A Passive Client Side Control Packet-based WiFi Traffic Characterization Mechanism

Lixing Song
Dept. of Computer Science and Software Engineering
Rose-Hulman Institute of Technology
Terre Haute, IN USA
song3@rose-hulman.edu

Alamin Mohammed, Aaron Striegel
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN USA
{*amohamm2,striegel*}*@nd.edu*

***Abstract*—WiFi has emerged as a pivotal technology for delivering Quality of Experience (QoE) to mobile devices. Unfortunately, exploding numbers of competing devices, potential encroachment by cellular technology, and dramatic increases in content richness deliver a more variable QoE than desired. Moreover, such variance tends to occur both across time and space making it a difficult problem to debug. Existing active approaches tend to be expensive or impractical while existing passive approaches tend to suffer from accuracy issues. In our paper, we propose a novel passive client-side approach that provides an efficient and accurate characterization by taking advantage of the properties of Frame Aggregation (FA) and Block Acknowledgements (BA). We show in the paper that one can accurately derive important metrics such as airtime and throughput with only a minimal amount of observed BAs. We show through extensive experiments the validity of our approach and conduct validation studies in the dense environment of a campus tailgate.**

I. Introduction

Since the advent of the modern smartphone, WiFi has served as a critical technology in satisfying the Quality of Experience (QoE) needs for mobile users. For most if not all users, WiFi is equated with high speed and has become a necessity for the modern venue. However, as most who have used WiFi at various public venues can attest, the actual QoE tends to vary considerably. While new standards seek to deliver a "better" WiFi, the increasing density of devices, richness of content, and encroachment on WiFi by cellular [1] make performance variance a likely normal for the foreseeable future.

For nearly all involved parties, such variance is incredibly frustrating. Performance is usually good enough, once in a while great, occasionally bad, and sometimes positively terrible. Moreover, debugging is difficult as the variance occurs across both time and space. The situation while tenable requires action but unfortunately much of the prior work tends to be ill-suited to solve the problem.

Traditionally, the common solution is for mobile device becomes an active sensor to actively determine end-to-end performance at the network and transport layers [2]–[4]. Examples such as Speedtest.net, iperf3, and Mobiperf [5] embody this approach whereby the currently connected WiFi is actively probed to determine peak network performance. Critically, the cost of such probing is often quite high both in terms of time and energy relegating such tests often to be reactive rather than longitudinal. Furthermore, such tests also have a negative impact on other users as the probe traffic can be intrusive to existing traffic. The net result is that active probing often misses the broader picture of the WiFi environment including the influence of other mobile nodes, channel airtime, transmission speed, queuing effects, and other subtle link properties[1].

In contrast to the mobile node as an active sensor, other work has operated from the perspective of the access point (AP) to afford a broader view of the wireless network [6]–[8]. By deploying well-equipped APs with coordination through a back-end controller, a rich set of WiFi characterization details can be gathered. Existing network performance can be gleaned from connected clients which in turn provides a wealth of performance data for the deployed network. As would be expected, such services tend to be expensive but often essential to any large-scale WiFi deployment. Notably, a key weakness of the AP-centric focus is that while APs can ably sense, the entire collection of APs represent a limited and stationary spatial distribution. Thus, such systems tend to focus largely from the perspective of the provided network, potentially missing client-side issues at the edge of the network.

Finally, the last set of approaches is to view the mobile client itself as a capable but purely passive wireless sensor [9]–[12]. In contrast to the AP-side approach, the client-side method acts exclusively as a "sniffer" on the WiFi network without AP-side information (queue length, transmission rates, etc.). The client-side approach provides increased flexibility with mobile nodes crowdsourcing the state of the WiFi network. Unfortunately, while interesting conceptually, the actual packet capture capabilities of most mobile devices tend to fare quite poorly. Packet capture is often inaccessible absent significant modifications by the device owner and as will be discussed later, packet capture often suffers severe losses when monitoring data packets. Critically, the mobile-centric approach does offer one highly desirable property, namely that of offering longitudinal data across the entirety of the potential client connection area.

Thus, the questions that we pose in this paper are: *How could we could radically improve the ability of a mobile client*

[1]In fairness to prior work, active end-to-end techniques were never intended to capture link properties.

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

to observe the network knowing the capture limitations of existing mobile devices? In particular, are there certain packets that are more easily observable but yet contain rich information to help characterize the network? In this paper, we propose a new technique that builds on the properties of frame aggregation and show how passively observable Block Acknowledgements map to a rich set of link characterization metrics such as airtime and throughput through extensive experimental studies. Although we believe other characterization metrics e.g, transmission rate and queuing information could also be derived, we only focus on airtime and throughput as their ground truth data can be easily obtained. The contributions of this paper are:

- *Sensing with Control Packets:* We show how Block Acknowledgements (BA) can be used to extract a rich set of WiFi characteristics. We define two metrics - *Aggregation Intensity (AI)* and *BA Time Gap* and show how these two metrics can be used to compute airtime and throughput. We demonstrate the accuracy, efficiency, and robustness of these mappings through extensive empirical studies across a wide variety of scenarios.
- *Real-World Data Validation:* We validate our approach and its accuracy by conducting experiments on a real-world dataset captured during a tailgate involving multiple 802.11ac access points. We conduct trace-driven experiments and show that the designed characterization can achieve a high correlation (0.8) with the observed ground truth for airtime and throughput.

II. Background & Motivation

Before diving into the details of our system, we discuss several key background concepts. First, in Section II-A, we discuss the need for using control packets followed by a basic primer on frame aggregation and block acknowledgement.

A. Why Control Packets?

The goal of a client-side characterization model is to provide traffic characterization by capturing most if not all traffic going on the channel(s). Unfortunately, the assumption of capturing all traffic is often impossible in practice. Eavesdropping on a WiFi channel tends to suffer from severe loss for several reasons. First, the data packets with high transmission rate usually have a limited communication range to be captured. Second, the different bandwidth capacities of devices impede capture as low capacity devices (e.g., low bandwidth, low supported rate) cannot collect traffic transmitted with high rates. In addition, with beamforming in advanced WiFi (i.e., 802.11ac wave-2 [13]), a device is not able to hear traffic from the directional antenna if it is not on the transmission path. These difficulties hinder a client from capturing the full picture of traffic on channel.

Fortunately, control packets tend not to suffer from the same issues with respect to capture. Since control packets are designed to be acknowledged by all nearby devices, they are set to be transmitted at the lowest rate in a non-directional, unencrypted manner. Further, the volume of the control packets is much sparser than of the data packets. From a capture standpoint, this implies that the mobile device is unlikely to be overwhelmed and could potentially ignore data packets for energy efficiency. The key aspect for a mobile device is that while data packets can lose in excess of 75% of packets when attempting to log all packets (from our experiments on 802.11n/2.4 GHz with signal -70 dBm or worse), it is quite rare to lose control packets while attempting to capture only control packets (success rate of 75% even with 802.11ac on 5 GHz with -85 dBm power).

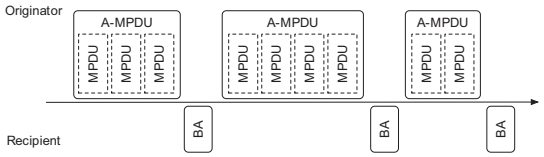


Fig. 1: Illustration of frame aggregation.

B. Why Frame Aggregation?

The general principle of frame aggregation is to assemble multiple data units to transmit as one aggregated frame[2]. The aggregated frame is created at two levels: the aggregate MAC protocol service unit (*A-MSDU*) and the aggregate MAC protocol data unit (*A-MPDU*). The A-MSDU is on the upper MAC layer which is further aggregated into the A-MPDU when pushed down to the physical layer. In this paper, we focus on leveraging the A-MPDU. As shown in Fig 1, each A-MPDU only requires a single Block Acknowledgment (BA) for notification of the receipt of multiple MPDUs (i.e., packets[3]). In order to support this one-to-many acknowledgement, the BA uses a bitmap field to explicitly indicate the failure or success of delivery of each MPDU.

As frame aggregation has become the default manner of sending data on modern WiFi, data transmission typically involve the exchange of a BA. These acknowledgements potentially provide opportunities to infer the data transmissions that have occurred. In particular, the information stored in BA frame allows one to know more about the data transmission beyond the number of packets. Particularly, we find that the information of how many MPDUs in an A-MPDU, dubbed Aggregation Intensity (AI), can embody a rich suite of information about the attributes of data traffic, e.g., queue length and transmission rate. In addition, we note that the time gap of BAs can also reveal other attributes about data transmission, e.g., transmission time of a packet. In next section (Sec. III), we discuss the technical details about how we can extract this information to achieve accurate characterization based on control packets.

III. WiFi Characterization via Control Packets

In order to characterize the WiFi channel, we define two base metrics: 1) the *Aggregation Intensity* (AI) and 2) the Block Ack (BA) Time Gap. Based on these two measurements,

[2]Noted that the term *packet* speaks to MAC and upper layers, while *frame* refers to PHY layer.

[3]*Packet and MPDU are used interchangeably for the rest of the paper.*

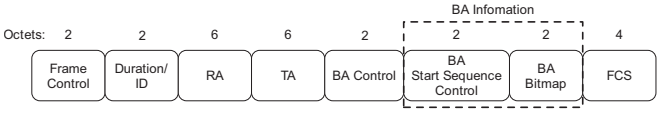


Fig. 2: Format of Block ACK frame.

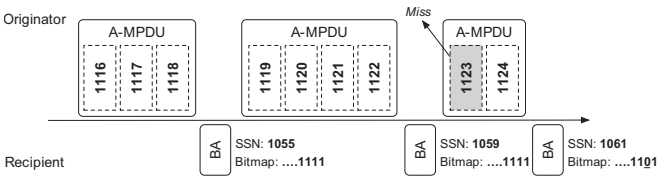


Fig. 3: Data transmission under frame aggregation. The sequence number (SSN) and bitmap (64 bits) are indicated.

we can further derive various important characterization metrics, e.g., channel airtime and throughput.

A. Aggregation Intensity (AI)

As noted earlier, frame aggregation allows multiple data units to be assembled into one aggregated frame (A-MPDU) for transmission. We describe the degree of aggregation using a metric called *Aggregation Intensity* (AI) that counts the number of MPDUs (e.g., packets) within one aggregated frame. Figure 1 shows an example where the three A-MPDUs have AIs of 3, 4, and 2, respectively.

Fundamentally, the value of AI is decided by several factors. When forming an A-MPDU, the scheduler looks into the queue and batches all the packets tagged with same TID (traffic identification) into a frame where the TID usually indicates the packets destined to the same address. Thus, the more packets with the same TID that are held in the queue, the larger the AI should potentially be with the maximum AI allowed in one A-MPDU is capped by 1) a maximum size and 2) a maximum transmission time T_{max}. Since the size limit is large (65,535 bytes), the AI is usually limited by the transmission time. For a certain packet size P, we have $T_{max} \leq \frac{AI \cdot P}{R}$ where R is the transmission rate. Thus the maximum AI can expressed as:

$$AI_{max} = \frac{R \cdot T_{max}}{P} \quad (1)$$

Extracting AI from BA: The computation of AI relies on two important fields in the BA frame: the Starting Sequence Control (SSC) and the Bitmap as shown in Figure 2. Each bit in the bitmap represents the receiving status (success/failure) of a MPDU. The SSC includes a sub-field called the Starting Sequence Number (SSN) which indicates the sequence number of the MPDU denoted by the first bit in the bitmap. Given a pair of consecutive BAs (sent from A to B), we can compute the AI and the loss of the corresponding A-MPDU (sent from B to A). For example, in Fig 3, we re-plot the case of Fig 1 to label the field information. For the first A-MPDU and its BA, since the last bit denotes the 1118 MPDU, the first bit should correspond to 1055 (1118 - 63) which is exactly the SSN of the BA. Combining the first and second BA, by subtracting their SSN, the AI of the A-MPDU between them can be computed

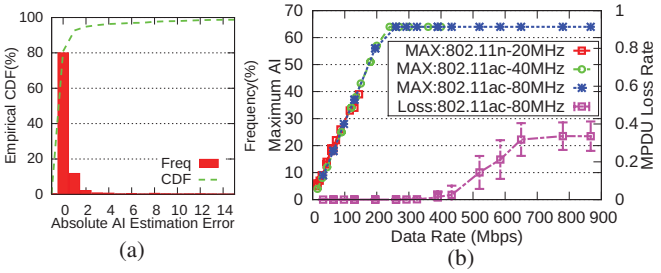


Fig. 4: (a) AI estimation error; (b) The maximum AI and loss rate across different transmission rate.

as 4 (1059 - 1055). When loss occurs as shown in the third A-MPDU, the zero in the bitmap indicates the miss of the 1123 MPDU. Thus, the loss count can be computed by the number of zeros in the bitmap.

Experimental Evaluation: We set up lab experiment to evaluate the performance of the proposed method. The general setting is as follows: we connected a server (HP ProBook) to a mobile client (HP ProBook equipped with EdiMax AC WiFi adapter) via an WiFi AP (TP-Link Archer c7 v.2). The AP is 802.11ac capable and configured to run `OpenWrt` which allows to adjust various settings, e.g., bandwidth (20/40/80 MHz), transmission rate, operating channel and so on. We generated traffic on WiFi by sending TCP flows (via `rsync`) from the server to the client. By using a third laptop (Lenovo P50 with Intel AC adapter) as the passive monitor node, we eavesdropped on traffic in the WiFi channel. In order to get the AI ground truth, we set the AP to run at a lower speed (802.11n 2.4GHz with 20MHz bandwidth) to allow us to capture most of the data packets. The ground truth for AI can be obtained from A-MPDU reference number in the radiotap header. Overall, we collected over 25,000 A-MPDUs and their BAs. In Fig 4a, we plot the CDF and the frequency of the absolute AI estimation error ($|groundtruth - estimated|$). It shows the designed method can achieve 81% of a perfect estimation of AI. For 97% of the estimations, the absolute error can be controlled within 5%.

By using the information extracted from BA, we continue to validate the relationship in Eq (1). We repeated the experiment above under different transmission rates with various bandwidth settings (20MHz, 40MHz and 80 MHz) on both 2.4GHz (802.11n) and 5GHz (802.11ac). As the result shown in Fig 4b, the observation matches Eq (1) well such that for all settings, the maximum AI linearly increases with the transmission rate until it reaches the maximum 64. This maximum is decided by the size of the bitmap used in this case. Moreover, we also plot the loss rate (number of loss in a time unit) of 80MHz bandwidth on 802.11ac. Given the static setting of the AP and the client, the channel quality (e.g., Signal-Noise Ratio) is fixed. When the SNR is inadequate to support the transmission rate, packet loss starts to occur. In our case, we see that when the transmission rate exceeds 400 Mbps, the loss rate starts to increase. Overall, the experiment results show that the designed method can help accurately capture AI and loss

across different network settings.

B. Measurement - BA Time Gap

While the AI provides the data packet depth, further understanding of the other properties of the packets (e.g., their cost on channel airtime) requires another metric, the BA time gap, which is the time gap between a BA and its *previously transmitted control packet* on a channel. Notably, any control packet could suffice as nearly all control packets are unencrypted. Nominally, this previous control packet is another BA as shown in Fig 1. It can also be other control packets. For example, with RTS/CTS enabled, a BA usually follows a CTS (Clear to Send). Once the control packets are captured, the BA gap can be simply computed by subtracting the packet timestamps recorded by the network adapter.

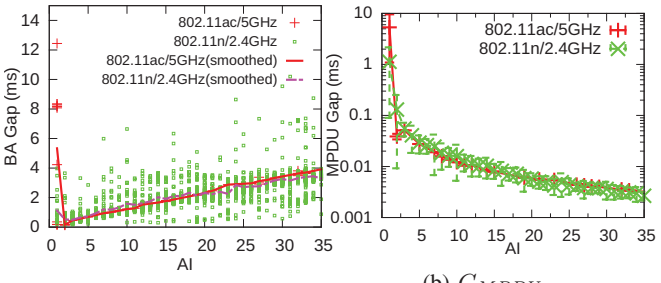
Since the BA is designed to follow the A-MPDU transmission, we argue that this time gap should be proportional to the airtime consumed by the data transmission. For a certain transmission rate, the more data assembled in an A-MPDU, the larger the BA gap should be. The relationship can be expressed as:

$$G_{BA} \propto \frac{AI \cdot P}{R} \tag{2}$$

where R is the transmission rate and P denotes the packet size. $AI \cdot P$ calculates the total size of the A-MPDU. To further calculate the average time consumed by each MPDU, we define a new metric — MPDU gap G_{MPDU} such that

$$G_{MPDU} \triangleq \frac{G_{BA}}{AI} \propto \frac{P}{R} \tag{3}$$

Once AI is estimated with the method mentioned above, G_{MPDU} can be computed. From Eq (2) and Eq (3), we see that the time gap information has the potential to reveal valuable attributes about data traffic, e.g., airtime and transmission rate. In order to validate the relationships in the equations, we conduct lab experiments to demonstrate the empirical observation of G_{BA} as well as G_{MPDU}.



(a) G_{BA} (b) G_{MPDU}

Fig. 5: G_{BA} (a) and G_{MPDU} (b) v.s. AI given certain transmission rate (i.e., 144.4 Mbps).

Experimental Validation: Using the prior experiment settings, we fixed the transmission rate on 144.4Mbps and repeated the TCP traffic on both 802.11ac 5GHz and 802.11n 2.4GHz. Fig 5a and Fig5b plot the G_{BA} and G_{MPDU} as a function of AI. The markers indicate the raw data, and the lines are the average for each AI value. As shown in Fig 5a, G_{BA} linearly increases with AI for all cases except for $AI = 1$. With the same transmission rate, the observation from 2.4GHz is identical to on 5GHz. The curves on Fig 5b reveal the similar observation: the G_{MPDU} is extremely high when $AI = 1$, then it quickly drops and gradually converges to a constant. Overall, except for when $AI = 1$, the empirical observation matches our conjecture in Eq (2), (3). The abnormal case on $AI = 1$ is mainly due to that the BA time gap can also be influenced by other factors, e.g., back off, collision and so on. When data size is small under $AI = 1$, the BA gap is seriously disturbed by the other factors. That is why the gap of regular ACK (which is designed to respond one data packet) cannot be used to indicate data transmission time. When AI increases with more data transmitted, the data transmission becomes the dominant factor to decide the BA gap. The influence from the other factors is gradually mitigated. That is why the curves of G_{MPDU} slightly drop while converging to a consistent value.

C. Deriving Further Metrics

Control packets can give high-level information about the WiFi environment, such as the number of APs, the number of clients, and so on. Beyond these metrics, we would argue that our proposed method can provide an insightful set of metrics including *throughput*, *loss*, and *airtime*. In this subsection, we will elaborate how to derive each of these metrics from the earlier two measurements.

Given the control packets collected during a certain time window ω, we can estimate the characterization metrics for this particular window. For throughput and loss, based on the previous discussion on AI, they are relatively straightforward to compute. *Throughput* can be approximated in the form of packet rate by summing up the AI in the time window ($\frac{\sum^{\omega} AI}{\omega}$). Similarly, loss rate can be calculated from the BA bitmap. For the other more complicated metric (i.e., airtime), it requires further processes to make an accurate estimation. Next, we will iterate on the methods to estimate these metrics along with the experimental evaluation. We also study the robustness of the different metrics regarding the setting of window size ω.

Note that the characterization can be perceived on different scales. With traffic captured on different channels, we can report characterization results on each channel. With the multiple APs operating on the same channel, we can further break down the traffic impact onto different APs. For example, the airtime can be estimated separately on each AP. , for the traffic between a pair of nodes, we can further divide them into different links, i.e., uplink and downlink, based on the traffic direction. In our case, the transmission rate and queue length characterization will mainly focus on the link level.

Airtime (a.k.a. channel utilization) describes how much time is occupied by the traffic on a channel. As one of the most important metrics to understand the load on WiFi channel, it is widely used for QoE/QoS based services [14]. This metric can be polled from certain types of AP with special hardware support. However, it is immensely difficult to estimate from the client side due to the severe loss when passively capturing the

data packets as mentioned earlier. Fortunately, exploiting the primitive measurements can help estimate the airtime without data packet.

Since control/management traffic is designed to be ultra light-weight, data transmission usually accounts for the primary airtime cost. Thus our method explicitly focuses on the consumption resulted from data transmission. According to the discuss above, we know that the BA time gap G_{BA} is a good approximate of the transmission time of the A-MPDU data frames. Intuitively, given the BAs captured in a time window ω, summing up the G_{BA} and dividing by ω will give the percent of time consumed by the data traffic. However, with the exception case of $AI = 1$ as discussed, we further filter out the G_{BA} whose AI is 1. Therefore, airtime can be finally estimated as

$$Airtime = \frac{\sum^{\omega} G_{BA}^{AI>1}}{\omega} \quad (4)$$

where $G_{BA}^{AI>1}$ specifically refers to the BA gap where $AI > 1$.

Experimental Evaluation and Improvement: Using the same experimental settings from before, we generated different sizes of TCP flow to cause different airtime costs on the WiFi channel. We varied the flow size from 20KB to 160MB. For each flow size, we kept repeating the flow in a back-to-back manner such that once the flow completes, we restart the flow immediately. Each flow size ran for 10+ seconds. Ground truth fir airtime was polled from the AP kernel via `iw` tool. With the control packets captured from the monitor node, we sliced the packets into continuous windows with a window size of ω. Then, we calculate the airtime for each window according to Eq (4).

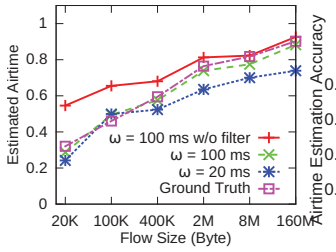


Fig. 6: Estimated airtime v.s. flow size w.r.t. window size ω.

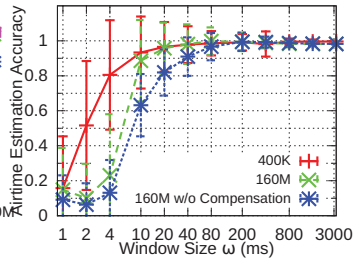


Fig. 7: Estimation accuracy v.s. window size ω w.r.t. flow size.

In Fig 6, we plot the estimated airtime under different flow sizes with respect to two window sizes ($\omega = 100\ ms$ and $\omega = 20\ ms$). In addition, we also plot the estimated result without filtering out the case where $AI = 1$. Note that each point in the figure is the average over all windows. As shown, without the filter, the airtime is significantly overestimated, especially when the flow size is small. After applying the filter, the result closely matches the ground truth. The result implies that ruling out the cases of $AI = 1$ will not harm airtime estimation. Because the occurrences of $AI = 1$ is rare, and it only takes up high percentage when traffic is light. Filtering out $AI = 1$ helps effectively relieve the overestimation caused by the random gaps when $AI = 1$.

Window compensation: Comparing the results from different ω, we notice the small window ($\omega = 20\ ms$) suffers from underestimation, especially when flow is heavy. The reason is that our method requires two consecutive BAs to infer a data frame in the middle. For each window, we are not able to infer the A-MPDU associated with the very first BA. Therefore we always lose one A-MPDU frame when estimating airtime. For example, in the case of Fig 1, with the three BAs collected, we can only infer the second and third A-MPDU but not the first one. When the window size is small and traffic load is heavy, the one A-MPDU loss becomes significant. In order to compensate for this loss, we assume the lost A-MPDU is exactly same as its adjacent A-MPDU inferred from the first pair of observed BAs. Thus, the airtime estimation will double count the first BA gap.

After applying the compensation mechanism, we re-plot the estimation accuracy ($1 - |\frac{estimated - ground\_truth}{ground\_truth}|$) for different window size ω in Fig 7. To reduce figure clutter, we selectively plot the cases of flow size 400KB and 160MB. In addition, we also plot the contrast case without the window compensation for 160MB flow. Overall, we can see that the result is unacceptably poor when window size is too small (i.e., $\omega < 20\ ms$). The reason is that the control packets distribution in such small scale is too random to be statistically meaningful. Many windows (10%-30%) in this case fail to capture even one packet. When the window size increases, this randomness is gradually smoothed out and the results eventually converge. With compensation enabled, we can achieve at least a 90% estimation accuracy when $\omega \geq 20\ ms$ which we note approaches WiFi scanning intervals.

IV. Performance Evaluation

Trace-Driven WiFi Scan Emulation: Modifying the scan function to implement the characterization on commodity devices is impractical, since the functionality of scan is programmed on the firmware. In this paper, we take the emulation approach to evaluate proposed system in a large scale setting. With the real world data captured through the monitor mode, we can conduct trace-driven evaluation upon the data. In the captured dataset, we assume that all the devices can perform such a modified scan function. When a probe request was sighted when a client was scanning, we assume the client was also performing the traffic characterization on WiFi channel. So the control packets collected in the following ω time window after the probe request will be used to calculate characterization result. We set $\omega = 20\ ms$ to satisfy the realistic setting for dwell time. In addition, we assume there is a crowdsource server which can gather results from the devices. So we can combine the results from multiple clients to have a more complete view of the WiFi channel. With more clients contributing, the more accurate and complete result we can get for the WiFi channel.

Settings: Next, we set up a controlled WiFi network to capture the trace for emulation. This network was deployed on campus to provide Internet access for a football tailgating party. The event was hosted in an outdoor tent where several

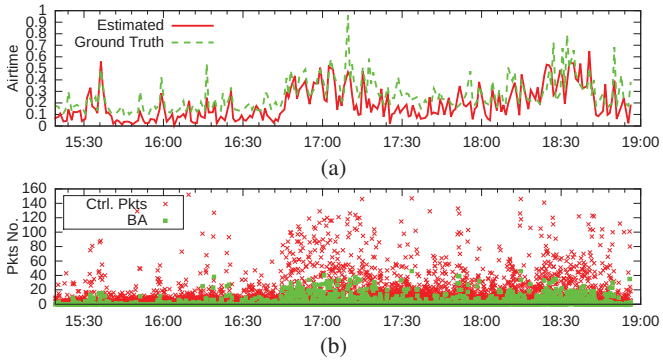


Fig. 8: Time series of (a) airtime estimated compared with ground truth and (b) number of packets captured during dwell time for each scan.

hundred people gathered for several hours before a football game. An Aruba 7010 wireless controller to manage multiple APs with the APs providing connectivity on multiple bands with one channel on each band: channel 1 for 2.4GHz (20 MHz) and 149 for 5GHz (80 MHz). By using the controller API, we obtained *ground truth* information through SNMP (Simple Network Manage Protocol) by periodically (i.e., every 45 seconds in this case) polling the controller with the SNMP `walk` command to record network condition, e.g., airtime, throughput and so on. To capture the raw data for characterization, we set up monitoring devices to listen on the two channels on which APs were operating. The monitor devices were placed near the APs in order to attempt capture all traffic on APs and to avoid hidden terminal issues. The captured traffic was saved into `pcap` files as the source for trace-driven emulation and will be made available as part of this paper publication (initial header bytes only).

TABLE I: Data Statistics Summary

	Campus	
	2.4GHz	5GHz
Time Duration	4+ hrs	
Data File Size	1.2GB	7.4GB
# of Sighted Clients	19,116	33,134
# of Active Clients	500	284
# of Block ACK	181,164	2,636,317

Data Summary: The data captured is summarized in Tab I. Over the four hour data collection, we gathered a total of 8.6GB of `pcap` files for only control traffic. There were over 51,000 unique devices[4] sighted during the study. However, many devices only showed up for a short time in part largely inflated by MAC spoofing for anonymization purposes. Among the devices, 480 clients (350 on 5GHz and 130 on 2.4GHz) were persistently sighted for over an hour. There were 784 active clients which launched data transmission with BA exchanges. We find that over 99% of the BA exchange involved our deployed APs. Therefore, we argue that the SNMP data should provide a reasonable ground truth about the WiFi environment.

Evaluation with Ground Truth: Among the characterization metrics derived from our method, the SNMP data provides

[4]A device is identified with a unique MAC address.

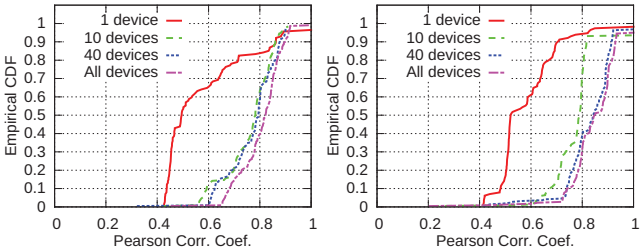


Fig. 9: Correlation between estimated value and ground truth w.r.t. number of devices used for generating result. All the results are from 5GHz band.

the ground truth for two metrics: *throughput* and *airtime*. Therefore, we focus our evaluation on these two commonly-used metrics. Since the SNMP data is sampled every 45 seconds, to create point-to-point matching data, the mean of the results from our method during the 45-second window is used to match with SNMP data. First, by assuming all devices report their characterization results to the crowdsource server, we exploit all scans to plot the time series in Fig 8. Since the usage of 2.4GHz band is sparse in this case, only 5GHz band data is used for this evaluation. In Fig 8a, we see that the estimated *airtime* closely follows the ground truth over time. As the throughput shows the similar pattern, we do not plot it to save space. Furthermore, in order to understand the computation cost of the characterization function, we plot the total number of control packets as well as the BAs captured in the dwell time during a scan in Fig 8b[5]. It shows that for each scan on a channel, the characterization only need to process 20-40 control packets with less than 20 BAs. When the channel is busy, the number can reach to 140 total packets with 40 BAs. It implies implementing the characterization onto scan will not cause significant computation load.

To further evaluate performance, we calculated the Pearson correlation coefficient[6] between the estimated value and the ground truth. A correlation value is calculated from a 90-minute window. By sliding this window across the entire time session, we can obtain over 200 estimation points.

By default, we include all the traffic data from the surrounding devices by assuming a perfect scenarios that all devices are willing to share their data. In addition, we also consider the impact of data sparsity that we are only able to use data from a number of voluntary devices (i.e., in a crowdsource manner). By varying the number of users whose data is used, we try to find out how many voluntary devices are needed to delivery accurate characterization. We sort the sighted devices by their presence time. Thus the x devices means the top x sighted devices. In Fig 9, we plot the empirical CDF of the correlation coefficient on both airtime (Fig 9a) and throughput (Fig 9b). As the results show, with all the available users' data, we can achieve the median coefficient of 0.817 for airtime and 0.837 for throughput. With less devices involved, the coefficient

[5]The figure is plotted after down-sampled to reduce visual clutter.

[6]The value ranges from -1 to 1, where 1 implies perfect linear relationship while -1 implies negative linear relationship.

decreases. Notably, once the client number increases to 10, the median correlation coefficient can reach up to 0.8 for both airtime and throughout.

V. Related Work

Notably, WiFi traffic characterization has been approached from multiple layers of the network stack. At the physical layer, there are works [15]–[17] that particularly target sensing the power on WiFi spectrum. For example, Airshark [15] exploits commodity WiFi device to detect non-WiFi interference (e.g., microwave oven). In contrast to physical layer works, our work primarily focus on the MAC layer. Works that focus on the MAC layer can be generally classified into two groups: 1) AP side approaches and 2) client side approaches. AP side approaches [6]–[8] usually require deploying controlled networks which can be expensive in terms of financial cost as well as human effort. For example, WiSe [8] set up home WLAN for 30 homes over 6 months to analyze their home wireless environments. More recently, [6] studied the data collected from approximately ten thousand radio access points and 5.6 million clients from one-week periods to provide a deeper understanding of real world network behaviour.

To avoid the cost of the AP-side approach, client-side approaches have the advantage of being cost efficient and flexible. Passive client monitoring works have been extensively exploited for different purposes. [18] manipulates smartphones to monitor WiFi traffic in order to infer the human activities. Another smartphone-based work [19] uses indoor WiFi monitor fused with mobile crowdsourse to achieve better localization. With large-scale studies, [20] explicitly seek the co-location between mobile users via Bluetooth information combined with WiFi scan. In addition, for security purpose, [21], [22] attempt to exploit WiFi monitor to detect rogue AP and potential attack. For the general purpose of traffic characterization, many prior works [9]–[12] focus on merging traffic or inferring unobserved traffic. Critically, many assume that most of the data traffic can be sensed passively which may not be accurate with more modern, faster WiFi approaches.

VI. Conclusion

In this paper, we presented an intriguing new approach for WiFi traffic characterization. We showed that it is possible to infer a variety of useful characterization metrics solely through the observation of Block Acknowledgements and other control packets. Moreover, we showed that such results tend to be reasonably stable even at very short time frames allowing for the potential to conduct such observations during normal WiFi scanning. We believe the implications for the work are considerable from both the end user standpoint, troubleshooting standpoint, and analysis / potential of cellular onto WiFi bands standpoint. We believe the topic is ripe for future work and plan to explore more extensive datasets including dense urban centers, newly deployed WiFi at dense environments such as stadiums, and various public venues.

References

[1] K. Fukuda and K. Nagami, "A measurement of mobile traffic offloading," in *Proc. of PAM*, ser. PAM'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 73–82.

[2] M. Li, M. Claypool, and R. Kinicki, "Wbest: A bandwidth estimation tool for ieee 802.11 wireless networks," in *Proc. of LCN*. IEEE, 2008, pp. 374–381.

[3] A. Farshad, M. Lee, M. K. Marina, and F. Garcia, "On the impact of 802.11n frame aggregation on end-to-end available bandwidth estimation," in *SECON*. IEEE, Jun. 2014, pp. 108–116.

[4] A. S. Lixing Song, "Leveraging Frame Aggregation for Estimating WiFi Available Bandwidth," in *SECON*. IEEE, Jun. 2017.

[5] J. Huang, C. Chen, Y. Pei, Z. Wang, Z. Qian, F. Qian, B. Tiwana, Q. Xu, Z. Mao, M. Zhang *et al.*, "Mobiperf: Mobile network measurement system," *Technical Report. University of Michigan and Microsoft Research*, 2011.

[6] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 153–165, Aug. 2015.

[7] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda, "Characterizing and improving wifi latency in large-scale operational networks," ser. MobiSys '16. New York, NY, USA: ACM, 2016, pp. 347–360.

[8] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through WiFi APs," in *Proc. of MobiCom*. New York, NY, USA: ACM, 2013, pp. 339–350.

[9] J. Yeo, M. Youssef, and A. Agrawala, "A framework for wireless lan monitoring and its applications," in *Proc of WiSe*, ser. WiSe '04. New York, NY, USA: ACM, 2004, pp. 70–79.

[10] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the mac-level behavior of wireless networks in the wild," in *Proc. of SIGCOMM*. New York, NY, USA: ACM, 2006, pp. 75–86.

[11] A. Mahanti, C. Williamson, and M. Arlitt, "Remote analysis of a distributed WLAN using passive wireless-side measurement," *Performance Evaluation*, vol. 64, no. 9–12, pp. 909 – 932, 2007, performance 200726th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation.

[12] D. Da Hora, K. Van Doorselaer, K. Van Oost, R. Teixeira, and C. Diot, "Passive wi-fi link capacity estimation on commodity access points," in *Traffic Monitoring and Analysis Workshop (TMA) 2016*, 2016.

[13] Cisco. (2015) 802.11ac: The fifth generation of wi-fi technical white paper. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-3600-series/white_paper_c11-713103.html

[14] A. Patro and S. Banerjee, "COAP: A Software-Defined Approach for Home WLAN Management Through an Open API," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 18, no. 3, pp. 32–40, Jan. 2015.

[15] S. Rayanchu, A. Patro, and S. Banerjee, "Airshark: Detecting non-wifi rf devices using commodity wifi hardware," in *Proc. of IMC*. New York, NY, USA: ACM, 2011, pp. 137–154.

[16] T. Zhang, A. Patro, N. Leng, and S. Banerjee, "A wireless spectrum analyzer in your pocket," in *Proc. of HotMobile*. New York, NY, USA: ACM, 2015, pp. 69–74.

[17] Y. Xie, Z. Li, and M. Li, "Precise Power Delay Profiling with Commodity WiFi," in *Proc. of MobiCom*. New York, NY, USA: ACM, 2015, pp. 53–64.

[18] Y. Chon, S. Kim, S. Lee, D. Kim, Y. Kim, and H. Cha, "Sensing wifi packets in the air: Practicality and implications in urban mobility monitoring," in *Proc. of UbiComp*. New York, NY, USA: ACM, 2014, pp. 189–200.

[19] V. Radu, L. Kriara, and M. K. Marina, "Pazl: A mobile crowdsensing based indoor wifi monitoring system," in *Proc. of CNSM*, Oct 2013, pp. 75–83.

[20] S. Liu and A. D. Striegel, "Exploring the potential in practice for opportunistic networks amongst smart mobile devices," ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 315–326.

[21] A. Chhetri and R. Zheng, "Wiseranalyzer: A passive monitoring framework for wlans," in *Proc. of Mobile Ad-hoc and Sensor Networks*, Dec 2009, pp. 495–502.

[22] Y. Sheng, G. Chen, H. Yin, K. Tan, U. Deshpande, B. Vance, D. Kotz, A. Campbell, C. Mcdonald, T. Henderson, and J. Wright, "Map: a scalable monitoring system for dependable 802.11 wireless networks," *IEEE Wireless Communications*, vol. 15, no. 5, pp. 10–18, October 2008.