qualcomm chipset. The achieved gain is up to 29% with a mean of 17%. Despite delays due to a user-space implementation, PNOFA significantly improves the throughput of the Qualcomm IPQ4019 chipset.

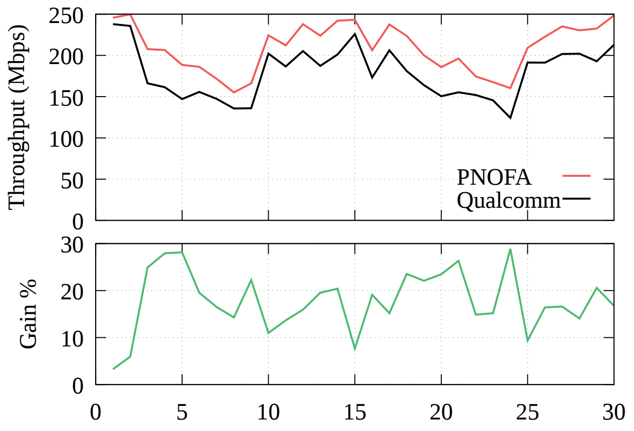


Figure 10: Average UDP throughput over time

We find that PNOFA outperforms state-of-the-art algorithms in a variety of scenarios. Across all scenarios studied the average throughput of PNOFA is within 97% of that of the statistically optimal algorithm. We also implement PNOFA on a Google Wifi access point and compare its performance with that of the proprietary frame aggregation algorithm used by the Qualcomm IPQ4019 chipset. Our experimental results show that PNOFA improves average throughput for UDP traffic by up to 17%.

8 CONCLUSIONS

This paper is the first to study the dependencies between consecutive A-MPDUs in determining the optimal A-MPDU length. We develop a standard compliant, Practical, Near-Optimal Frame Aggregation algorithm (PNOFA) PNOFA which estimates the expected subframe delivery ratios to determine the number of frames to aggregate that will result in high throughput.

9 ACKNOWLEDGMENTS

We acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Foundation for Innovation (CFI), the Ontario Research Fund (ORF), and Google. We also thank Kamal Rahimi Malekshan for many fruitful discussions during the early phases of this work.

REFERENCES

[1] Ali Abedi, Tim Brecht, and Andrew Heard. 2018. T-SIMn: A trace collection and simulation framework for 802.11n networks. *Computer Communications* 117 (2018).

[2] Ali Abedi, Andrew Heard, and Tim Brecht. 2015. Conducting Repeatable Experiments and Fair Comparisons Using 802.11n MIMO Networks. *SIGOPS Oper. Syst. Rev.* 49, 1 (2015).

[3] Seongho Byeon, Kangjin Yoon, Okhwan Lee, Sunghyun Choi, Woonsun Cho, and Seungseok Oh. 2014. MoFA: Mobility-aware Frame Aggregation in Wi-Fi. In *CoNEXT*.

[4] Seongho Byeon, Kangjin Yoon, Changmok Yang, and Sunghyun Choi. 2017. STRALE: Mobility-aware PHY rate and frame aggregation length adaptation in WLANs. In *INFOCOM*.

[5] Wireless CAT. 2020. Qualcomm Chipsets. (2020). https://wikidevi.wi-cat.ru/Qualcomm

[6] Jon Dugan. 2020. IPerf: The ultimate speed test tool for TCP, UDP and SCTP. http://sourceforge.net/projects/iperf/. (2020).

[7] Felix Fietkau and D Smithies. 2010. Minstrel HT: New rate control module for 802.11n. (2010).

[8] Nasreddine Hajlaoui, Issam Jabri, Malek Taieb, and Maher Benjemaa. 2012. A frame aggregation scheduler for QoS-sensitive applications in IEEE 802.11n WLANs. In *International Conference on Communications and Information Technology (ICCIT)*.

[9] Andrew Heard. 2016. *T-SIMn: Towards a Framework for the Trace-Based Simulation of 802.11n Networks.* Master's thesis. University of Waterloo.

[10] Google Inc. 2019. Google Wifi. (2019). https://store.google.com/product/google_wifi

[11] Shervin Khastoo, Tim Brecht, and Ali Abedi. 2020. NeuRA: Using Neural Networks to Improve WiFi Rate Adaptation. In *MSWiM*.

[12] Yuxia Lin and Vincent W. S. Wong. 2007. Frame aggregation and optimal frame size adaptation for IEEE 802.11n WLANs. In *GLOBECOM*.

[13] Jouni Malinen. 2020. hostapd. https://wireless.wiki.kernel.org/en/users/documentation/hostapd. (2020).

[14] Ioannis Pefkianakis, Yun Hu, Starsky H.Y. Wong, Hao Yang, and Songwu Lu. 2010. MIMO rate adaptation in 802.11n wireless networks. In *MobiCom*.

[15] Qualcomm Technologies, Inc. 2019. IPQ4019. (2019). https://www.qualcomm.com/products/ipq4019

[16] Anwar Saif and Mohamed Othman. 2013. SRA-MSDU: Enhanced A-MSDU frame aggregation with selective retransmission in 802.11n wireless networks. *Journal of Network and Computer Applications* (2013).

[17] Anwar Saif, Mohamed Othman, Shamala Subramaniam, and Nor Asila Wati Abdul Hamid. 2012. An Enhanced A-MSDU Frame Aggregation Scheme for 802.11n Wireless Networks. *Wireless Personal Communications* (2012).

[18] T Selvam and Srimanthula Srikanth. 2010. A frame aggregation scheduler for IEEE 802.11n. In *National Conference On Communications (NCC)*.