Computer Communications 180 (2021) 259–270

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# Demystifying frame aggregation in 802.11 networks: Understanding and approximating optimality

Ali Abedi [a,*], Tim Brecht [a], Omid Abari [b]

[a] Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
[b] Computer Science Department UCLA, Los Angeles, California, USA

A R T I C L E I N F O

*Keywords:*
802.11 networks
Frame aggregation
A-MPDU
Optimal length
Protocol optimization

A B S T R A C T

MAC-layer frame aggregation has significantly improved the efficiency of IEEE 802.11n and 802.11ac 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. In theory, larger A-MPDUs amortize overheads over more bits and therefore increase throughput. However in practice, throughput can be negatively impacted if too many frames are aggregated, due to higher error rates at the end of A-MPDUs. Determining the optimal number of subframes is challenging because error rates can be higher in the later part of the A-MPDU which change with factors such as mobility, speed and transmission rate. Additionally, there are dependencies between consecutive A-MPDUs due to software retransmissions.

We develop a model of A-MPDU frame aggregation and use it to design a *statistically optimal* algorithm. Using our understanding of that algorithm we 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 as a user-space process on an 802.11ac Wave 2, Google Wifi access point. We find that when compared with the frame aggregation algorithm provided in the device's Qualcomm IPQ 4019 chipset's firmware, PNOFA increases average throughput by 17% and 13% for UDP and TCP traffic on the scenarios tested.

## 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 aggregated frame containing multiple subframes is transmitted instead.

Maximizing throughput during transmission requires optimizing the number of subframes (MPDUs) to aggregate 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 is 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 (for more details see Section 2.1). 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.

Fig. 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. As will be seen later in this paper the MPDU delivery ratios vary over time and change with several factors including speed of movement and transmission rate. As a result, the optimal A-MPDU size also changes over time.

Recent studies such as MoFA [1] and STRALE [2] have developed frame aggregation algorithms that use heuristics to adapt to changing channel conditions in an attempt to maximize throughput. In this paper, we develop an analytic model that can be used to determine

* Corresponding author.
*E-mail addresses:* ali.abedi@uwaterloo.ca (A. Abedi), brecht@uwaterloo.ca (T. Brecht), omid@cs.ucla.edu (O. Abari).

https://doi.org/10.1016/j.comcom.2021.09.019
Received 31 March 2021; Received in revised form 8 July 2021; Accepted 20 September 2021
Available online 26 September 2021

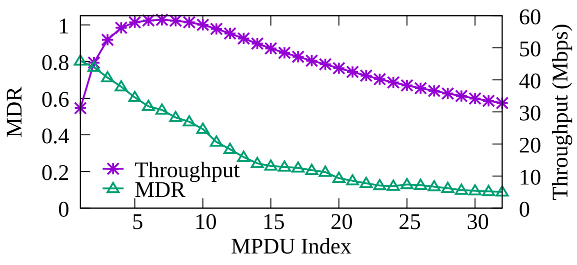


**Fig. 1.** Impact of A-MPDU size on throughput.

the statistically optimal number of frames to aggregate to maximize throughput. Our model is also 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. Using our analytical model, 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 that are very close to those obtained using the statistically optimal algorithm across a wide variety of scenarios and devices.

We also implement PNOFA on a Google Wifi access point [3] that utilizes the modern Qualcomm IPQ 4019 [4] system-on-chip that supports Wave 2 802.11ac features. This chipset is used in at least 65 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 (e.g., MoFA and STRALE) because they would require modifications to closed-source firmware. Our experimental results show that PNOFA can provide throughput improvements over the frame aggregation algorithm in the Qualcomm IPQ 4019 chipset.

Our contributions in this paper are:

- We determine *statistically optimal* A-MPDU sizes by modeling A-MPDU frame aggregation and block ACK window advancement. 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 properties that PNOFA relies on enable a user-space implementation on a Google Wifi access point. Our experimental results show that for the scenarios examined, PNOFA improves UDP and TCP throughput when compared to the Qualcomm IPQ 4019 chipset's frame aggregation algorithm on this device by 17% and 13%, respectively.

The organization of this paper is as follows: In Section 2, we study the challenges of A-MPDU frame aggregation. We then review the state-of-the-art solutions for this problem in Section 3. We then present our statistically optimal algorithm and practical near-optimal algorithm in Sections 4 and 5. Finally, we evaluate the performance of PNOFA and compare its performance with the state-of-the-art algorithms using trace-based simulation and empirical evaluations in Sections 6 and 7.

## 2. Aggregation challenges

In this section, we explain why creating an effective frame aggregation algorithm is an important, interesting, and challenging problem.

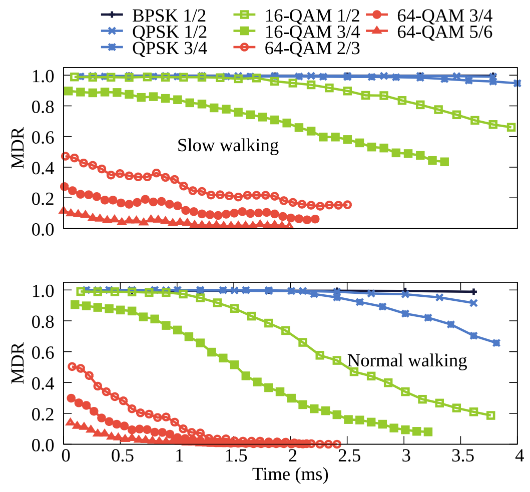


**Fig. 2.** Impact of modulation, coding and movement on MPDU delivery ratio (MDR).

### 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. [1] 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.

To provide some examples of channel compensation limitations, we conduct an experiment where we measure the MPDU delivery ratios for all transmission rates using a technique presented in Section 6.1. Fig. 2 shows the average MPDU delivery ratios for 8 modulation and coding schemes over the 200 s experiment for slow and normal walking speeds in the top and bottom plots. All 8 configurations use 3 spatial streams, a long guard interval, and a 20 MHz channel. The $x$-axis shows the location within the A-MPDU for each MPDU, in terms of the elapsed time from the PHY header. Due to limits on the total number of bytes in an aggregated frame the transmission of A-MPDUs with faster modulation and coding rates end sooner. As can be seen in these figures, the speed of movement, modulation and coding schemes have a direct impact on how fast the MPDU delivery ratio decreases for MPDUs that are farther from the beginning of the A-MPDU. As a result, a frame aggregation algorithm has to constantly monitor these changes to find the best A-MPDU length. For example, when comparing results using 16-QAM 3/4 across the two graphs the delivery ratio drops to a little below 0.5 for the last MPDU in the frame when the walking speed is slow. However, the delivery ratio for the last frame is below 0.1 when the walking speed is normal.

### 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. 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 which covers all subframes instead of one acknowledgment per subframe. The receiver of an aggregated frame selectively acknowledges the successful reception of multiple subframes with one block ACK. 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 802.11ac 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. For example, if the starting sequence number is 1000, the first and second bits acknowledge sequence number 1000 and 1001. 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 [6]. 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 (i.e., beyond 1063) 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.

In this example, we clearly see how the fate on subframes in one A-MPDU can affect the next A-MPDUs in terms of the maximum number of MPDUs that can be aggregated. 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. For example, does an optimal frame aggregation algorithm need to consider all possible outcomes for subsequent A-MPDUs to optimize the length of the next A-MPDU? In Section 4, we show that despite this chain of dependencies an optimal decision can be made for the current A-MPDU without considering future A-MPDUs.

## 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. We first provide a brief overview of the different categories of frame aggregation algorithms. We then describe related A-MPDU aggregation algorithms, the focus of this paper.

### 3.1. Frame aggregation scheduling

Frame Aggregation Schedulers [7,8] are concerned with optimizing the transmissions under unsaturated link conditions. The frame aggregation schedulers wait for new packets to become available for transmission on the sending device in order to create longer aggregated frames to improve the efficiency. These algorithms attempt to find a balance between delay and throughput as waiting too long for new packets may impact the QoS of the ready-for-transmission packets. Our work is concerned with ensuring maximum throughput, which occurs when there are no delays between packets.

### 3.2. A-MSDU frame aggregation

The second category of frame aggregation algorithms is concerned with finding the optimal length of an Aggregated MAC Service Data Unit (A-MSDU). In A-MSDU frame aggregation, there is one MAC header for all subframes, consequently if one MSDU fails the entire frame is lost. Therefore, A-MSDU frame aggregation algorithms [9–11] try to balance efficiency and frame error rates by dynamically adjusting the number of subframes aggregated in an A-MSDU. Similarly, another group of algorithms [12–14] try to optimize throughput by adjusting the size of MPDUs (i.e., not the number of MPDUs in an A-MPDU). This category of algorithms is similar to A-MSDU frame aggregation, because both of them ultimately change the size of MPDUs, either by changing the number of MSDUs in an MPDU or by changing the payload of MPDUs without using A-MSDU aggregation.

Our work is complementary to frame aggregation schedulers and A-MSDU frame aggregation algorithms so we do not consider them in this work.

### 3.3. A-MPDU frame aggregation

This category of frame aggregation algorithms is concerned with finding the optimal number of subframes to aggregate in an A-MPDU. The most closely related papers to our work are those describing the MoFA [1] and STRALE [2] frame aggregation algorithms. As a result, we implement these algorithms and compare their performance to that of our proposed algorithm. We now briefly review MoFA and STRALE followed by a description of the ChASER algorithm.

To overcome channel compensation limitations, Byeon et al. [1] 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. [2] 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. [1,2] 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 [2]. 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.

Okhwan et al. [15] propose ChASER an algorithm that re-estimates the CSI multiple times during the A-MPDU transmission to solve channel compensation limitations in the physical layer. The authors utilize a software radio to implement a prototype of their algorithm. This approach is costly and impractical for large-scale adaptation by commercial products [2]. In contrast, PNOFA is a standard compliant, practical algorithm that could easily be implemented in existing devices.

Although some past work has identified the problem of higher error rates at the end of A-MPDUs, their solutions are based on heuristics or hardware modifications. In contrast, for the first time, we study the

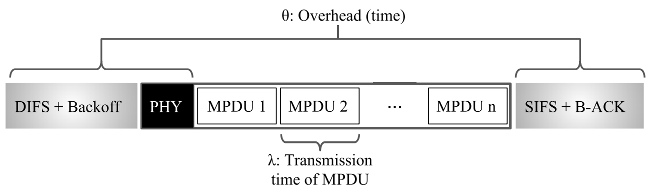


**Fig. 3.** 802.11 A-MPDU and transmission overheads.

inter-dependency between multiple A-MPDUs and derive a statistically optimal algorithm and a practical near-optimal A-MPDU frame aggregation algorithm. Moreover, the simple design of our algorithm enables the first implementation of a custom A-MPDU frame aggregation on a commercial 802.11ac access point. Finally, this work has enabled us to devise an algorithm that combines statistically optimal frame aggregation with a novel statistically optimal rate adaptation algorithm [16]. These algorithms provide a strong foundation for comparing new and existing algorithms.

## 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 $\theta$ includes all overheads associated with transmitting an A-MPDU, including PHY header, DIFS, SIFS, backoffs, and block ACK as illustrated in Fig. 3. $\lambda$ 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. 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 throughput, we need to find the value of $n$ that maximizes Eq. (3). The derivative of Eq. (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. Note that MoFA [1] and STRALE [2] use a similar formula to compute what the best length would have been for past A-MPDUs. Then they 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 Eq. (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 dependency 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 Eq. (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 cannot 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 Fig. 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 $\theta$ denotes all overheads when transmitting an A-MPDU. In both cases, since two A-MPDUs are transmitted, the total overhead will be $2\theta$. The transmission time of all 65 (i.e., 64+1 or 32+33) MPDUs will be $65\lambda$. 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 Figs. 4a and 4b, when creating A-MPDU$_x$, we need to schedule $i \geq 0$ MPDUs for retransmission, because they failed in previous A-MPDUs.

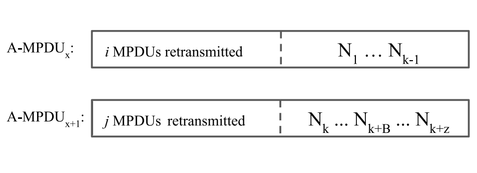
Assume that the optimal length of A-MPDU$_x$ without considering the dependency between A-MPDUs is $i + k$, therefore, we can add $k$ new MPDUs to A-MPDU$_x$, denoted $N_1 \ldots N_k$. To prove the theorem, we compare the possible outcomes of sending $i + k$ or $i + k - 1$ subframes in an A-MPDU.

**Aggregating $i + k - 1$ MPDUs** (Fig. 4a):

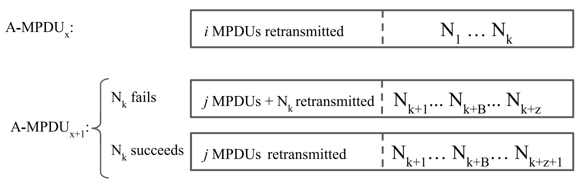
Assume that we add $k - 1$ new MPDUs to A-MPDU$_x$. When creating A-MPDU$_{x+1}$, we need to retransmit $j \geq 0$ MPDUs. The length of an A-MPDU can be limited by the block ACK window, the length suggested by Eq. (3) and possibly the maximum size of the frame. Since the last transmitted MPDU was $N_{k-1}$, 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 Eq. (3) is denoted by $N_{k+z}$. To illustrate a sample A-MPDU, in Fig. 4a, $N_{k+B}$ 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** (Fig. 4b):

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



**(a) Transmitting $k-1$ new MPDUs**



**(b) Transmitting $k$ new MPDUs**

**Fig. 4.** Transmitting $k-1$ versus $k$ new subframes in an A-MPDU.

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 Eq. (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$_x$ (Fig. 4a). As a result, there is no throughput gain from aggregating fewer MPDUs than what Eq. (3) suggests. This proof can be recursively applied on smaller A-MPDUs to show that smaller MPDUs do not provide higher throughput either. Hence, the best throughput is achieved when the length from Eq. (3) is used. □

Note that aggregating more frames than what Eq. (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 Eq. (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), in contrast to practical (i.e., online) algorithms like PNOFA, which estimates MDRs using the fate of previous A-MPDUs. To implement SO in our trace-driven evaluation, we calculate the expected throughput for all A-MPDU sizes from 1 to a maximum of 32 subframes[1] using Eq. (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. In these experiments, we use a constant transmission rate (i.e., MCS = 14, LGI, and 20 MHz channel) to eliminate the effect of the rate adaptation algorithm on the throughput. Each constant length setting algorithm chooses the same aggregation length throughout the experiment. In this 10 min experiment, we gradually increase the speed of movement from 0.5 m/s to 1.5 m/s. As illustrated in Fig. 5 (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.

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

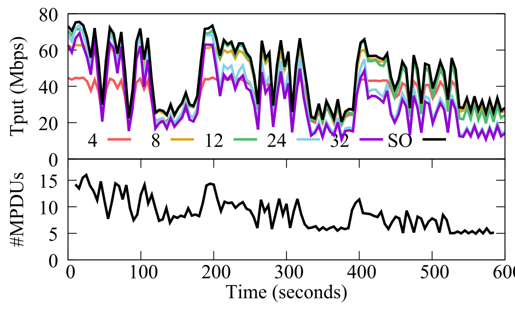


**Fig. 5.** SO versus constant A-MPDU sizes.

Fig. 5 (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 while walking 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 during the experiment.

## 5. PNOFA

The statistically optimal algorithm (SO) utilizes Eq. (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 some practical challenges.

### 5.1. PNOFA Algorithm

We now describe how we handle the practical challenges and present pseudocode for the implementation of PNOFA in Fig. 6. 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).

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 Eq. (3)). The number of additional MPDUs is determined by the ExtraMPDUsWindow and $\lambda$ (the MPDU transmission time at rate R)[line 9]. This approach aggregates slightly more than the optimal number of MPDUs, in order

```
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
```

**Fig. 6.** PNOFA algorithm.

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). In Section 6, we show that the throughput obtained using PNOFA is very close to that of the statistically optimal algorithm.

PNOFA is a lightweight algorithm that can be implemented on all WiFi devices. We now briefly explain the complexity of this algorithm. PNOFA first calculates MDRs over the last averaging window that contains $k$ samples. This operation is a simple averaging and hence its complexity is $O(k)$. $k$ is bounded by the number of A-MPDUs that can be transmitted in an averaging window. As we explain in the next section, the averaging window is typically a few hundred milliseconds and therefore $k$ is typically under 100. A careful implementation can avoid recalculating the averaging window for every A-MPDU by only adding new samples and removing those that are no longer inside the window. In the next step, PNOFA calculates the expected throughput for all A-MPDU lengths from 1 to 64. Therefore, the complexity of this step is $O(1)$. Our implementation of PNOFA on a commercial access point shows that the AP's WiFi chipset can easily handle the extra computations.

### 5.2. PNOFA parameters

We now explain the choice of values used for ExtraMPDUsWindow and AveragingWindow in our prototype.

**ExtraMPDUsWindow**: We have empirically determined how many subframes to add to an A-MPDU beyond the value obtained from Eq. (3). Because the amount of 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

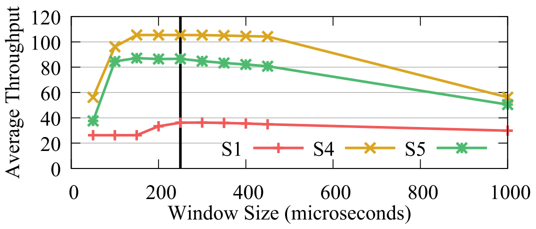


**Fig. 7.** ExtraMPDUsWindow sizes and throughput.

devices and walking speeds (refer to Section 6.2 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 s. 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 s with a mix of slow and normal walking speeds.

Fig. 7 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, throughput 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. We have found that 250 microseconds allows PNOFA to quickly adapt when the A-MPDU length should be increased while otherwise incurring very little overhead.

**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 ms 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.

The near optimal throughput of PNOFA obtained in Section 6 indicates that these choices of parameters are effective across a variety of channel conditions and transmission rates.

## 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 [17] 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 [18] has demonstrated that T-SIMn is extremely realistic and highly accurate. A key to this trace-driven evaluation is that information about the properties of the channel that impact throughput are captured in the traces. With this approach traces are collected during real experiments using real devices in a variety of environments that are representative of those in which the devices are likely to be used. 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.

#### 6.1.1. Trace-driven evaluation platform

To evaluate the performance of different frame aggregation algorithms, for 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 at the maximum length to measure the MPDU delivery ratios of all MPDU indexes. In the next step, T-SIMn uses the collected trace to evaluate different frame aggregation algorithms.

*Implementation:* We ported code from the publicly available STRALE implementation [2] to T-SIMn. We have also implemented MoFA as described by Byeon et al. [1]. Finally, the statistically optimal algorithm and PNOFA are implemented as described in Sections 5 and 4.3. 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 [19].

#### 6.1.2. 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 (AR9380 chipset) that supports up to three streams (i.e., 3 × 3:3 MIMO configuration). We create an 802.11n AP using `Hostapd` [20] on this machine. This device uses a modified `ath9k` (Atheros) device driver that enables round-robin trace collection. Traces are used as input to the trace-based evaluation.

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 (3 × 3:3 MIMO configuration) or a TP-Link Archer T9UH dual-band wireless 802.11ac USB adapter (this supports up to a 4 × 4:4 MIMO configuration). We have also used a laptop equipped with 802.11ac Intel AC 3160 WiFi chipset (1 × 1:1 configuration) as the client in some experiments. The 802.11ac devices support a backward compatible mode which allows them to be used with the 802.11n access point. In this mode, they are limited to 3 × 3 MIMO transmissions because our access point supports up to 3 spatial streams. Since the goal of these frame aggregation algorithms is to maximize MAC-layer throughput we use `iperf` [21] 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, since the WiFi channel is a shared medium. Therefore, the performance of frame aggregation algorithms is critical in both saturated (e.g., large downloads) and bursty (e.g., video streaming) traffic scenarios.

In this section, we evaluate the performance of PNOFA in determining the optimal number of frames to aggregate. Our main goal in this section is to determine how close PNOFA is to the statistically optimal

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

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

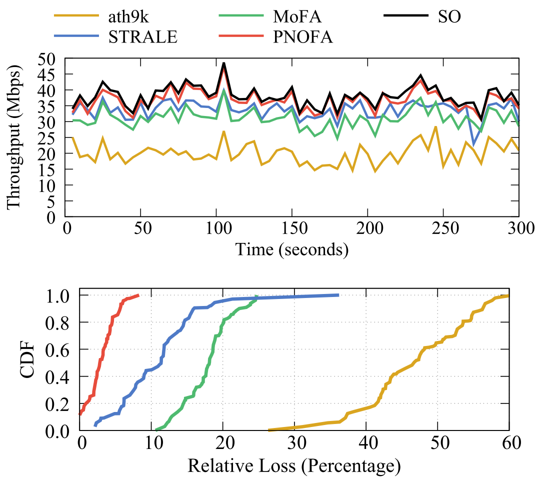


**Fig. 8.** Throughput and CDF of relative loss.

algorithm. We also examine the performance of existing state-of-the-art frame aggregation algorithms namely, MoFA [1] and STRALE [2]. Additionally, we study the frame aggregation algorithm used in the widely-used `ath9k` driver (subsequently referred to as `ath9k`). This algorithm always aggregates as many MPDUs as possible because it was created before channel compensation limitations had been discovered.

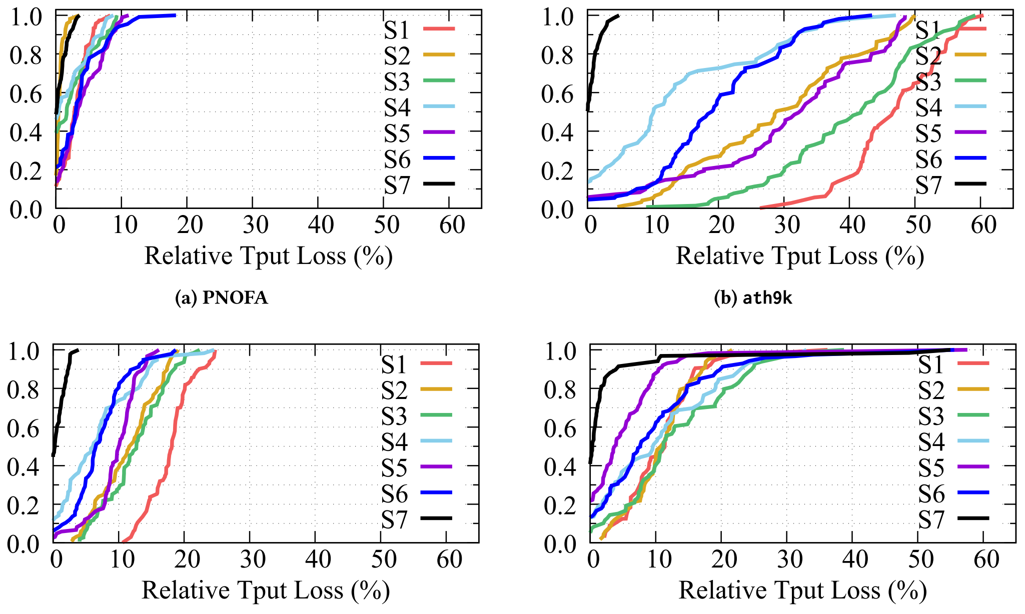
### 6.2. Scenarios studied

We conduct experiments using several scenarios to evaluate the performance of PNOFA (and other competing frame aggregation algorithms) using different network configurations and devices. 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 s at normal walking speed. Scenarios 2 through 6 start with 200 s at a low walking speed followed by 200 s of normal walking speed (i.e., mixed speed). Finally, in Scenario 7, a stationary experiment is conducted for 400 s.

### 6.3. Performance details for Scenario 1

We start by examining the throughput obtained using the different frame aggregation algorithms in Scenario 1 (S1). Fig. 8 (top) shows the achieved throughput over time (averaged over 5 s intervals) for all algorithms and the statistically optimal algorithm (labeled SO). This graph shows that for the duration of the experiment 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 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



(a) PNOFA (b) `ath9k` (c) MoFA (d) STRALE

**Fig. 9.** CDFs of throughput loss, relative to optimal.

relative to that of SO in Fig. 8 (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. The `ath9k` algorithm achieves the lowest throughput because it always aggregates as many MPDUs as possible.

## 6.4. Performance with different scenarios

Fig. 9 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. The purpose of these graphs is to examine how close each algorithm is to optimal across the variety of scenarios being studied.

Fig. 9a shows the performance of PNOFA across all 7 scenarios described in Table 1. This graph 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 for MPDUs (obtained from the trace), the estimates of frame error rates from PNOFA 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.

Fig. 9b shows results for the `ath9k` algorithm. It only performs well in one scenario, S7. In S7, 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 (as do all algorithms for S7). Fig. 9b 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.

Figs. 9c and 9d show the performance of MoFA and STRALE, respectively. These graphs demonstrate 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.

Figs. 9c and 9d also reveal that the performance of MoFA and STRALE are not robust in terms 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 the fact that these algorithms are based on heuristics that are not necessarily accurate for all scenarios.

Thanks to the repeatability of the trace-driven evaluation, we could collect traces in environments with WiFi and non-WiFi interference and then fairly compare the competing algorithms. These evaluations show that the PNOFA's performance is resilient to WiFi and non-WiFi interference because its throughput is always very close to that of the statistically optimal algorithm. In contrast, we believe that some of the performance issues of MoFA and STRALE could be rooted in confusing subframe errors caused by interference with errors due to limitations in channel estimation.

## 6.5. Analysis of algorithm differences

To better understand the differences between the behavior of PNOFA, STRALE and MoFA, Fig. 10 shows several different metrics for each algorithm across two scenarios (S1 on the top and S2 on the bottom). 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.

When examining scenario S1 (Fig. 10 top), one can see that over the length of the experiment the mean throughput (Fig. 10a top) 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 (Fig. 10b top), which results in higher frame error rates (Fig. 10c top). 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 on average relatively close to that of PNOFA. However, its mean PHY rate is slightly lower than that of PNOFA and MoFA (Fig. 10d top) because STRALE sometimes reduces the transmission rate to help cope with the channel compensation problem. The shorter A-MPDU frames combined

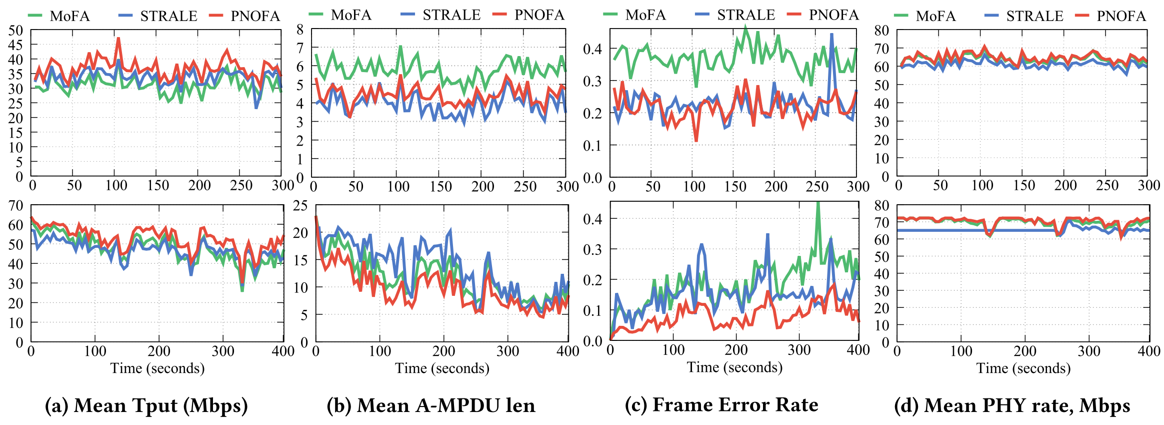


**Fig. 10.** Comparing metrics for different algorithms: Scenario S1 (top) and S2 (bottom)

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 average PHY rate result in lower throughput than PNOFA.

Scenario S2 (Fig. 10 bottom) presents a slightly different picture, especially with regards to average lengths of frames aggregated. For the first 240 s of the experiment the mean A-MPDU length used by STRALE is noticeably higher than that of MoFA and PNOFA and then is fairly close to or slightly lower than MoFA for the remainder of the experiment (Fig. 10b bottom). During that first 240 s the throughput of MoFA is mostly higher than that of STRALE (Fig. 10a bottom). This is because during that time the frame error rates are relatively similar (Fig. 10c bottom), but STRALE is using PHY rates that are lower than MoFA (Fig. 10d bottom). During the interval from 240 to 400 s, STRALE reduces the mean number of frames it aggregates which combined with lower PHY rates results in lower frame error rates and throughput that is mostly higher than MoFA. Throughout this experiment the mean number of frames aggregated by PNOFA is almost always lower and throughput is almost always higher than STRALE and MoFA.

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 throughput caused by choosing lower transmission rates. Our evaluations in Sections 6.4 and 6.5 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. The key is PNOFA's ability to balance between efficient aggregated frames (with longer A-MPDUs) and sufficiently low error rates (with shorter A-MPDUs).

## 6.6. *The impact of network configurations*

In this section, we characterize the impact of network configurations such as the channel bandwidth and the spectrum on the MPDU delivery ratios. Our goal is to obtain deeper insights into the differences in scenarios studied and their impact on frame aggregation algorithms.

### 6.6.1. *Effect of frequency band*

We now compare the effects of frequency bands on A-MPDU aggregation. We run two experiments in the 2.4 and 5 GHz spectrums using a constant transmission rate (i.e., MCS = 14, long guard interval, and 20 MHz channel) at two different speeds of movement. Fig. 11 (top) plots the relative MPDU delivery ratio over time for different bands and mobility speeds when other parameters remain constant. We plot relative values instead of absolute values in order to easily compare delivery ratios on the same plot. This graph show that the MPDU delivery ratio degrades much faster for the 5 GHz band than the 2.4 GHz band. This is due to the fact that the channel coherence time is much shorter in the 5 GHz band, and therefore the channel changes much faster. Therefore, the 5 GHz band is more prone to problems caused by channel compensation limitations. This observation explains why `ath9k` performs better in Scenario 6 (i.e., 2.4 GHz) than Scenario 5 (i.e., 5 GHz).

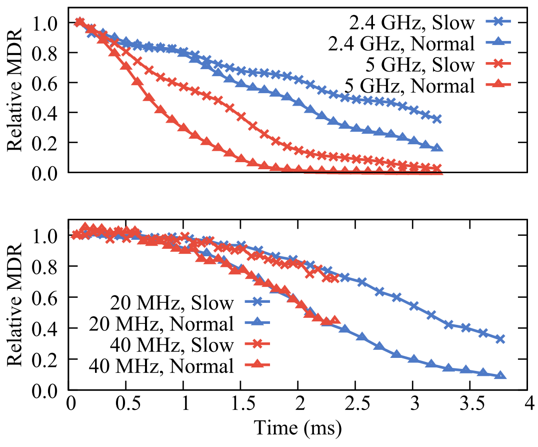


**Fig. 11.** Impact of frequency and channel bandwidth on MPDU delivery ratios.

### 6.6.2. *Effect of channel bandwidth*

In the 802.11n and 802.11ac standards, the number of pilot subcarriers used for channel compensation is 4 and 6 in 20 MHz and 40 MHz channels. Since the ratio of pilots to all subcarriers is lower in the 40 MHz channels, one might think that channel correction could be less effective. We run two experiments using 20 and 40 MHz channel bandwidths using the 5 GHz spectrum with MCS = 12 and long guard intervals at two different speeds. Fig. 11 (bottom) plots the relative MPDU delivery ratio over time for different channel bandwidth and mobility speeds while other parameters are unchanged. The figure shows that despite the different ratios of pilot subcarriers, the relative MPDU delivery degrades at the same rate for both 20 MHz and 40 MHz channels. However, for the same A-MPDU length, the transmission time is much longer for 20 MHz channels. Therefore, the network configurations that utilize a 20 MHz channel encounter more changes in the channel conditions during the transmission of an A-MPDU and as a result they are more prone to problems caused by calibration compensation than those with 40 MHz channels. This observation explains why `ath9k` performs better in Scenario 4 (i.e., 40 MHz) than Scenario 3 (i.e., 20 MHz).

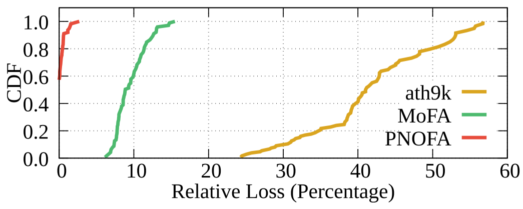


**Fig. 12.** Performance of frame aggregation algorithms in isolation from rate adaptation.

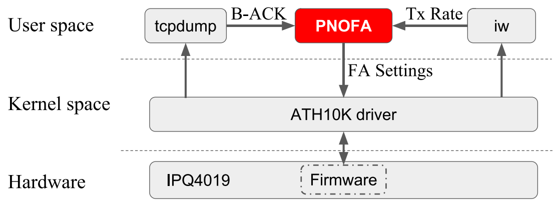


**Fig. 13.** The architecture of system implementation.

### *6.7. Frame aggregation in isolation*

In practice frame aggregation and rate adaptation algorithms work together to determine the transmission configuration. For this reason, the previous section evaluates the performance of frame aggregation algorithms with a widely used rate adaptation algorithm. Since a frame aggregation algorithm may impact the error rate by changing the length of transmitted A-MPDUs, it may in turn modify the operation of the rate adaptation algorithm. For instance, in many of the scenarios studied here, reducing the length of A-MPDUs reduces the overall error rate. Consequently, the rate adaptation algorithm may decide to switch to a faster rate.

It is not possible to separate the impact of these algorithms when analyzing performance under the representative scenarios used in the preceding sections. However, we believe that it is interesting to examine the performance of frame aggregation in isolation from rate adaptation algorithms.

We have designed an experiment where the transmission rate remains constant during the experiment. As a result, any observed performance difference is only due to the frame aggregation algorithm. In this experiment, the client device is carried for 200 s at a slow walking speed (approximately 1.0 m/s), followed by another 200 s of normal walking speed (approximately 1.4 m/s). The transmission rate is set to MCS=14, long guard interval, and 20 MHz channel (i.e., 117 Mbps). Fig. 12 plots the CDF of the relative throughput loss of PNOFA, MoFA and `ath9k` when compared to the statistically optimal algorithm. STRALE cannot be evaluated in this experiment because by design it interacts with the rate adaptation algorithm (adjusting both frame aggregation and rates together). The results in this graph show that the throughput obtained from PNOFA is always within 3% of optimal. The throughput of MoFA and `ath9k` deviates from optimal by up to 15% and 57%, respectively. This experiment confirms again that the throughput obtained using PNOFA is very similar to that of the SO algorithm, throughout the entire experiment.

## 7. Experimental evaluation

### *7.1. Implementation*

We have implemented PNOFA on a commercial Google Wifi access point [3]. This device utilizes a Qualcomm IPQ 4019 [4] system-on-chip that supports Wave 2 802.11ac features. This chipset is widely used in over 65 models of WiFi access points including wireless mesh systems offered by Google, ASUS, D-Link, Linksys, Netgear, TP-LINK and Samsung [5]. Fig. 13 illustrates the architecture of our system implementation. PNOFA runs as a user-space process and receives block ACK and PHY-layer rate information from `tcpdump` [22] and `iw` [23], respectively. In addition, PNOFA adaptively adjusts the A-MPDU length via an interface provided by the ath10k driver [24] that enables manual A-MPDU size adjustments per station.

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.

Although in theory it might be possible to implement PNOFA in the `ath10k` driver, that requires recompiling and installing new operating system firmware on the access point. We expect that PNOFA would perform better if implemented in the `ath10k` driver or by manufacturers in the WiFi chipset's closed-source firmware. Neither of these approaches is an option because the source is not available for the Google Wifi operating system firmware or for the Qualcomm IPQ4019 firmware. In contrast, our user-space implementation requires no modifications to the existing 802.11ac drivers or chipset firmware. Therefore, it can be installed on commercially deployed access points.

To the best of our knowledge, PNOFA is the first A-MPDU frame aggregation algorithm that can be implemented as a user-space process on an 802.11ac platform. This is possible with PNOFA because it relies on statistical properties of recent frames (i.e., the 200 ms averaging window) to make decisions about the size of future frames. Such an implementation is not possible for MoFA and STRALE since they make decisions about the size of the next A-MPDU based on the fate of previous A-MPDU. However, the delay in receiving this information in the application layer is too large. In addition and more importantly, STRALE also modifies the rate adaptation algorithm, which is not possible because it is also implemented in the chipset's closed-source firmware. On the other hand, PNOFA can be implemented as a user-space process and does not require any modifications to the driver or firmware.

### *7.2. Performance results*

To evaluate the performance of PNOFA, we setup an experiment where a Google Wifi access point transmits UDP or TCP packets using `iperf` [21] 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 6.1.2) 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 s trials for each algorithm. We use a randomized interleaved trials [25] 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.

Table 2 summaries our evaluation results for the UDP and TCP experiments. The table shows the average throughput obtained from ten trials along with 95% confidence intervals. PNOFA outperforms the frame aggregation algorithm of the Qualcomm IPQ4019 chipset in terms of the average throughput by 17% and 13% for UDP and TCP traffic.

We now examine how the performance gains change over time. Fig. 14 shows the average UDP throughput (of 10 trials) obtained by each algorithm over the duration of the experiment. Each data point is the average throughput obtained during that one second window across

**Table 2**
Average throughput of 10 trials (Mbps).

| Protocol | PNOFA | Qualcomm | Gain |
|---|---|---|---|
| UDP | $210.3 \pm 3.0$ | $179.5 \pm 0.8$ | 17.2% |
| TCP | $162.9 \pm 1.1$ | $144.7 \pm 4.6$ | 12.6% |

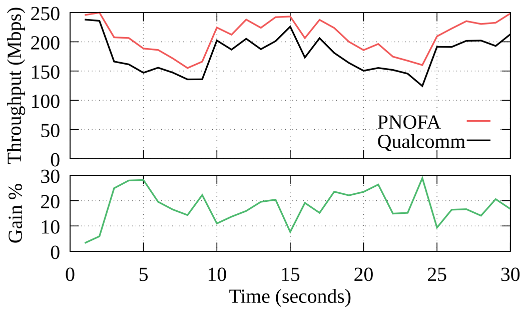


**Fig. 14.** Average UDP throughput over time.

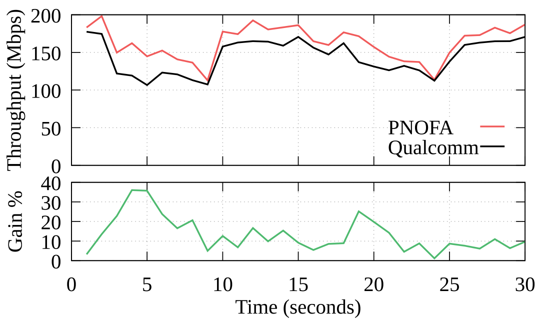


**Fig. 15.** Average TCP throughput over time.

all 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%.

Fig. 15 shows the throughput achieved by each algorithm with TCP traffic. Similar to the UDP results, PNOFA consistently achieves higher throughput. The gain is up to 36% with a mean of 13% for the duration of the experiment.

Despite delays due to a user-space implementation PNOFA significantly improves the throughput of the Qualcomm IPQ4019 chipset. An interesting observation is that during our experiments we noticed that the Qualcomm chipset restricts the duration of A-MPDUs to about 2 ms which is far less than the 5.5 ms limit specified by the IEEE 802.11ac standard. Note that in our experiments this hard coded limit also applies to PNOFA. Specifically, if PNOFA determines that the best size of an A-MPDU is more than 2 ms, the IPQ4019 chipset restricts the packet to 2 ms and our algorithm cannot take advantage of more efficient longer frames. If this restriction is removed, PNOFA's gains may be even larger than those observed in these experiments. We speculate that this self-imposed static limit was introduced by Qualcomm to ameliorate the channel compensation problem. Despite these potential efforts, we are still able to obtain improvements which demonstrates the advantages of an algorithm that dynamically adjusts to changing channel conditions.

## 8. Conclusions and future work

In this paper, we derive a model for determining the statistically optimal number of frames to aggregate in modern 802.11 networks. We then develop a standard compliant, Practical, Near-Optimal Frame Aggregation algorithm (PNOFA). PNOFA estimates the expected subframe delivery ratios to determine the number of frames to aggregate and approximates the statistically optimal algorithm.

We evaluate the performance of several competing frame aggregation algorithms using trace-driven evaluations. We find that PNOFA outperforms state-of-the-art algorithms in a variety of scenarios. Most importantly, across all seven scenarios studied the average throughput of PNOFA is within 97% of the average throughput of the statistically optimal algorithm. We also implement PNOFA on a Google Wifi access point and compare its performance with that of the frame aggregation algorithm used by the Qualcomm IPQ4019 chipset on this device. 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. Our experimental results show that PNOFA improves average throughput for both UDP and TCP traffic by up to 17% and 13%, respectively.

We believe that maximizing WiFi throughput requires a holistic approach that combines frame-aggregation and rate adaptation. Therefore, in future work we plan to design a combined frame aggregation and rate adaptation algorithm that incorporates ideas from PNOFA. Additionally, as the number of physical bitrates continues to grow finding the optimal bitrate becomes increasingly challenging. As a result, we plan to utilize relationships between different bitrates [26] to use only a subset of physical bitrates to reduce sampling overhead and increase throughput.

## CRediT authorship contribution statement

**Ali Abedi:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Tim Brecht:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Omid Abari:** Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Foundation for Innovation (CFI), the Ontario Research Fund (ORF), Canada, and Google.

## References

[1] S. Byeon, K. Yoon, O. Lee, S. Choi, W. Cho, S. Oh, MoFA: Mobility-aware frame aggregation in Wi-Fi, in: CoNEXT, 2014.
[2] S. Byeon, K. Yoon, C. Yang, S. Choi, STRALE: Mobility-aware PHY rate and frame aggregation length adaptation in WLANs, in: INFOCOM, 2017.
[3] G. Inc., Google wifi, 2019, URL https://store.google.com/product/google_wifi.
[4] Qualcomm Technologies, Inc, IPQ4019, 2019, URL https://www.qualcomm.com/products/ipq4019.
[5] W. CAT, Qualcomm chipsets, 2020, URL https://wikidevi.wi-cat.ru/Qualcomm.
[6] I. Pefkianakis, Y. Hu, S.H. Wong, H. Yang, S. Lu, MIMO rate adaptation in 802.11n wireless networks, in: MobiCom, 2010.
[7] T. Selvam, S. Srikanth, A frame aggregation scheduler for IEEE 802.11n, in: National Conference on Communications (NCC), 2010.
[8] N. Hajlaoui, I. Jabri, M. Taieb, M. Benjemaa, A frame aggregation scheduler for QoS-sensitive applications in IEEE 802.11n WLANs, in: International Conference on Communications and Information Technology (ICCIT), 2012.
[9] Y. Lin, V.W.S. Wong, Frame aggregation and optimal frame size adaptation for IEEE 802.11n WLANs, in: GLOBECOM, 2007.
[10] A. Saif, M. Othman, S. Subramaniam, N.A.W.A. Hamid, An enhanced A-MSDU frame aggregation scheme for 802.11n wireless networks, Wirel. Pers. Commun. (2012).
[11] A. Saif, M. Othman, SRA-MSDU: Enhanced A-MSDU frame aggregation with selective retransmission in 802.11n wireless networks, J. Netw. Comput. Appl. (2013).
[12] N. Hajlaoui, I. Jabri, M. Jemaa, Analytical study of frame aggregation in error-prone channels, in: IWCMC, 2013.

[13] K.-T. Feng, P.-T. Lin, W.-J. Liu, Frame-aggregated link adaptation protocol for next generation wireless local area networks, EURASIP J. Wireless Commun. Networking (2010).

[14] X. He, F. Li, J. Lin, Link adaptation with combined optimal frame size and rate selection in error-prone 802.11n networks, in: IEEE International Symposium on Wireless Communication Systems, 2008.

[15] O. Lee, W. Sun, J. Kim, H. Lee, B. Ryu, J. Lee, S. Choi, ChASER: Channel-aware symbol error reduction for high-performance WiFi systems in dynamic channel environment, in: INFOCOM, 2015.

[16] S. Khastoo, T. Brecht, A. Abedi, NeuRA: Using neural networks to improve WiFi rate adaptation, in: MSWiM, 2020.

[17] A. Abedi, T. Brecht, A. Heard, T-SIMn: A trace collection and simulation framework for 802.11n networks, Comput. Commun. 117 (2018).

[18] A. Heard, T-SIMn: Towards a Framework for the Trace-Based Simulation of 802.11n Networks (Master's thesis), University of Waterloo, 2016.

[19] F. Fietkau, D. Smithies, Minstrel HT: New rate control module for 802.11n, 2010.

[20] J. Malinen, Hostapd, 2020, https://wireless.wiki.kernel.org/en/users/documentation/hostapd.

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

[22] tcpdump, http://www.tcpdump.org/.

[23] iw, https://wireless.wiki.kernel.org/en/users/documentation/iw.

[24] Ath10k, 2019, https://wireless.wiki.kernel.org/en/users/drivers/ath10k.

[25] A. Abedi, A. Heard, T. Brecht, Conducting repeatable experiments and fair comparisons using 802.11n MIMO networks, SIGOPS Oper. Syst. Rev. 49 (1) (2015).

[26] A. Abedi, T. Brecht, Examining relationships between 802.11n physical layer transmission feature combinations, in: MSWiM, 2016.