Sniffing Only Control Packets: A Lightweight Client-Side WiFi Traffic Characterization Solution

Lixing Song, *Member, IEEE*, Aaron Striegel, *Senior Member, IEEE*, and Alamin Mohammed, *Graduate Student Member, IEEE*

***Abstract*—The advancement of the Internet of Things (IoT) is bringing unprecedented convenience into our daily life. However, with the relentlessly increasing number of mobile devices connected to the Internet, the wireless network environment is becoming more crowded than ever before. Particularly, WiFi, with its evolving role in IoT, is shouldering a tremendous amount of traffic from IoT and other mobile devices. As a result, exploding numbers of competing devices, encroachment by cellular technology, and dramatic increases in content richness deliver a more variable Quality of Experience (QoE) on WiFi than desired. Moreover, such variance tends to occur both across time and space making it an extremely difficult problem to debug. Existing active approaches tend to be expensive or impractical while existing passive approaches tend to be too narrow. To conduct efficient and nonobtrusive WiFi traffic characterization, in this article, we propose a novel passive client-side approach that delivers efficient and accurate characterization by taking advantage of the properties of frame aggregation (FA) and block acknowledgment (BA). The devised approach requires only capturing and analyzing certain types of control packets thus making it feasible to deploy on IoT devices that have limited computation power. We show in this article that we can accurately derive important characterization metrics, such as airtime, queuing information, and transmission rates 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.**

***Index Terms*—Crowdsensing and Crowdsourcing, medium access control protocols, mobile and ubiquitous systems, wireless network measurement.**

I. Introduction

PROPELLED by the development of the Internet of Things (IoT), the number of mobile devices dedicated to IoT services is expected to be tripled from 2018 to 2023 according to Cisco Annual Internet Report [1]. Notably, WiFi, with its low cost and wide-spreading deployment, becomes a major channel for IoT devices to communicate. Due to the limited spectrum resource, the overwhelming traffic from countless IoT devices and other end users makes the Quality of Experience (QoE) on WiFi variant and unpredictable. While new standards seek to deliver a "better" WiFi, the increasing density of devices, richness of content, and recent encroachment on WiFi by cellular (LTE-U) make the performance variance on WiFi being the new normal for the foreseeable future.

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

In the archetypal form of wireless characterization, the mobile device becomes an active sensor to actively determine end-to-end performance at the network and transport layers [2]–[4]. Prior works, such as Speedtest.net, iperf3, or Mobiperf [5], embody this approach whereby the currently connected WiFi path is actively probed to determine network performance. While such an approach can provide longitudinal data for a given WiFi link, the cost of such characterization is often quite high both in terms of time and energy. Moreover, such tests also have a negative impact on other users as the probe traffic can be intrusive to existing traffic. Most importantly, 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, other work has operated from the perspective of the access point (AP) to afford a much deeper 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 an AP can ably sense, the entire collection of APs represents a limited and stationary spatial distribution. Thus, such systems tend to focus largely from the perspective of the provided network (a reasonable presumption), potentially missing broader trends in the overall wireless ecosystem

Manuscript received November 2, 2020; accepted November 23, 2020. Date of publication December 1, 2020; date of current version April 7, 2021. This work was supported by NSF under Award 1718405. *(Corresponding author: Lixing Song.)*

Lixing Song is with the Computer Science and Software Engineering Department, Rose–Hulman Institute of Technology, Terre Hauet, IN 47803 USA (e-mail: song3@rose-hulman.edu).

Aaron Striegel and Alamin Mohammed are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: striegel@nd.edu; amohamm2@nd.edu).

Digital Object Identifier 10.1109/JIOT.2020.3041671

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

and potentially missing client-side issues at the edge of the network.

Finally, one last approach to characterization is to view the mobile client itself as a capable 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 question that we pose in this article is: How could we radically improve the ability of a mobile client 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? Most importantly, is there a way that we can do this in an extremely energy-efficient manner, observing for only a brief period of time but still capturing the essence of a given channel? In this article, we seek to demonstrate that we can accomplish these goals. We propose a new technique that builds on the properties of frame aggregation (FA), specifically the block acknowledgment (BA), and show how BA map to a rich set of link characterization metrics, such as airtime, transmission rate, and queuing information through extensive experimental studies. Moreover and perhaps most excitingly, beyond our prior work [13], we show that the stable observation time for BA can be sufficiently satisfied during a normal WiFi scan period (20 ms). It means that we can essentially utilize our method for "free" via *de facto* WiFi scan. The implications of this work are considerable, expanding every mobile device to not only observing the nearby APs but also characterizing the WiFi channel(s). The contributions of this article are as follows.

1) *Sensing With Control Packets:* We show how observing control packets, especially BA, can be used to extract a rich set of WiFi characteristics. We define two important primitive metrics—aggregation intensity (AI) and *BA time gap* and show how these two primitive metrics can be used to compute airtime, transmission rate, and queue length. We demonstrate the accuracy, efficiency, and robustness of these mappings through extensive empirical studies across a wide variety of scenarios.
2) *Robustness Across Short-Time Windows:* We show that only an extremely limited window of control packets is necessary to extract a stable view of the WiFi link for characterization (20 ms). We introduce several key concepts and assumptions necessary to work within such a small time window as well as present extensive experiments to validate our results.
3) *Viability With WiFi Scanning:* We study the feasibility of using the limited window afforded by WiFi scanning for characterization. We analyze extensive pools of WiFi scanning data taken from 41k different devices to show the potential time available for observations via real-world data. We show that the minimum scan time sits roughly at 20 ms (Apple iPhone) while nearly all 90% of devices scan at least every 6.5 min.
4) *Real-World Data Validation:* We validate our approach and its accuracy by conducting experiments on a real-world data set captured during a tailgate involving multiple 802.11ac APs. We conduct trace-driven experiments and show that with only a handful of devices (10), the designed characterization can achieve a high correlation (0.8) with the observed ground truth for airtime and throughput estimation.

The remainder of this article is organized as follows. We start with the intuition and background of leveraging control packets for passive traffic characterization in Section II. Followed by Section III, we detail the methods to derive various characterization metrics based on certain observations extracted from control packets. In Section IV, we present an empirical study to verify the idea of using the WiFi scan for characterization. Then, we demonstrate the results of performance evaluation on real-world empirical data sets in Section V. The related work is discussed in Section VI. Finally, we summarize and conclude this article in Section VII

II. Background and 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 a basic primer on FA and BA3. Next, in Section II-B, we describe how WiFi scanning typically operates and why it provides the potential for characterization.

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 a 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 and low supported rate) cannot collect traffic transmitted with high rates. In addition, with beamforming in advanced WiFi (i.e., 802.11ac wave-2 [14]), 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 the channel.

Fortunately, control packets tend not to suffer from the same issues with respect to (w.r.t.) capture. First, since control packets are designed to be acknowledged by all nearby devices, they are set to be transmitted at the lowest rate with a nondirectional manner. Second, the volume of the control packets is much sparser than the data packets. From a capture standpoint, this implies that the mobile device is unlikely to be

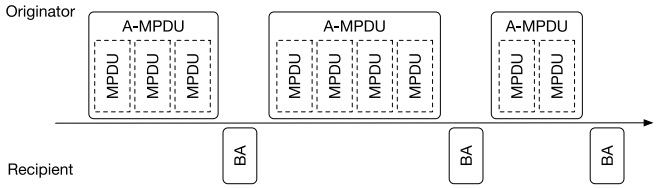


Fig. 1. Illustration of data transmission under FA.

overwhelmed with control packets 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). By leveraging the FA, we posit that the control packets observed on mobile devices can help deliver efficient traffic characterization.

Frame Aggregation: The general principle of FA is to assemble multiple data units to transmit as one aggregated frame.[2] The aggregation can be operated at two levels: 1) aggregate MAC protocol service unit (A-MSDU) and 2) aggregate MAC protocol data unit (A-MPDU). A-MSDU is on the upper MAC layer which can be further aggregated again into A-MPDU when pushing into the physical layer. Therefore, the frame transmitted in the air is eventually expressed in the form of A-MPDU. In this article, we will focus on leveraging A-MPDU. As shown in Fig. 1, in tandem with BA, each A-MPDU only requires one BA to notify the receipts of multiple MPDUs (i.e., packets)[3]. In order to support this one-to-many acknowledgment, a BA uses a bitmap field to explicitly indicate the failure or success of delivery of every single MPDU.

As FA has become the default manner of sending data on modern WiFi (802.11ac), data transmission will always invoke an exchange of a BA. These acknowledgment potentially provide opportunities to infer the data transmissions occurred. In particular, the information stored in the 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 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., the transmission time of a packet. In Section III, we discuss the technical details about how we can manipulate the information to achieve accurate characterization based on control packets.

B. WiFi Scan For Characterization

The core foundation of client-side characterization is the "sniff" function that allows a client can capture the ongoing traffic on a channel. This function is normally implemented as a special mode called the *monitor* mode. However, since monitor mode will suspend all communications, it is not desirable to frequently put a client into this mode for the purpose of characterization. Fortunately, we find that the *WiFi scan* operation innately embodies such a sniff function.

WiFi scan is the operation that a device uses to find nearby APs to associate with. Technically, there are two kinds of scan a device can perform: *passive* and *active* scans. When performing the passive scan, the client radio listens on different channels iteratively for periodic beacons sent from APs. With an active scan, the client broadcasts a probe request on a channel and then waits for the responses (probe response or beacon) from AP(s). This process is repeated on each channel. When waiting on the possible response, the WiFi chip suspends all ongoing communications just like the monitor mode. The difference is that it decodes all the captured packets (at least the header) but only reports the desired ones (beacons and probe responses) to the upper layer. The "monitor" behavior of the WiFi scan provides an intriguing opportunity for the purpose of characterization.

We propose that it would not be untoward to modify the WiFi scan to collect all control packets captured during the scanning process. In this article, we show that one could use these packets to conduct entire channel traffic characterization. The usage of the WiFi scan naturally provides iteration across multiple channels with periodic invocations, potentially giving a continuous view of the WiFi environment. When coupled with crowdsourced information, the humble WiFi now becomes an exciting new opportunity for network characterization.

III. WiFi Characterization via Control Packets

In order to characterize the WiFi channel, as part of the contributions of this article, we design two primitive metrics: 1) the *AI* and 2) the Block Ack (BA) time gap. Based on the two measurements, we can further derive various important characterization metrics, e.g., channel airtime, physical layer transmission rate, queue length, and so on.

A. Primitive Measurements

1) PM1 (Aggregation Intensity): FA allows multiple data units to be assembled into one aggregated frame (A-MPDU) and sent together. The degree of aggregation can be useful and we describe it using a metric called AI that counts the number of MPDUs (i.e., packets) within one aggregated frame. As an example in Fig. 1, the three A-MPDUs have AI of 3, 4, and 2, respectively. 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 the same traffic identification (TID) into a frame where the TID usually indicates the packets destined to the same address. Thus, the more packets with the same TID are held in the queue, the larger AI should potentially be. Furthermore, 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 B) and rarely reached, the AI is usually limited

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

[3]Packet and MPDU are used interchangeably for the remainder of this article.

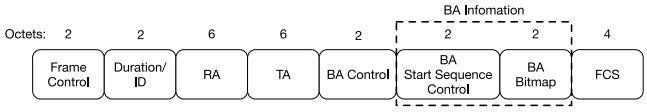


Fig. 2. Format of Block ACK frame.

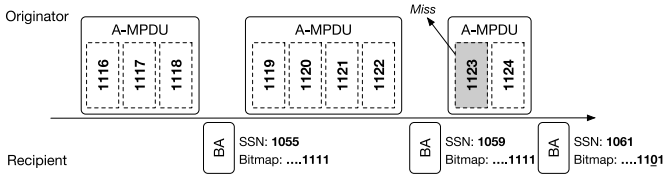


Fig. 3. Data transmission under FA. The sequence number (SSN) and bitmap are indicated. A 64 bitmap is used in this case.

by the transmission time. For a certain packet size P, we have $T_{\max} \leq ([\mathrm{AI} \cdot P]/R)$ where R is the transmission rate. Thus, the maximum AI can be expressed as

$$\mathrm{AI}_{\max} = \frac{R \cdot T_{\max}}{P}. \tag{1}$$

We see that the AI is a function of multiple traffic factors, including queue length, transmission rate, and other factors. As we will present later, the distribution of AI can be used to effectively reveal traffic conditions.

Extracting AI From BA: Computing AI relies on two important fields in the BA frame: the starting sequence control (SSC) and the bitmap as shown in Fig. 2. Each bit on the bitmap represents the receiving status (success/failure) of an MPDU. The SSC includes a subfield called 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[4] of the corresponding A-MPDU (sent from B to A). For example, in Fig. 3, we replot 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 as 4 (1059 - 1055).

Experimental Evaluation: We set up a 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 a 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 2.4 GHz with 20-MHz bandwidth) to allow us

[4]The loss information can provide additional measurements about network traffic but we will focus on AI in this article and leave loss information for future exploration.

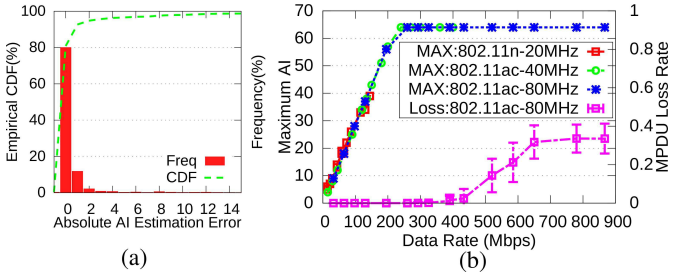


Fig. 4. (a) AI estimation error. (b) Maximum AI and loss rate across different transmission rates.

to capture most of the data packets. The ground truth for AI can be obtained from the A-MPDU reference number in the radiotap header. Overall, we collected over 25 000 A-MPDUs and their BAs. In Fig. 4(a), we plot the cumulative distribution function (CDF) and the frequency of the absolute AI estimation error (|ground truth—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 (1). We repeated the experiment above under different transmission rates with various bandwidth settings (20, 40, and 80 MHz) on both 2.4 GHz (802.11n) and 5 GHz (802.11ac). As the result shown in Fig. 4(b), the observation matches (1) very well that for all settings, the maximum AI linearly increases with the transmission rate until reaches the maximum 64. This maximum is decided by the compressed BA used in this case. Moreover, we also plot the loss rate (number of losses in a time unit) of 80-MHz bandwidth on 802.11ac. Given the static setting of the AP and the client, the channel quality (e.g., signal-to-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 Mb/s, the loss rate starts to increase. Overall, the experimental results show that the designed method can help accurately capture the AI and loss across different network settings.

2) PM2 (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 primitive measurement (PM)—*BA time gap*. We define the BA time gap as the time gap between a BA and its *previously transmitted control packet* on a channel. Notably, any control packet can suffice with us particularly noting that all control packets tend to be unencrypted. Normally, 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, BA usually follows a CTS. Once the control packets are captured, the BA gap can be simply computed by subtracting the packet timestamps recorded by the network adapter.

Since BA is designed to follow closely 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,

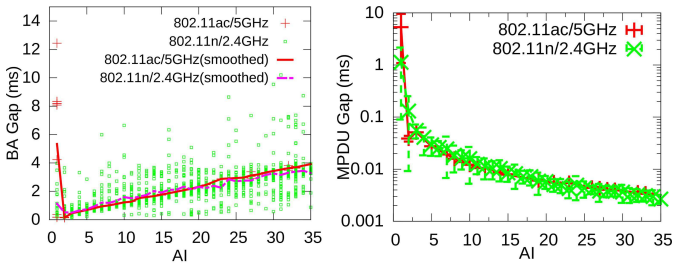


Fig. 5. (a) G_{BA} and (b) G_{MPDU} versus AI given certain transmission rate (i.e., 144.4 Mb/s).

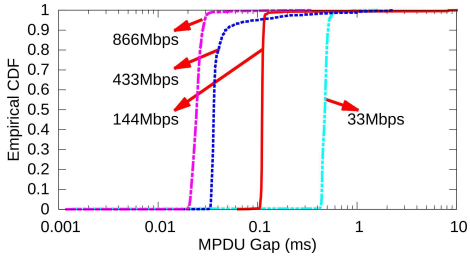


Fig. 6. CDF of MPDU gap (when AI > 1) w.r.t. transmission rate.

the larger the BA gap should be. The relationship can be expressed as

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

where R is the transmission rate and P denotes the packet size. $\mathrm{AI} \cdot P$ 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

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

Once AI is estimated with the method mentioned above, G_{MPDU} can be computed. From (2) and (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}.

Experimental Validation: Using the prior experiment settings, we fixed the transmission rate on 144.4 Mb/s and repeated the TCP traffic on both 802.11ac 5 GHz and 802.11n 2.4 GHz. Fig. 5(a) and (b) plots 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. 5(a), G_{BA} linearly increases with AI for all cases except for $\mathrm{AI} = 1$. With the same transmission rate, the observation from 2.4 GHz is identical to on 5 GHz. The curves on Fig. 5(b) reveal a similar observation: the G_{MPDU} is extremely high when $\mathrm{AI} = 1$, then it quickly drops and gradually converges to a constant. Overall, except for when $\mathrm{AI} = 1$, the empirical observation matches our conjecture in (2) and (3). The abnormal case on $\mathrm{AI} = 1$ 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 the data size is small under $\mathrm{AI} = 1$, 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 of the other factors is gradually mitigated. That is why the curves of G_{MPDU} slightly drop while converging to a consistent value.

According to (3), G_{MPDU} should be inversely proportional to the transmission rate given a certain packet size. We continue the validation by varying the transmission rates with fixed packet size (i.e., MTU). In Fig. 6, we plot the empirical CDF of G_{MPDU} under four different transmission rates, including 33, 144, 433, and 866 Mb/s. Because of the inconsistent behavior under $\mathrm{AI} = 1$ as discussed above, for the remainder of this article, G_{MDPU} will be calculated by filtering out the gaps whose $\mathrm{AI} = 1$. The result in Fig. 6 matches the expectation of (3) that the higher transmission rate is the smaller G_{MPDU} is observed. In addition, for certain transmission rates, G_{MPDU} shows a highly concentrated distribution with marginal deviation. This robustness provides the potential to infer the transmission rate from G_{MPDU}. In the next section, we will elaborate how to exploit the PMs to deliver comprehensive traffic characterization regarding to various metrics.

B. Deriving the Characterization 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 a more insightful set of characterization metrics, including *throughput*, *loss*, *airtime*, *transmission rate*, and *queue length*. In this section, we will elaborate how to derive each of these metrics from the PMs.

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 the packet rate by summing up the AI in the time window $([\sum^{\omega} \mathrm{AI}]/\omega)$. Similarly, the loss rate can be calculated from the BA bitmap. For the other more complicated metrics (i.e., airtime, transmission rate, and queue information), they require further processes to make an accurate estimation. Next, we will iterate on the methods to estimate these metrics along with the experimental evaluation. Particularly, we 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. Furthermore, 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

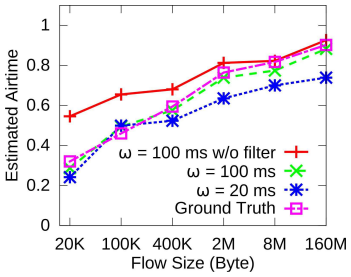


Fig. 7. Estimated airtime versus flow size w.r.t. window size ω.

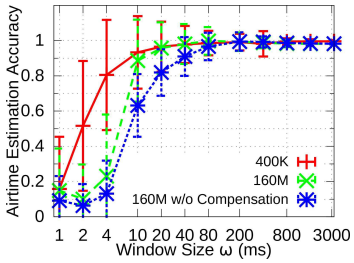


Fig. 8. Estimation accuracy versus window size ω w.r.t. flow size.

rate and queue length characterization will mainly focus on the link level.

1) Airtime: Airtime (also known as 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 [15]. 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 PMs can help estimate the airtime without a data packet.

Since control/management traffic is designed to be ultra lightweight, data transmission usually accounts for the primary airtime cost. Thus, our method explicitly focuses on the consumption resulted from data transmission. According to the above discussion, 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

$$\text{Airtime} = \frac{\sum^{\omega} G_{\text{BA}}^{\text{AI}>1}}{\omega} \tag{4}$$

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

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

In Fig. 7, we plot the estimated airtime under different flow sizes with regarding to two window sizes ($\omega = 100$ ms and $\omega = 20$ ms). In addition, we also plot the resulted calculated without filtering out 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 $= 1$ are rare, and it only takes up a 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 the flow is heavy. The reason is that our method requires two consecutive BAs to infer a data frame in the middle. Therefore, we are not able to infer the A-MPDU associated with the very first received BA. In that case, 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 the traffic load is heavy, this one A-MPDU loss becomes significant. In order to compensate for this loss, we assume the lost A-MPDU is exactly the same as its adjacent A-MPDU inferred from the first pair of observed BAs. So the airtime estimation will double count the first BA gap.

After applying the compensation, we replot the estimation accuracy $(1 - |[(estimated - ground\_truth)/(ground\_truth)]|)$ for different window size ω in Fig. 8. To reduce figure clutter, we selectively plot the cases of flow size 400 kB and 160 MB. In addition, we also plot the contrast case without the window compensation for 160-MB flow. Overall, we can see that the result is unacceptably poor when the window size is too small (i.e., $\omega < 20$ ms). The reason is that the control packets distribution in such a 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 the compensation enabled, we can achieve at least 90% estimation accuracy when $\omega \geq 20$ ms.

2) Transmission Rate: The transmission rate is the data rate used to send a frame in the physical layer. It is useful to understand and diagnose the performance on WiFi (e.g., load balance [16]). But this information normally is not readily available at the AP side as well as the client side. Fortunately, with the properties of the MPDU gap (in Fig. 6), we can use it to make a reasonable inference to the transmission rate. However, although the MPDU gap is primarily decided by the transmission rate, it can also be influenced by other factors, including flow size, protocol/band (802.11n on 2.4 GHz versus 802.11ac on 5 GHz), interference, and so on. In order to make G_{MPDU} a reliable transmission rate indicator, it must be resistant to the influence of the other factors.

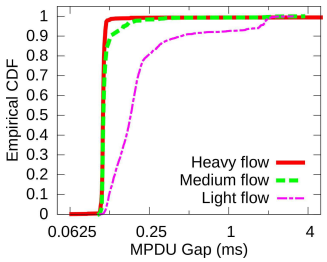


Fig. 9. CDF of G_{MPDU} w.r.t. flow size.

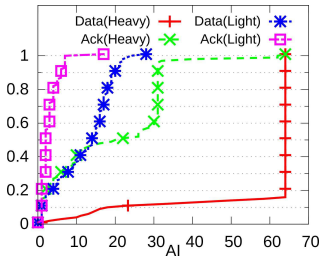


Fig. 10. CDF of AI on data or Ack stream w.r.t. flow size.

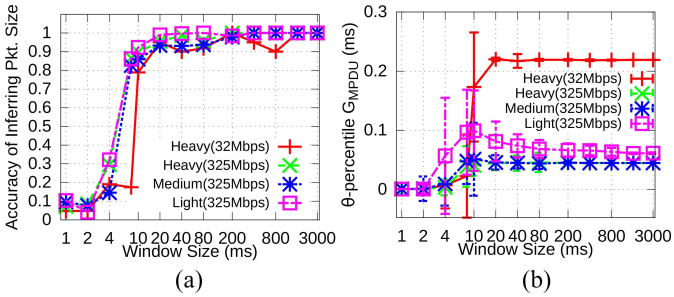


Fig. 11. Impact of window size ω on the two components of estimating transmission rate. (a) Inferring packet size. (b) Measuring G^{θ}_{MPDU}.

Through our empirical study from experiments, we find that *flow size* cause the most significant impact on the MPDU gap distribution in addition to the transmission rate. The influence of other factors is marginal. To demonstrate this impact, we use the trace from previous experiments to study the distribution of the MPDU gap under different flow sizes. In Fig. 9, we plot the CDF of G_{MPDU} under 802.11ac 5 GHz with fixed transmission rate (130 Mb/s). The flow size of heavy, medium, and light refer to 160-MB flow, 8 MB, and 400 kB, respectively. For each flow size, we collected over 2000 BAs to generate the result. We see that when the flow size decreases, the MPDU gap becomes larger. It is because with the less intensive traffic from the light flow, the gaps can get much looser compared to under heavy flows where frames are tightly pushed together.

θ-Percentile G_{MPDU}: A closer look into Fig. 9 reveals that even though the overall distribution varies across different flow sizes, the variation is only significant on the high values (e.g., above 90-percentile). While the gaps do not vary a lot at the low percentiles (e.g., below 10-percentile). Therefore, we define the θ-percentile MPDU gap measured for a link within a time window as the effective MPDU gap to infer the transmission rate. Note that by doing this, we assume the transmission rate on this link during the window time is consistent. By properly setting θ, we can alleviate the impact resulted from flow variation. The optimal setting of θ will be explored through extensive experiments later. Once the effective G_{MPDU} is calculated, we can estimate the transmission rate by

$$\text{Rate} = \frac{P}{G^{\theta}_{\text{MPDU}}} \tag{5}$$

where P is the packet size and G^{θ}_{MPDU} is the θ-percentile MPDU gap observed in a window.

Inferring the Packet Size: Unfortunately, without accessing data packets, we cannot determine the packet size. We posit that with several assumptions, one can give a fair approximation for packet size. First, given the pervasive nature of TCP traffic, we consider only two types of packet sizes: 1) MTU for data packets and 2) a fixed small size for TCP Ack packets. Thus, the task of inferring packet size can be reduced to select one size from the two. Second, for the traffic between a pair of nodes, we assume one direction (i.e., link) is consistently being data stream and the other is Ack stream[5] during the time window. To decide which link is the data stream, we argue that the data stream always has higher AI than the Ack stream. Because Ack packets are small and sparse in time, they normally cannot cause a high degree of aggregation. To validate this assumption, we collected the experiment trace[6] to plot the AI distribution from data or Ack stream under different flow sizes. In Fig. 10, we see that for a certain flow, the AI of the data link is always greater than of the Ack link. This difference becomes more evident when the flow size increases. Therefore, by selecting the link with a higher average AI, we can determine the data link and assign it with the MTU packet size. Because of the light and sparse traffic on the Ack link which makes gaps fluctuate, we normally choose to the data link to estimate the transmission rate. Eventually, the rate can be computed from (5).

Experimental Evaluation: With the same experiment setting as used earlier, we varied the transmission rate to evaluate the performance under different settings. We focus on 802.11ac on 5 GHz, since it is more challenging to estimate the rate with the wider range of values (up to 877 Mb/s) compared to 802.11n 2.4 GHz. We start with investigating the impact of window size ω. The performance of transmission rate estimation is decided by two components: 1) inferring packet size and 2) measuring G^{θ}_{MPDU}. Fig. 11 plots the impact of ω on the two aspects under different rates and flow sizes. We tentatively chose $\theta = 10$ for G^{θ}_{MPDU} in this case. To compute the accuracy of inferring packet size, if the proposed method makes the right choice to identify data and Ack stream in a window, we call it a correct inference. Then, the accuracy is the percentage of correct inference over all cases. As we can see in Fig. 11(a), regardless of transmission rate and flow size, the accuracy can reach above 90% once $\omega \geq 20$ ms. The similar pattern is observed in Fig. 11(b) that the measured gaps start to be consistent from $\omega = 20$ ms. Combining the result

[5]If only one direction traffic is detected, we assume it is the data stream.

[6]The traces are selected from 802.11ac 5 GHz under fixed transmission rate 325 Mb/s.

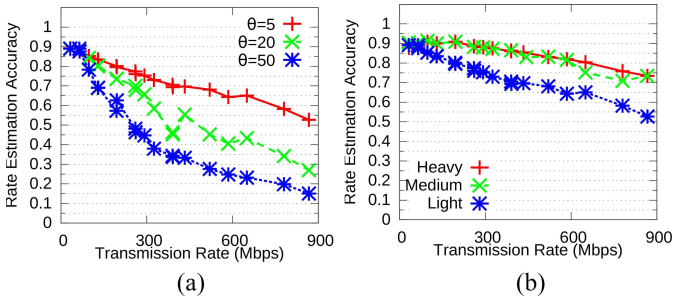


Fig. 12. Transmission rate estimation accuracy from (a) different θ and (b) different flows.

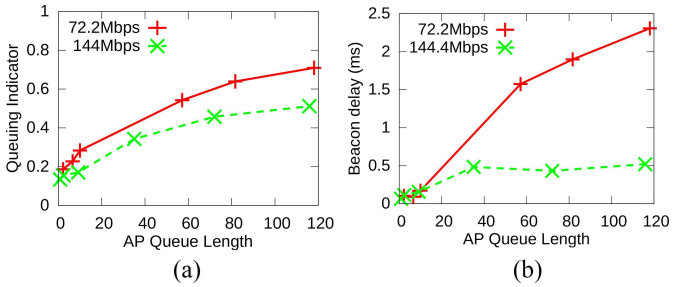


Fig. 13. (a) Queuing Indicator performance compared with (b) beacon delay [20].

from the previous experiment in Fig. 8, it implies that the window size needs to be at least 20 ms to yield a reliable result. Moreover, from Fig. 11(b), we see that the low rate (32 Mb/s) clearly has larger G_{MPDU} than the higher rate (325 Mb/s) as expected. However, a comparison across different flows shows that light flow presents slightly larger values than the medium and heavy flows. This is due to the inappropriate setting of θ. In the following, we continue the experiments to search the optimal setting of θ.

θ Setting: From the observation from previous experiments in Fig. 9 and 11(b), we see that the *light* flow is the most troubled case to obtain effective θ-percentile G_{MPDU}. So we chose the light flow with $\omega = 20$ ms as the target to search θ. Fig. 12(a) shows the transmission rate estimation accuracy $(1-|[(estimated - ground\_truth)/(ground\_truth)]|)$ from three ω settings. We see the estimation accuracy drops with the transmission rate increasing. It is because when the MPDU gap is small under the high transmission rate, it is more vulnerable to be bothered by random noises (e.g., back off and time skew). From the three settings of θ, we see that $\theta = 5$ yields the best performance. The large values of $\theta = 20, 50$ suffer from the overestimated gap under the light-flow traffic (recall Fig. 9).

With the $\theta = 5$, we evaluated the performance of transmission rate estimation across different flow sizes. As shown in Fig. 12(b), when flow size increases, the estimation accuracy goes up benefited from the robust gap measured from heavy traffic. Overall, we can achieve the estimation accuracy above 50% under 802.11ac/5 GHz. In the low transmission rate range (< 300 Mb/s), the accuracy can be improved to at least 75%. Given that this measurement runs at a per-packet basis, this result is reasonably good to offer insightful characterization to the traffic profile in a WiFi environment. In the following part, when discussing the next metric, we will show how we can further assess the confidence of a rate estimation result by using the queue metric.

3) Queuing Indicator: Queuing can affect network performance in various ways [17], [18]. Understanding queuing information can help one diagnose or improve network conditions. This information requires to access the kernel level on the device. It is difficult to access from the client side, especially with a passive approach. Fortunately, the PM AI innately embodies property to infer queue length. Recall the process of forming an aggregated frame, the number of MPDUs will be assembled in an A-MPDU—AI—is decided by the number of MPDUs with the same TID (i.e., same destination address) in the queue. It implies AI is strongly correlated with the queue length. To approximate the queue length, we devise a metric—*queue indicator* (QI) such that

$$\text{QI} \triangleq \frac{\text{AI}}{\text{AI}_{\max}}. \tag{6}$$

Since different transmission rates allow different $\text{AI}_{\max}$ (1), the same AI may indicate different queuing degree under different transmission rates. Therefore, we define QI as a normalized AI. The value is between 0 and 1 which describes how much the aggregation potential is being utilized. It can translate how intensive is the backlog queuing effect. For the traffic in a time window, QI can be computed for each link. The AI then can be represented as the mean AI on a link, and maximum AI can be calculated from the transmission rate according to (1). By plugging in (5) into (1), we can compute $\text{AI}_{\max}$ as

$$\text{AI}_{\max} = \frac{T_{\max}}{G^{\theta}_{\text{MPDU}}} \tag{7}$$

where $T_{\max}$ is the maximum transmission time allowed. It is set by WiFi adapter manufacture (e.g., 4 ms in `ath9k`[19]).

Experimental Validation: Since our devised metric is an indicator of the queue length, the output will be a correlated reference rather than the length of the hardware queue. In order to evaluate the performance, we varied the TCP flow sizes to cause different queue lengths. Similar to the setting in Fig. 7, six flow sizes were used in this experiment ranging from 20 kB to 160 MB. The ground-truth queue length can be polled from the Linux kernel. As the performance of QI largely depends on the robustness of G^{θ}_{MPDU}, the impact of window size θ is similar to the previous case. Thus, we skip the ω impact study, and set $\omega = 20$ ms as the minimal feasible value. In Fig. 13, we plot the performance of QI contrasted with a prior work [20]. Reference [20] uses the similar passive approach to infer queue length. But it exploits the delay of the beacon frame from AP as the indicator. The designed QI [Fig. 13(a)] monotonically increases with the queue length regardless of the transmission rate. However, the beacon delay [Fig. 13(b)] is only responsive on low transmission rate (72.2 Mb/s). When the transmission rate is high (144.4 Mb/s), beacon delay is not sufficiently sensitive to capture the subtle changes on the queue length.

Facilitating Transmission Rate Estimation: In addition, QI can also serve as a confidence metric for facilitating transmission rate estimation. When the serious queuing effect occurs,

TABLE I
CHARACTERIZATION METRICS SUMMARY

Metric	Derivation Formula	Note
Airtime (Section III-B1)	$\frac{\sum^{\omega} G_{BA}^{AI>1}}{\omega}$	$G_{BA}^{AI>1}$ specifically refers to the BA gap where $AI > 1$; ω is the measure window;
Transmission Rate (Section III-B2)	$\frac{P}{G_{MPDU}^{\theta}}$	P is the inferred packet size based on the traffic flow direction; G_{MPDU}^{θ} refers to the θ percentile value among the measured per-MPDU gaps $G_{MPDU} = \frac{G_{BA}}{AI}$;
Queuing Indicator (Section III-B3)	$\frac{AI \cdot G_{MPDU}^{\theta}}{T_{max}}$	T_{max} is is the maximum transmission time allowed, which is a constant defined by the WiFi adapter;

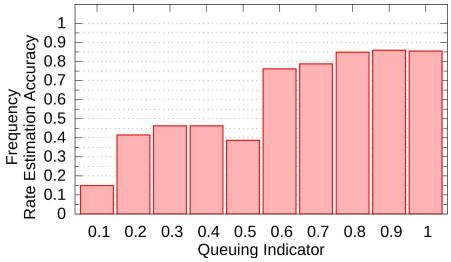


Fig. 14. Queuing indicator versus frequency of relative estimation accuracy of transmission rate.

the frames are transmitted in a more compact way. Thus, the time gap can be more robust under high QI. Therefore, we can use QI to assess the confidence about the estimated transmission rate. To demonstrate this property, we replot the accuracy of transmission rate estimation as the function of QI from previous experiments. In Fig. 14, we see that QI can intrinsically reveal the estimation accuracy for transmission rate estimation. When QI is greater than 0.5, the accuracy can hold at least 75%. The feature can help us effectively filter out the erroneous rate estimations.

4) Characterization Metrics Summary: We summarize the derivation formulas for calculating different metrics based on the PMs AI and G_{BA} in Table I. The table also provides explanations to the measurement variables used in each equation. The detailed derivation and evaluation for each metric can be found in the corresponding section labeled in the table.

IV. WiFi Scan for Characterization

Thus far, we have demonstrated that how one can characterize WiFi channel traffic by merely using the control packets. More importantly, the experimental result shows that we can archive reliable characterization toward various metrics even under a short time window (e.g., 20 ms). As one of the contributions of this article, we propose that the characterization design can be implemented on the existing WiFi scan function by taking advantage of its periodic behavior of listening on WiFi channels. By using the control packets captured during the scan, we are able to accomplish characterization on WiFi channels over time without triggering an extra process.

However, the performance of adapting WiFi scan to characterize is further decided by 1) *how frequently* the scan is performed and 2) *how long* it listens on a channel every time. If the scan is rarely performed, the results over time can be sparse and can suffer from staleness. In addition, if the duration that a scan keeps listening on a channel is not sufficiently long, the characterization result might be inaccurate as discussed before when ω is too small. In order to validate whether the *de facto* scan operation is adequate to be adapted for the characterization purpose, we conduct real-world analysis to study the scan behavior.

A. Feasibility Study—Dense Data Set

We define the time interval between consecutive scans as the *scan interval*. During each scan, the time duration that the radio keeps listening on a channel is called *dwell time* [21]. The scan interval depends on various factors, such as WiFi connected or not, phone screen on/off, WiFi setting page on/off, and so on. For the dwell time, it is set by the manufacture so that different WiFi chips have different settings. In order to understand the scan behavior in the wild, we conduct the analysis on *scan interval* and *dwell time* from a real-world data set which involves high user diversity.

Data Set Summary: We studied WiFi scanning by drawing from control packets captured in prior work [22]. In order to gain user diversity as well as density, we collected the data on a university football game day in two network scenarios. The first scenario was a tailgating party before the game started, and the second was inside the stadium during the game. Overall, we collected over 272 000 probe requests (139 817 from the stadium and 132 274 from the tent) on one channel (chan no. 1). Notably, over 41 000 WiFi devices (22 434 from the stadium and 19 116 from the tent) contributed to this study.

Scan Interval: For energy efficiency, we know the scan cannot be triggered more than once per second. Therefore, the scan interval can be measured with the time difference between consecutive probe requests that were sent from the same device and set apart more than 1 s. The empirical CDF is plotted in Fig. 15(a). The observation shows that, about half of the scans have an interval of less than 20 s and the 75th and 90th percentiles are 134.90 and 390.73 s. The frequent scan behavior in the data set is due to the fact that 1) many devices often had their screens on (e.g., people watched the game news on their phones) and 2) many of them did not have WiFi connected, especially in the stadium where no WiFi is available.

Dwell Time: In order to measure the dwell time, we exploit the channel leakage from the overlapped channels on the 2.4-GHz band. The intuition is that the packet transmitted on a channel (e.g., channel 2) can be heard on the adjacent channel

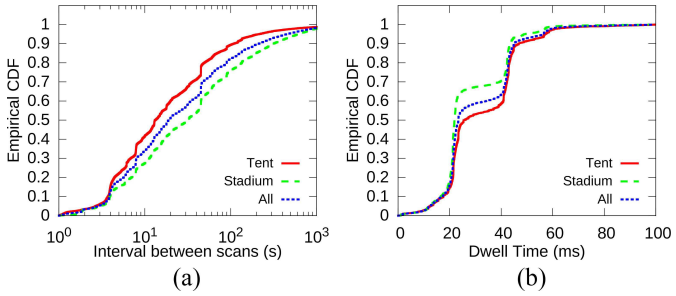


Fig. 15. Empirical CDF of (a) scan interval and (b) dwell time observed in the real-world data set.

(e.g., channel 1). For a probe request packet, the parameter *current channel* indicates its target channel. By measuring the time gap between two consecutive probe requests (with consecutive sequence number) from the same device but target on different channels, we can calculate the dwell time. We plot the empirical CDF of the observed dwell time in Fig. 15(b). The result reveals a clustered distribution that about half of the scans have dwell time of 20 ms and another 40% have 40 ms. By resolving the MAC addresses, we find that Apple and Lenovo devices usually use the 20 ms setting, and Samsung and HTC prefer 40 ms.

Implication: Through the empirical study, we see that for each device, the scan operation is performed at least every few minutes (e.g., 6.5 min for 90% devices). In each scan, the device listens on a channel for at least 20 ms and about half of the devices listen for 40 ms. Recall the experimental results in Section III, the characterization result is robust once the window time $\omega \geq 20$ ms. It implies that if the characterization function is implemented on the existing WiFi scan operation, we can innately obtain the traffic condition on different channels every few minutes. In the next section, we will evaluate the performance of this proposed mechanism under a real-world scenario.

V. Performance Evaluation

Trace-Driven WiFi Scan Emulation: Modifying the scan function to implement the characterization on commodity devices is impractical, since the functionality of the scan is programmed on the firmware. In this article, we take the emulation approach to evaluate the 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 data set, 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 the WiFi channel. So the control packets collected in the following ω time window after the probe request will be used to calculate the 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.

TABLE II
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

Setting: Following that, we set up a controlled WiFi network to capture the trace for emulation. Similar to the tailgating scenario before, the network was deployed on campus to provide Internet access for a football tailgating party. The event was hosted in an outdoor tent where several hundred people gathered for several hours before the football game started. We used Aruba 7010 wireless controller to manage the multiple APs. The APs provide connection on dual bands with one channel on each band: channel 1 for 2.4 GHz and 149 for 5 GHz. By using the controller, we could obtain the *ground-truth* information through the simple network manage protocol (SNMP). We periodically (i.e., every 45 s in this case) polled the controller with SNMP `walk` command to record network condition, e.g., airtime, throughput, and so on. To capture the raw data for characterization, we set up the monitor devices to listen on the two channels which APs were operating on. The monitor devices were placed near the APs to attempt capture all traffic on APs and avoid the hidden terminal issue. The captured traffic was saved into `pcap` files as the source for trace-driven emulation.

Data Summary: The data captured is summarized in Table II. Over the 4-h data collecting, we gathered total 8.6-GB `pcap` files for only control traffic. There were over 51 000 unique devices[7] sighted during the study. But many devices only showed up for a short time. Among them, 480 clients (350 on 5 GHz and 130 on 2.4 GHz) were persistently sighted 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. It means almost all data traffic occurred on our deployed network. Therefore, the SNMP data should provide complete ground truth about the WiFi environment.

Evaluation With Ground Truth: Among the characterization metrics derived from our method, SNMP data only provide ground truth for two metrics: 1) *throughput* and 2) *airtime*. Therefore, we focus evaluation on the two commonly used metrics. Since SNMP data are sampled every 45 s, to create point-to-point matching data, the mean of the results from our method during the 45th s 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. 16. Since the usage of the 2.4-GHz band is sparse in this case, only 5-GHz band data are used for this evaluation. In Fig. 16(a), we see that the estimated *airtime*

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

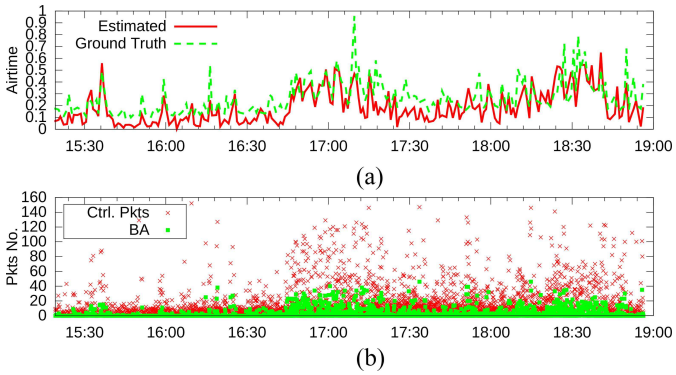


Fig. 16. Time series of (a) airtime estimated compared with ground truth and (b) number of packets captured during dwell time for each scan.

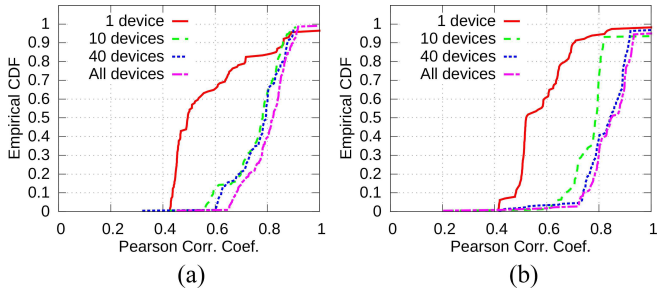


Fig. 17. Correlation between the estimated value and ground truth w.r.t. number of devices used for generating result. All the results are from 5-GHz band. (a) Airtime. (b) Normed throughput.

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. 16(b).[8] It shows that for each scan on a channel, the characterization only needs 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 the scan will not cause significant computation load.

To further quantize the performance, we calculated the Pearson correlation coefficient[9] between the estimated value and ground truth. A correlation value is calculated from a 90-min window. By moving this window across the entire time session, we can obtain over 200 estimation points. In addition, to explore the impact of crowdsourcing, we assume that only part of the clients voluntarily report their data to the crowdsource server. Therefore, only the scans from certain devices are exploited for characterization. By varying the number of devices involved, we try to find out how much clients are needed to deliver accurate characterization through the crowdsource manner. We sort the sighted devices by their present time. Thus, the x devices mean the top x sighted devices. In Fig. 17, we plot the empirical CDF of the correlation coefficient on both airtime [Fig. 17(a)] and throughput [Fig. 17(b)]. As the results show, with all the available devices, we can achieve the median coefficient of 0.817 for airtime and 0.837 for throughput. With less devices involved, the coefficient decreases. Notably, once the client number increases to 10, the median correlation coefficient can reach up to 0.8 for both airtime and throughout. It implies that for a certain WiFi environment, if more than ten devices contribute to their characterization results, we can provide the accurate channel traffic characterization through crowdsourcing. We argue that benefited from the low cost and nonobtrusive nature of our passive approach, it creates less barrier to convince users to contribute into the crowdsourcing. In addition, the characterization result is usually more useful under a user-dense scenario to either glean insightful observation or help debug network problems. Given the proposed method merely requires ten or more clients to get a satisfactory result, it is fairly feasible to get that small number of voluntary clients in a user-dense (ideally with the client number greater than 50) scenarios.

Empirical Study on Other Metrics: Without the ground truth, we cannot provide deterministic performance evaluation for other characterization metrics, e.g., transmission rate. Instead, we use the characterization results from all clients as an empirical study to analyze the WiFi traffic. In Fig. 18, we plot the empirical CDF of the several metrics computed from our method. Particularly, we compare the different distribution from 2.4 and 5 GHz bands. Starting from the AI in Fig. 18(a), since we had much more traffic load on 5 GHz, the AI on 5 GHz is higher than on 2.4 GHz. Notably, there is about 20% of AI equal to 1 on 2.4 GHz which could be due to the many light-flow traffic. The QI in Fig. 18(b) reveals that with the higher capacity, although there were more traffic on 5 GHz, it caused less backlog pressure compared to on 2.4 GHz. Finally, by filtering out the results with $QI \leq 0.1$, we plot the estimated transmission rate in Fig. 18(c). The result matches our expectation that with 802.11ac supported on 5 GHz, the transmission rate on 5 GHz (with the median around 200 Mb/s) is greater than 2.4 GHz (with the median of 100 Mb/s). Overall, we see the rich set of characterization metrics from our method can help reflect insightful attributes about the WiFi traffic.

VI. Related Work

The work of WiFi traffic characterization has been investigated from different network layers. At the physical layer, there are works [23]–[25] particularly target on sensing the power on the WiFi spectrum. For example, Airshark [23] exploits commodity WiFi device to detect non-WiFi interference (e.g., microwave oven). Different from the physical layer works, our work primarily focuses on the MAC layer. The works on the MAC layer can then be generally classified into two groups: 1) the AP-side approach and 2) the client-side approach. The AP side works [6]–[8] usually require deploying controlled networks which is expensive in terms of financial cost as well as human effort. WiSe [8] set up home WLAN for 30 homes over six months to analyze their home wireless environments. More recently, [6] studied the data collected from approximately 10 000 radio APs and 5.6 million clients from one-week periods to provide a deeper

[8]The figure is plotted after downsampled to reduce visual clutter.

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

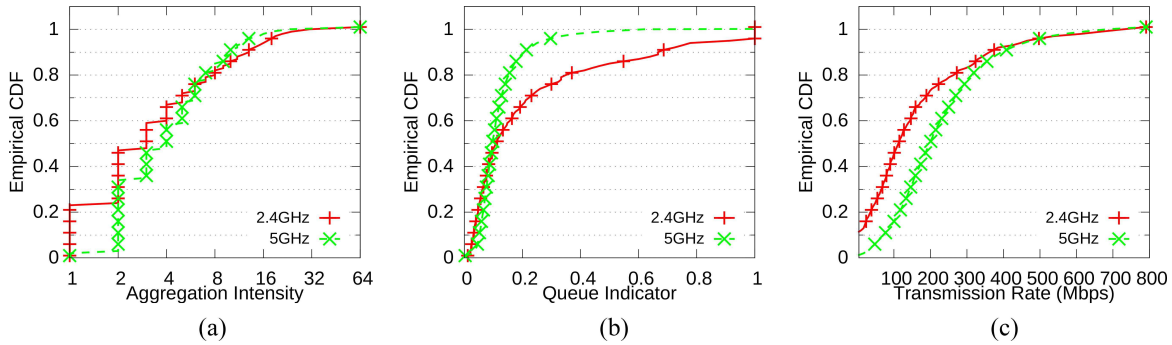


Fig. 18. Empirical observation on other characterization metrics. (a) AI. (b) QI. (c) Transmission rate when QI > 0.1.

understanding of real-world network behavior. For the works focusing on measurements on the transport layer or above, the emphasis is usually on data analysis (e.g., traffic classification [26]) instead of the techniques of collecting measurements which this article focuses on.

To avoid the cost of the AP-side approach, the client-side method shows its advantage of being cost efficient and flexible. Unlike existing works like [27] that collecting mobile traffic only generated from the measuring client itself, the client measurement in this article captures surrounding traffic from other clients in the vicinity to the measuring client, which is intrinsically a more challenging problem. The passive client monitor works in this domain have been extensively exploited for different purposes. Reference [28] manipulates smartphones to monitor WiFi traffic in order to infer human activities. Another smartphone-based work [29] uses an indoor WiFi monitor fused with mobile crowdsource to achieve better localization. With large-scale studies, [30] explicitly seeks the co-location between mobile users via Bluetooth information combined with the WiFi scan. In addition, for security purpose, [31] and [32] 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. They assume that most of the data traffic can be sensed passively which may not be accurate with more modern, faster WiFi approaches.

VII. Conclusion

Given the increasingly crowded WiFi environment due to the growth of IoT devices and other mobile devices, it is important for end users to understand the surrounding WiFi channel condition for making potential adaptive choices. In this article, 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 BA 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 considered 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 data sets, including dense urban centers, newly deployed WiFi at the stadium, and various public venues. We will also further look into novel applications of applying this method to facilitate better network service. For example, one of the promising usages is to leverage the ongoing passive measurements to improve DASH video streaming bitrate adaptation under congested WiFi environments.

References

[1] "Cisco annual Internet report (2018-2023)," Cisco, San Jose, CA, USA, White Paper, Mar. 2018.
[2] M. Li, M. Claypool, and R. Kinicki, "WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks," in *Proc. 33rd IEEE Conf. Local Comput. Netw.*, 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 *Proc. SECON*, Jun. 2014, pp. 108–116.
[4] A. Striegel and L. Song, "Leveraging frame aggregation for estimating WiFi available bandwidth," in *Proc. SECON*, Jun. 2017, pp. 1–9.
[5] J. Huang *et al.*, "MobiPerf: Mobile network measurement system," Univ. Michigan, Microsoft Res., Ann Arbor, MI, USA, Rep., 2011. [Online]. Available: https://www.measurementlab.net/tests/mobiperf/
[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 *et al.*, "Characterizing and improving WiFi latency in large-scale operational networks," in *Proc. MobiSys*, New York, NY, USA, 2016, pp. 347–360.
[8] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through WiFi APs," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2013, pp. 339–350.
[9] J. Yeo, M. Youssef, and A. Agrawala, "A framework for wireless LAN monitoring and its applications," in *Proc. 3rd ACM Workshop Wireless Security*, New York, NY, USA, 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. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, New York, NY, USA, 2006, pp. 75–86.
[11] A. Mahanti, C. Williamson, and M. Arlitt, "Remote analysis of a distributed WLAN using passive wireless-side measurement," *Perform. Eval.*, vol. 64, nos. 9–12, pp. 909–932, 2007.
[12] D. N. Da Hora, K. Van Doorselaer, K. Van Oost, R. Teixeira, and C. Diot, "Passive Wi-Fi link capacity estimation on commodity access points," in *Proc. Traffic Monitor. Anal. Workshop (TMA)*, 2016.
[13] L. Song, A. Mohammed, and A. Striegel, "A passive client side control packet-based WiFi traffic characterization mechanism," in *Proc. IEEE Int. Conf. Commun.*, Dublin, Ireland, Jun. 2020, pp. 1–7. [Online]. Available: https://doi.org/10.1109/ICC40277.2020.9148619
[14] 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
[15] A. Patro and S. Banerjee, "COAP: A software-defined approach for home WLAN management through an open API," *SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 18, no. 3, pp. 32–40, Jan. 2015.

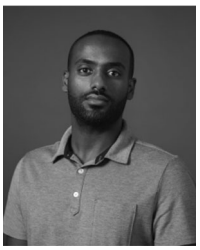
[16] Y. Bejerano, S.-J. Han, and L. E. Li, "Fairness and load balancing in wireless lans using association control," in *Proc. 10th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2004, pp. 315–329.
[17] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon, "Experimental study of router buffer sizing," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas.*, New York, NY, USA, 2008, pp. 197–210.
[18] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2014, pp. 187–198.
[19] Erikarn. (2013). *Ath9k Documentation*. [Online]. Available: https://github.com/erikarn/ath9k-docs/blob/master/ath9k-xmit.txt
[20] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley, "Facilitating access point selection in IEEE 802.11 wireless networks," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, Berkeley, CA, USA, 2005, p. 26.
[21] X. Chen and D. Qiao, "HaND: Fast handoff with null dwell time for IEEE 802.11 networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[22] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is there WiFi yet? How aggressive probe requests deteriorate energy and throughput," New York, NY, USA, 2015, pp. 317–323.
[23] S. Rayanchu, A. Patro, and S. Banerjee, "Airshark: Detecting non-WiFi RF devices using commodity WiFi hardware," in *Proc. 2011 ACM SIGCOMM Conf. Internet Meas. Conf.*, New York, NY, USA, 2011, pp. 137–154.
[24] T. Zhang, A. Patro, N. Leng, and S. Banerjee, "A wireless spectrum analyzer in your pocket," in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl.*, New York, NY, USA, 2015, pp. 69–74.
[25] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity WiFi," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2015, pp. 53–64.
[26] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.
[27] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app traffic capture and ground-truth creation," in *Proc. 4th Int. Conf. Comput. Commun. Security (ICCCS)*, 2019, pp. 1–8.
[28] 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. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, New York, NY, USA, 2014, pp. 189–200.
[29] V. Radu, L. Kriara, and M. K. Marina, "Pazl: A mobile crowdsensing based indoor WiFi monitoring system," in *Proc. 9th Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2013, pp. 75–83.
[30] S. Liu and A. D. Striegel, "Exploring the potential in practice for opportunistic networks amongst smart mobile devices," in *Proc. MobiCom*, New York, NY, USA, 2013, pp. 315–326.
[31] A. Chhetri and R. Zheng, "WiserAnalyzer: A passive monitoring framework for WLANs," in *Proc. 5th Int. Conf. Mobile Ad-hoc Sens. Netw.*, Dec. 2009, pp. 495–502.
[32] Y. Sheng *et al.*, "Map: A scalable monitoring system for dependable 802.11 wireless networks," *IEEE Wireless Commun.*, vol. 15, no. 5, pp. 10–18, Oct. 2008.

Lixing Song (Member, IEEE) received the B.S. degree from Wuhan University, Wuhan, China, in 2011, the M.S. degree from Ball State University, Muncie, IN, USA, in 2013, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA, in 2018.

He is currently an Assistant Professor with the Department of Computer Science and Software Engineering, Rose–Hulman Institute of Technology, Terre Hauet, IN, USA. His research interests lie in the area of wireless networking and mobile computing.

Aaron Striegel (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Iowa State University, Ames, IA, USA, in 2002, under the direction of Dr. G. Manimaran.

He is currently a Professor of Computer Science and Engineering with the University of Notre Dame, Notre Dame, IN, USA, where he also serves on the Executive Committee of the Wireless Institute and serves as the Bachelor of Arts in Computer Science Program Director. His research interests focus on instrumenting the wireless networked ecosystem to gain insight with respect to user behavior and optimizing network performance. Flagship projects of him include the NetSense, NetHealth, and Tesserae involving the instrumentation and analysis of data from hundreds of smartphones and wearables over a nearly seven-year period of continuous data streaming. His research interests include heterogeneous network optimization (cellular and WiFi), content distribution via edge device prestaging, and network security dynamics.



Alamin Mohammed (Graduate Student Member, IEEE) received the B.S. degree (*cum laude*) in computer engineering from Bahir Dar University, Bahir Dar, Ethiopia, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA.

His research interests include security and privacy and wireless network characterization.