FAST MOBILE NETWORK CHARACTERIZATION: DESIGN, IMPLEMENTATION AND EVALUATION

A Dissertation

Submitted to the Graduate School of the University of Notre Dame in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Lixing Song

Aaron Striegel, Director

Graduate Program in Computer Science and Engineering Notre Dame, Indiana

July 2018
The Internet today is actively embracing the evolution of mobilization. In 2017, mobile devices consumed 68% of overall Internet traffic. This number is expected to grow sevenfold in the next five years. Due to the limited resources on wireless spectrum, the relentlessly growing mobile data demand can portend an ominous future for the Quality of Experience (QoE) on mobile networks (e.g., WiFi and cellular). In addition, with the dynamics of the wireless channels, the performance on mobile networks tends to vary significantly often making the wireless link as the bottleneck to decide QoE. Network characterization is a tool to understand the performance of mobile networks. Unfortunately, existing solutions are either cumbersome or inaccurate. Some are effective but the cost is prohibitively expensive (e.g., cost tens of megabyte data and take the order of tens of seconds to finish). Other methods are lightweight but yield low accuracy. Critically, the different transmission schemes under modern mobile networks make conventional characterization methods fail due to the issues with the lower-layer behaviors. Therefore, the key challenge is how to conduct mobile network characterization in an accurate and efficient manner.

In this dissertation, I propose a test suite of mobile network characterization using both active and passive approaches on WiFi and cellular networks. By carefully studying the data transmission behaviors at the lower layers (e.g., the physical layer),
my approach manipulates the observations of traffic patterns on the upper layers (e.g., the transport layer) to enable accurate and efficient characterization methods. Specifically, in this dissertation, I design two active available bandwidth estimation tools focusing on WiFi and LTE networks. By leveraging the aggregation/batching properties on WiFi and LTE, I design a probe packet train that utilizes the concept of self-induced congestion. I implement the estimation tools in an HTTP-based platform—\textit{FMNC} (Fast Mobile Network Characterization). This system is designed to transmit packet sequence in a sliced, structured, and reordered manner. This feature enables mobile users to run the designed tests without installing a dedicated application on the client side. In addition to the active approaches, I also propose an efficient passive client-side traffic characterization method on WiFi networks. By exploiting the frame aggregation feature on WiFi, the method can piggyback on a WiFi scan operation and achieve accurate traffic characterization with minimal traffic capture (i.e., control traffic only). I conduct two case studies using the passive characterization method: a real-world measurement case and an application case in video streaming. The measurement study reveals interesting observations under different network scenarios. The application case helps improve stall rate of video streaming under WiFi. Overall, all the proposed methods/applications in this dissertation have been carefully evaluated through extensive in-lab and real-world experiments.
To my beloved wife Panpan Cao,

whose unconditional love and support made this work possible.
<table>
<thead>
<tr>
<th>Chapter 4: FMNC on the WiFi Network</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Problem Demonstration</td>
<td>41</td>
</tr>
<tr>
<td>4.1.1 Frame Aggregation Overview</td>
<td>41</td>
</tr>
<tr>
<td>4.1.2 AB Estimation under the Impact of Frame Aggregation</td>
<td>42</td>
</tr>
<tr>
<td>4.2 Exploiting Frame Aggregation for Link Characterization</td>
<td>43</td>
</tr>
<tr>
<td>4.2.1 Frame Aggregation Characterization</td>
<td>44</td>
</tr>
<tr>
<td>4.2.2 Manipulating Frame Aggregation to Detect Link Congestion</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Probe Packet Train Design</td>
<td>52</td>
</tr>
<tr>
<td>4.3.1 Probe Packet Train Format</td>
<td>53</td>
</tr>
<tr>
<td>4.3.2 Parameter Setting</td>
<td>54</td>
</tr>
<tr>
<td>4.3.3 Estimation Algorithm</td>
<td>55</td>
</tr>
<tr>
<td>4.4 Experimental Evaluation</td>
<td>56</td>
</tr>
<tr>
<td>4.4.1 Cross Traffic</td>
<td>57</td>
</tr>
<tr>
<td>4.4.2 Interference</td>
<td>63</td>
</tr>
<tr>
<td>4.4.3 Rate Limiting</td>
<td>65</td>
</tr>
<tr>
<td>4.5 Real-World Deployment</td>
<td>65</td>
</tr>
<tr>
<td>4.5.1 AB/AT Analysis</td>
<td>67</td>
</tr>
<tr>
<td>4.6 Conclusion</td>
<td>69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5: FMNC on the LTE Network</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Background and Motivation</td>
<td>72</td>
</tr>
<tr>
<td>5.1.1 Challenges of Estimating Available Bandwidth on LTE</td>
<td>72</td>
</tr>
<tr>
<td>5.2 System Design</td>
<td>75</td>
</tr>
<tr>
<td>5.2.1 TBS Fluctuation as the Network Congestion Indicator</td>
<td>76</td>
</tr>
<tr>
<td>5.2.2 Probe Train Design</td>
<td>79</td>
</tr>
<tr>
<td>5.2.3 FMNC-Based Esitimation</td>
<td>83</td>
</tr>
<tr>
<td>5.3 A Bayesian Change Point-Based Estimation Algorithm</td>
<td>84</td>
</tr>
<tr>
<td>5.3.1 Data Preprocessing</td>
<td>85</td>
</tr>
<tr>
<td>5.3.2 Model Design</td>
<td>85</td>
</tr>
<tr>
<td>5.3.3 Estimation ( \hat{\tau} ) via Gibbs Sampling</td>
<td>86</td>
</tr>
<tr>
<td>5.3.4 Interpreting Estimation Result</td>
<td>88</td>
</tr>
<tr>
<td>5.4 Performance Evaluation</td>
<td>89</td>
</tr>
<tr>
<td>5.4.1 Effectiveness Validation</td>
<td>89</td>
</tr>
<tr>
<td>5.4.2 Parameter Setting</td>
<td>92</td>
</tr>
<tr>
<td>5.4.3 Longitudinal Evaluation</td>
<td>94</td>
</tr>
<tr>
<td>5.4.4 In-Town Drive Evaluation</td>
<td>98</td>
</tr>
<tr>
<td>5.5 Conclusion</td>
<td>101</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6: Passive WiFi Characterization via WiFi Scan</th>
<th>103</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Preliminary Feasibility Study</td>
<td>103</td>
</tr>
<tr>
<td>6.1.1 Why Control Packets?</td>
<td>104</td>
</tr>
</tbody>
</table>
6.1.2 WiFi Scan for Characterization ................................................. 106
6.2 WiFi Characterization via Control Packets ........................................ 107
  6.2.1 Primitive Measurements .................................................. 107
  6.2.2 Deriving the Characterization Metrics ................................. 114
6.3 WiFi Scan for Characterization .................................................... 126
  6.3.1 Feasibility Study ............................................................. 127
6.4 Performance Evaluation ............................................................. 129
  6.4.1 Setting ............................................................................. 130
  6.4.2 Evaluation with Ground Truth ........................................... 131
  6.4.3 Empirical Study on Other Metrics ....................................... 134
6.5 Conclusion ................................................................................. 135

Chapter 7: Case Studies of WiFi Passive Characterization .................... 136
  7.1 Empirical Case Study in the Wild ........................................... 136
    7.1.1 Result and Analysis ...................................................... 138
  7.2 Application: A Cross-Layer Stall-Free WiFi Video Streaming Mechanism 140
    7.2.1 Problem Demonstration .................................................. 140
    7.2.2 SEWS: A Stall-Free WiFi Streaming Adaptation Algorithm ...... 143
    7.2.3 Performance Evaluation .................................................. 145

Chapter 8: Future Work .................................................................. 152
  8.1 Research Problem .................................................................. 152
  8.2 Industrial Application ............................................................... 153

Bibliography .................................................................................. 155
FIGURES

1.1 Illustration of AB, AT and Elasticity ............................................... 7
2.1 Packet Gap Model illustration .......................................................... 20
2.2 Packet Rate Model illustration .......................................................... 21
3.1 FMNC work flow. ........................................................................... 32
3.2 FMNC client side screenshot. .............................................................. 33
3.3 FMNC server major components. ......................................................... 34
3.4 Sliced, structured, and reordered packets. ........................................... 35
3.5 FMNC server code structure. ............................................................... 37
3.6 A snapshot of FMNC packet log record. .............................................. 38
4.1 Illustration of frame format via A-MPDU. .......................................... 41
4.2 The observed receiving packet gap and packet rate in a packet sequence under 802.11g v.s. 802.11n. The dashed line indicates the probe rate. 43
4.3 An example of scheduling under frame aggregation. .......................... 46
4.4 The curve of the probe packet queuing delay v.s. cross traffic load. ...... 47
4.5 Aggregation Intensity v.s. probe packet gap ($G_{snd}$). ....................... 48
4.6 Aggregation Intensity v.s. cross traffic load w.r.t. probe packet gap ($G_{snd}$). ................................................................. 49
4.7 $AI - AI_{base}$ v.s. cross traffic load w.r.t. the probe rate. The link capacity is 80Mb/s. ............................................................... 51
4.8 eCDF of $AI - AI_{base}$ under un/congested link w.r.t. the probe rate. ... 52
4.9 Probe packet train format with parameters labeled. ............................ 53
4.10 802.11g (2.4GHz) without frame aggregation. ................................. 57
4.11 802.11n (2.4GHz) MIMO with frame aggregation. ............................ 58
4.12 802.11ac (5GHz) MIMO with frame aggregation. ............................. 59
4.13 Queuing delay observed by PathChirp without (a) and with (b) frame aggregation. ........................................................................... 60
4.14 AB classification result w.r.t. flow size of TCP cross traffic. ............. 61
4.15 Experiment result from varying cross traffic versus interference traffic.  
4.16 $AI - AI_{\text{base}}$ of each sub-train w.r.t. rate limitation and link utilization.  
4.17 eCDF of AB resolution time by RTT.  
5.1 Channel resource allocation on LTE downlink.  
5.2 AB estimation (PathChirp) on Ethernet v.s. LTE under similar network setting.  
5.3 The PHY layer observation under different network competition conditions.  
5.4 TBS fluctuation in the absence v.s. presence of *iperf3* traffic w.r.t. different send rates.  
5.5 Step by step estimation algorithm illustration on real data. (a) and (b) are from the same test.  
5.6 (a) AB (PHY Trace) v.s. AT; (b) AB (PHY Trace) v.s. AB (Timestamp).  
5.7 Time series plot of AT (Cubic) and AB measurement.  
5.8 The estimation difference distributions under different $L$.  
5.9 The estimation difference distribution of the filtered out results with different $p_\theta$.  
5.10 Time-series plot of AT and AB estimation from the longitudinal run.  
5.11 Pearson correlation coefficient over the smoothed result.  
5.12 Absolute estimation error.  
5.13 Test time cost analysis.  
5.14 Drive test route map. Different colors denote different cells and the radius size indicates the RSRQ.  
5.15 AT and AB under different cells. RSRQ of each cell is also plotted.  
5.16 The distribution of (a) AT - AB and (b) AT under different moving speeds.  
6.1 Illustration of data transmission under frame aggregation.  
6.2 Format of Block ACK frame.  
6.3 Data transmission under frame aggregation. The sequence number (SSN) and bitmap are indicated. A 64 bitmap is used in this case.  
6.4 (a) AI estimation error; (b) The maximum AI and loss rate across different transmission rate.  
6.5 $G_{BA}$ (a) and $G_{MPDU}$ (b) v.s. AI given certain transmission rate (i.e., $144.4$ Mbps).  
6.6 CDF of MPDU Gap (when $AI > 1$) w.r.t transmission rate.  
6.7 Estimated airtime v.s. flow size w.r.t. window size $\omega$.  

vii
6.8 Estimation accuracy v.s. window size $\omega$ w.r.t. flow size. .......... 117
6.9 CDF of $G_{MPDU}$ w.r.t. flow size. ................................. 118
6.10 CDF of AI on data or Ack stream w.r.t. flow size. ................. 119
6.11 The impact of window size $\omega$ on the two components of estimating transmission rate. ........................................... 121
6.12 Transmission rate estimation accuracy from different $\theta$ (a) and different flows (b). ......................................................... 122
6.13 (a) Queuing Indicator performance compared with (b) beacon delay. 110 124
6.14 Queuing Indicator v.s. frequency of relative estimation accuracy of transmission rate. ....................................................... 125
6.15 eCDF of (a) scan interval and (b) dwell time observed in real world dataset. ................................................................. 128
6.16 Time series of (a) airtime estimated compared with ground truth and (b) number of packets captured during dwell time for each scan. 132
6.17 Correlation between estimated value and ground truth w.r.t. number of devices used for generating result. All the results are from 5GHz band. .............................................................. 132
6.18 The empirical observation on other characterization metrics. .......... 133
7.1 Empirical observation of various characterization metrics from university campus and city downtown. ................................. 139
7.2 Time series plot of the latest throughput-based approach. The three vertical lines are the starts of the first and second competing client, and the end of all competing traffic respectively. The shadow box at (b) indicates video stalling. ........................................ 142
7.3 Implementation software work flow. .................................... 145
7.4 Time series plot by using the proposed method (in contrast with Fig 7.2). The experiment setting is exactly same as in Fig 7.2. .... 146
7.5 Performance comparison under heavy-flow competing traffic. ....... 148
7.6 Performance comparison under on-off light/medium-flow competing traffic. ................................................................. 150
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Proposed solutions summary.</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Proactive bandwidth estimation related works summary.</td>
<td>17</td>
</tr>
<tr>
<td>4.1</td>
<td>Probe packet train design parameters description.</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Cost comparison across different methods.</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>Data summary.</td>
<td>70</td>
</tr>
<tr>
<td>4.4</td>
<td>Result analysis based on $AT$ classification.</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>Packet train format parameters</td>
<td>82</td>
</tr>
<tr>
<td>6.1</td>
<td>Data statistics summary at the tent.</td>
<td>131</td>
</tr>
<tr>
<td>7.1</td>
<td>Data summary of the real-world study.</td>
<td>137</td>
</tr>
<tr>
<td>7.2</td>
<td>Stall rate for different algorithms.</td>
<td>141</td>
</tr>
</tbody>
</table>
Soli Deo Gloria! To the glory of God alone! Praise Lord for HIS mercies that we are not consumed. Thank God for giving me the passion and enough wisdom to accomplish this work.

I would like to express my sincere gratitude to my advisor, Prof. Aaron Striegel, for his invaluable guidance, inspiration, and suggestions in my life. Beyond the instructions in my research study, he also sets a good example for me as being a loving husband and patient father. I would also like to thank my committee members, Prof. Christian Poellabauer, Prof. Dong Wang and Prof. Richard Billo, for their generous attention.

In addition, I want to thank my dear colleagues for being supportive as research consultants and friends: Xueheng Hu, Rachael Purta, and Gonzalo Martinez. Last but not least, I would like to express my gratitude to our wonderful Administrative Assistant, Joyce Yeats, who is always patient to help me regarding many miscellaneous issues.
CHAPTER 1
INTRODUCTION

Since the debut of commercial Internet service in the late 1980s, the advent of the Internet brought on a “new industrial revolution”. Over the past two decades, with advances of wireless network technologies, the Internet has actively embraced an era of mobilization. Mobile networks (e.g., WiFi, Cellular and etc) have enabled a cordless experience of the Internet and have quickly became the preferred way for accessing the Internet. In the dawning of the next generation of wireless communication – 5G, mobile networks are settled for explosive growth of mobile devices and a relentlessly booming data load. Given the limited spectral resources for wireless channels, the Quality of Experience (QoE) on mobile networks will be quite vulnerable. Therefore, understanding mobile network performance becomes a critical problem. Understanding mobile network performance in an efficient and accurate way is the problem this dissertation aims to solve.

The remainder of this chapter is organized as follows. Section 1.1 presents a brief review of the history of mobile networks, e.g., cellular and WiFi. Section 1.2 addresses the primary problems the modern mobile networks are facing today. Next, Section 1.3 discusses the network characterization methodology, including different approaches (e.g., proactive versus passive) and different metrics (e.g., achievable throughput versus available bandwidth). Section 1.4 explores the different challenges under the modern mobile networks and then presents the designed solutions and contributions of the works in this dissertation. Finally, Section 1.5 gives a high-level overview of the content of this dissertation.
1.1 Evolution of the Mobile Network

The history of mobile networks can be viewed from the development of two types of networks: cellular networks and WLAN (Wireless Local Area Network). Harkening back to 1940s, the cellular phone system was emerged as the first deployment of mobile networks. However, it was not until the 1990s when 2G (Global System for Mobile communications) were introduced mobile networks started to be able to deliver Internet data traffic [3]. Due to bandwidth constraints and the prohibitively expensive cost, data service via cellular network was largely limited. Since late 1990s, cellular networks have quickly embraced a data-focused adaptation. From 3G to 4G LTE (Long-Term Evolution), the cellular network is capable of providing an unabridged flavor of Internet experience.

The researches of WLAN started back circa 1970. Due to the expensive hardware cost at the early years, the various industry solutions and proprietary protocols were replaced by standards (e.g., IEEE 802.11) at the end of 1990s [5]. The first version of WiFi was released in 1997 and clarified in 1999. From the very first version of WiFi (IEEE 802.11 legacy) which supports merely 2 Mb/s data rate, the modern WiFi (IEEE 802.11ac) can provide exceptional transmission speed up to 1,300 Mb/s and has become the default way for people to access the Internet.

As of today (2018), with the advent of the smartphone and numerous mobile devices, wireless data usage has exploded. Mobile network traffic has started to dominate the Internet consumption. According to Cisco [26], WiFi and mobile-connected devices generated 68% of all Internet traffic by 2017. Particularly, 71% of all mobile communication flows over WiFi. Furthermore, the rise of the Internet of Things (IoT) will only accelerate the development of mobile networks. Globally, mobile data traffic will increase sevenfold between 2016 and 2021. Mobile networks have penetrated to every aspect of people’s daily life and become an indispensable part of the modern society.
To address these newfound demands, the next generation of mobile networks (5G) aims to bring the radical performance improvements to achieve 1 ms latency, 10 Gbps throughput, and 100 billion connection support. In order to satisfy these requirements, existing mobile networks will need to be largely revamped and various new communication systems will be incorporated. For example, the newly-evolved LTE-U (LTE on the Unlicensed band) is looking for manipulating the WiFi spectrum to shoulder the traffic load on cellular networks. In addition, millimetre-wave and other novel communication systems (e.g., visible light communication) will be utilized to provide extra bandwidth for data transmission. As particularly focusing on IoT, network operators are eagerly providing support for low-cost communication (e.g., NB-IoT). Overall, today’s mobile network is continually improving to meet the ever-growing bandwidth requirement from mobile users.

1.2 Problem Motivation

The prosperity of wireless technologies brings tremendous challenges to existing mobile networks. By 2021, there will be 1.5 mobile devices per capita. The average smartphone will generate 6.8 GB of traffic per month by then, a fourfold increase over the 2016 average of 1.6 GB per month. Most importantly, the massive amount of devices must share limited spectral resources. Namely, the mobile devices running on the same band can interfere with each other and compete for the channel resources. Furthermore, varying roles and user cases for wireless devices will become all the more stark and variant. To address those demands, It is fundamental need to better understand the mobile network performance, which forms the motivation of this dissertation. Generally, the primary motivations of understanding the mobile network performance can be summarized as 1) improved network management and 2) enhanced QoE of mobile users.

- **Improved Network Management:** Meticulous network management usually re-
lies on accurate performance feedback. Network characterization plays an indis-
pensable role to provide instant network activity information. Unfortunately,
such information in modern mobile networks is not readily available. On unli-
censed spectrum, e.g., WiFi, the lack of a central controller makes data trans-
mision behave non-deterministically. Understanding the network performance
is paramount important for channel resource allocation. As an example of user
association, if a WiFi client can be imparted about the performance on different
alternative WiFi access points (AP), the client then can judiciously associate
with the optimal AP with the best performance.

- *Enhanced QoE:* The high density of devices as well as data demands portend
a serious performamnce challenges for mobile networks. The serious competi-
tion among devices can potentially suppress throughput and create significant
bandwidth variations, thus resulting in mundane QoE. As many of today’s In-
ternet services are sensitive to network conditions, the performance of mobile
networks plays a crucial role in deciding QoE. For example, as one of the most
popular content service, mobile video accounted for 60 percent of total mobile
data traffic in 2016 [26]. Most mobile video streaming usually employs the
DASH (Dynamic Adaptive Streaming over HTTP) protocol [112], which can
automatically adjust the video quality based on the current network condition.
Therefore, understanding network performance is critical to deliver the best
video streaming experience.

1.3 Network Characterization

Technically, the problem of understanding network performance can be referred
to *network characterization.* In order to conduct characterization, there are primarily
two approaches based on the whether the characterization requires involving extra
traffic or not: 1) proactive [16, 18, 19, 30, 31, 49, 50, 53, 54, 66, 73, 90, 92, 105,
106, 109] and 2) passive [13, 81, 89, 107, 117, 121]. The proactive approach needs
to inject test traffic to examine a certain network property. For example, the well-
known tool ping [113] is a simple proactive method to compute the end-to-end path
delay. In contrast, the passive approach requires no traffic injection that only needs
to observe/analyze the ongoing traffic in a stealth manner. As an example of passive
characterization methods, SNMP [114] is a network monitoring tool/protocol that
collects and organizes network information in a passive way. This dissertation focuses
on the network characterization for mobile networks with both proactive and passive manner. In the next part, the two approaches and the corresponding methodologies will be further discussed.

1.3.1 Proactive Bandwidth Estimation

Proactive network estimation can help measure a variety of network properties, including latency/delay, loss, bandwidth, etc. These metrics can be measured at different scales, e.g., link, path, region and so on. Some of the metrics are straightforward and relatively trivial to obtain, e.g., delay and loss. These metrics are readily available and can be directly observed from traffic transmission. While other metrics (e.g., bandwidth), which are used to reveal some latent properties of the networks, require delicate estimation mechanism to capture. As the focus of the proactive characterization, this dissertation particularly aims at estimating the end-to-end bandwidth.

End-to-end bandwidth characterization targets measuring the bandwidth condition from a client to a server which is a directional metric \[19\]. The bandwidth condition can be characterized from different aspects, including bandwidth capacity \[16, 18, 19, 30, 31, 50, 53, 54, 109\], achievable throughput (AT) \[2, 8, 41, 67, 71, 72, 74, 76, 78\], available bandwidth (AB) \[49, 66, 73, 90, 92, 105, 106\], etc. Bandwidth capacity implied by its name simply refers to the physical capacity of a link regardless of the instant traffic load. Conversely, the achievable throughput and the available bandwidth are temporary estimations that describe a temporary condition of the bandwidth. The two metrics are similar in that both of them attempt to measure a “usable” portion of the bandwidth. However, the definitions of the term “usable” differ significantly between the two metrics which leads to intrinsically different measurement philosophies. The following subsections further discuss the two metrics.
Achievable Throughput

One of the most common mechanisms for assessing link bandwidth is the Available Throughput (AT) test popularized by sites such as Speedtest.net [76], iperf [2], and others (Mobilyzer TCP throughput test [74]). Technically, an AT test measures the actual throughput by running a TCP flow over a pre-defined period of time (ex. 10 seconds) and then observing the steady-state throughput achieved by said flow. The estimation of AT usually requires congesting the network which can aggressively consume the network resource and even crowd out the existing traffic. In terms of the test cost, AT estimation can be expensive regarding both time duration as well as data consumption.

Available Bandwidth

In contrast, Available Bandwidth describes the spare or residual capacity of the a during a certain period of time. For an end-to-end path, AB refers to the available bandwidth of the narrow link which has the minimum available bandwidth. Given a $L$-hop end-to-end path, assume $C_i$ is link capacity of $i$-th link, and $\overline{u}_i(t - \tau, t)$ is the average utilization of the link from time $t - \tau$ to $t$. Therefore, we define the instantaneous available bandwidth $AB^t$ at time $t$ for the path as:

$$AB^t = \min_{i=1,\ldots,L} C_i(1 - \overline{u}_i(t - \tau, t))$$  \hspace{1cm} (1.1)

AB estimation techniques can be lightweight as it requires no flooding traffic. Essentially, AB estimation methods leverage the queuing observation/inference on the network path to decide bandwidth condition. Unfortunately, for mobile networks (i.e., WiFi and cellular), a key challenge for estimating AB (as will be demonstrated shortly) is that the aggregated transmission mechanisms impose a deleterious effect

\footnote{It should be noted that, narrow link is talking to available bandwidth, which is different from a tight link where minimal capacity occurs.}
on existing AB techniques. Critically though, before unfolding the challenges posed for AB, it is important to distinguish what AB captures versus what AB does not capture, particularly as it relates to AT.

**Elasticity**

Effectively, AB represents the minimum throughput that a TCP flow could achieve at that point in time while AT represents what it would have actually achieved at that point in time. We call the difference between the two terms as the *elasticity*. The elasticity is a function of the existing cross traffic (flow number, relative RTT, relative link qualities) and link capacity. Figure 1.1 illustrates the difference between the two measurements. Although that information would be nearly impossible for a mobile client to know, the concept of elasticity can be helpful to illustrate the difference between *AB* and *AT* by categorizing the key zones or cases where such differences occur:

- **AB = AT**: *AB* will equal *AT* when either the link is entirely open (no existing traffic) or all existing traffic on the link is UDP-based (not TCP friendly). A newly formed TCP flow would not be able to easily crowd out the existing traffic and thus residual capacity represents actual capacity. In practice, such a case would be quite rare.

- **AB < AT**: The most common case will be where *AB* represents the obtainable
bandwidth without suppressing existing traffic. While AT measures the maximum achievable by further pushing back existing traffic with a TCP flow(s). That additional bandwidth gain of AT above AB is the *elasticity*.

- $AB > AT$: For cases where rate-limiting may be in place, the lightweight nature of an $AB$ test may not trip the rate-limiting features of a link. Hence, $AB$ may actually exceed $AT$.

### 1.3.2 Passive Traffic Characterization

As another approach to characterize network performance, passive monitoring is a non-intrusive method compared to the proactive method discussed in the previous subsection. Due to the nature of wireless communication, the transmission media (i.e., spectrum) of wireless networks is shared by the mobile users. This property enables the traffic to be captured by nearby devices on the same spectrum. However, since the traffic on the licensed spectrum (i.e., cellular) is usually encrypted, this method is more suitable on unlicensed spectrum such as WiFi. Therefore, the passive traffic characterization in this dissertation explicitly applies only to WiFi.

The key advantage of passive network characterization over proactive approach is its non-intrusiveness. Without actively injecting any traffic, the passive characterization behaves in a surreptitious way to monitor and collect network statistics over time. In constrast to the proactive methods which often aim to capture one aspect of the network condition (e.g., iperf3 [2] is primarily used to estimate the network throughput), a passive characterization system can provide a more comprehensive characterization over many network properties. Therefore, this method is usually employed for network management [114], longitudinal network monitoring, etc. For example, the *SNMP* system installed on Aruba WiFi access point offers hundreds of network attributions, e.g., number of associated clients, traffic load per client, packet count of different traffic type and etc.

For implementation, passive network characterization can take place in either a centrally-controlled manner (i.e., server side) or an ad hoc way (i.e., client side).
The central-controlled method usually runs on some management infrastructure that can provide a global view of the network traffic. However, the deployment cost of the centrally-controlled network characterization is usually expensive that requires extra software and hardware support. Under the environment of WiFi networks, the centrally-controlled network characterization usually occurs on WiFi APs, where installation of a dedicated system is required and the storage and re-directing of the captured information need to be well-maintained. In contrast, ad hoc passive characterization exploits each network client as a “sensor” to capture a local view of the network. This approach can be quite flexible and cost-efficient to deploy. The downside is that the observation captured from a client may not be complete. In WiFi networks, each WiFi client is potentially capable of picking up nearby traffic on the same channel. However, any transmissions that occur outside of its vicinity is unknown to this particular client. Furthermore, as directional transmission (i.e., beamforming) and other technical issues that can result in capture loss, the client side passive characterization faces several challenges to achieve a high characterization accuracy. Depending on the purposes of network characterization, the two methods have their own advantages and disadvantages. This dissertation is mostly interested in a passive client side approach by utilizing its flexibility and low-cost implementation while looking for overcoming the accuracy issues in WiFi network environment.

1.4 Challenges and Proposed Solutions

This dissertation will explore network characterization on WiFi and cellular networks from both proactive and passive perspectives. As the summary shows in Table 1.1, three solutions are proposed, including 1) Fast Mobile Network Characterization on WiFi (FMNC-WiFi), 2) Fast Mobile Network Characterization on cellular (FMNC-Cellular) and 3) Passive WiFi Characterization via Scan (CharScan).
TABLE 1.1

PROPOSED SOLUTIONS SUMMARY.

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Network Type</th>
<th>Approach</th>
<th>Aggregation Scheme</th>
<th>FMNC Based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WiFi</td>
<td>Cellular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMNC-WiFi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FMNC-Cellular</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CharScan</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Specifically, FMNC-WiFi focuses on estimating the available bandwidth on WiFi networks in a fast and efficient manner. Similarly, FMNC-cellular is an extension of FMNC-WiFi to achieve the same goal on cellular networks. Both of these methods are built upon an implementation system called FMNC which is designed to slice and permute packets for proactively probing. CharScan, on the other hand, is a passive characterization solution on WiFi network that aims to provide accurate network statistics with minimum cost.

Overall, the whole tuition behind the three solutions is to leverage the lower layer data transmission properties on modern mobile networks to infer traffic condition. Particularly, this dissertation manipulates the batched/aggregated transmission behavior on mobile networks as an enabler to achieve accurate and efficient network characterization. In order to boost throughput, modern mobile networks (cellular and WiFi) exploit different mechanisms to attempt to aggregate multiple packets to send together, such as frame aggregation on WiFi and transport block on LTE networks. Moreover, the specific policies of how to aggregate are determined by the current network condition. Thus the observation of aggregation on upper layers can innately embody rich information regarding network performance. This intuition is
the core idea that motivates the works in this dissertation. The next subsections will discuss the detailed challenges and contributions of each of the proposed works.

1.4.1 FMNC-WiFi

Today, nearly all mobile devices try in some form to aggressively push users to WiFi in order to deliver what is predicted to be a better user Quality of Experience (QoE). However, while WiFi can offer improved performance, there are a growing number of instances where the WiFi may not be the optimal choice. A necessary metric as to whether WiFi will be positive or negative for the user QoE is an accurate characterization of the available bandwidth on WiFi link. Unfortunately, the very timing patterns that allow AB tests to operate more efficiently tend to struggle with the increased noise in wireless and most importantly, are outright broken under modern WiFi variants (802.11n, 802.11ac) that utilize Frame Aggregation (FA). Under the frame aggregation, the assumption of FIFO per-packet based scheduling does not hold. The aggregated transmissions behave like a batch scheduler which dramatically distorts the timing characteristics of received packets.

This work explores the intrinsic properties of frame aggregation for the detection of congestion and creates a proof of concept system, AIWC (Aggregation Intensity based Wifi Characterization). The evaluation experiments show that AIWC possesses robustness across varying bandwidth levels, interference, and traffic patterns. The contributions of the work are as follows:

- **Frame Aggregation as the key enabler:** I show how frame aggregation embodies a rich set of WiFi link characteristics with the queuing effects at the specific AP being of particular interest.

- **Aggregation Intensity based solution:** I demonstrate how one can induce frame aggregation to measure link congestion and in turn to measure the available bandwidth. I introduce the concept of Aggregation Intensity (AI) to capture frame depth which can then be manipulated with targeted packet sequences to capture the likely AB.
• **Implementation:** I construct a proof of concept system AIWC that realizes the proposed approach through a libpcap-based server with customized TCP flows. Our system operates in-band within TCP to avoid modifications to the end client by leveraging components from TCP Sting [96] and RIPPS [65].

• **Experimental evaluation:** I demonstrate the robustness of AIWC across varying bandwidth levels, interference, and traffic patterns. I show significant performance improvements versus the existing body of AB literature [34, 91, 106].

1.4.2 FMNC-Cellular

The work aims to extend the proposed available bandwidth technology on WiFi to the modern cellular network, i.e., LTE. Similar to WiFi, LTE also adopts packet aggregation/batch scheme akin to the frame aggregation on WiFi. However, the aggregation scheduling policies are intrinsically different on WiFi versus LTE. Therefore, the aggregation intensity design proposed in FMNC-WiFi is not valid in the LTE network environment. A different approach is needed to conquer the problem of available bandwidth estimation on LTE network.

In this work, I propose a TBS fluctuation-based available bandwidth estimation tool for LTE network. Nominally, batch transmission behavior of LTE troubles traditional AB methods by distorting packet dispersion pattern at the receiver. Hence, the designed estimation method leverages the TBS (Transport Block Size) behavior of the PHY layer to accurately identify potential bandwidth competition on the LTE network. The main contributions of this work can be summarized as follows:

• **TBS as an effective congestion indicator:** The traditional AB estimation methods presume a non-batch per-packet transmission scheme, which cannot hold on LTE network. The bursty and spiky packet arrival behavior severely hampers the existing AB estimation approaches. To overcome this issue, I carefully investigate the data transmission behavior on LTE and find that the consistency of TBS can serve as an effective indicator to reflect the degree of traffic competition occurring at eNodeB when it attempts to schedule bandwidth resource to multiple mobile users.

• **Bayesian change point estimation algorithm:** To handle the noisy measurements, I exploit a probabilistic model to translate the problem of estimating AB
into a change point detection upon the observed measurements. The Bayesian approach carefully considers the uncertainties involved in the estimating process. By utilizing *Gibbs sampling*, I solve the problem and further manipulate the sampled posterior distributions to assess the estimated result.

- **Real world experimental validation/evaluation:** Throughout the work, I ran real-world experiments to validate the effectiveness of the proposed technical designs. In addition, I conduct longitudinal experiment run to monitor the bandwidth condition. As a result, I show that our AB estimation tool can accurately capture the bandwidth variations on LTE which yields over 0.75 *Pearson* Correlation Coefficient with the parallel-running throughput test over a day-long run.

1.4.3 CharScan

As the previous two works emphasize on the bandwidth estimation by using a proactive approach, in this work, I will explore the WiFi traffic characterization in a passive way. The idea is to view the mobile client itself as a capable sensor [27, 63, 64, 119]. The client-side method acts exclusively as a “sniffer” on the WiFi network but 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. However, one critical technical obstacle is that, the actual *sniffing* (i.e., packet capture) capabilities of most mobile devices tend to be fare poorly. Packet capture is often inaccessible absent significant modifications by the device owner and as will be discussed later, often suffer severe losses when monitoring data packets.

In order to overcome the packet capture loss issues, I design a lightweight characterization solution which merely requires capturing control packets. This work proposes a new technique that builds on the properties of frame aggregation, specifically the Block Acknowledgement, and show how Block Acknowledgements 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, I show that the stable observation time for Block Acknowledgements can be sufficiently satisfied during a normal WiFi scan period (20 ms). It means that I
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 work are:

- **Sensing with control packets:** I show how observing control packets, especially Block Acknowledgement (BA), can be used to extract a rich suite of WiFi characterization. I 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. I demonstrate this mapping through extensive empirical studies across a wide variety of scenarios to demonstrate accuracy, efficiency, and robustness.

- **Demonstrate a need for only limited control packets:** I show that only an extremely limited window of control packets are necessary to extract a stable view of the WiFi link characterization (20 ms). I introduce several key concepts and assumptions necessary to work within such small time window as well as present extensive experiments.

- **Extensive exploration of WiFi scanning behavior:** I study the feasibility of adapting the scan for characterization. By analyzing extensive pools of WiFi scanning data (41k devices), I investigate the typical scan behavior in the wild. I 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 minutes.

- **Demonstrate robustness on real-world data:** I validate our approach and its accuracy by conducting experiments on a real-world dataset captured during a tailgate involving multiple 802.11ac access points. I conduct trace-driven experiments and show that with only a handful of devices (i.e., 10), the designed characterization can achieve a high correlation efficiency (0.8) with the observed ground truth for airtime and throughput estimation.

- **A video streaming application case:** As an application study case, I coupled the designed method into a WiFi video streaming solution. Since the WiFi video streaming is utterly vulnerable to channel bandwidth variation, the traffic competition occurred on WiFi channel usually leads to severe stalling on WiFi video streaming. To solve this problem, I incorporated the airtime estimation from the proposed characterization method to help WiFi client acknowledge of channel condition.
1.5 Content Overview

In the rest of the dissertation, I will give a background and overview of related works in Chapter 2. As the key implementation platform for the two proactive methods (FMNC-WiFi and FMNC-Cellular), I will go through the design principles and code structure of the FMNC in Chapter 3. Then, the detailed design and performance evaluation of the three proposed solutions will be elaborated in Chapter 4: FMNC-WiFi, Chapter 5: FMNC-Cellular, and Chapter 6: CharScan. As the case studies for the passive method, Chapter 7 will present an empirical study in the wild of using CharScan, and an application in WiFi video streaming. Finally, I will close the dissertation with future work discussions in Chapter 8.
CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter will give a detailed literature review regarding the topics studied in this dissertation. Specifically, I will explore the existing works in the general fields of active bandwidth estimation (Section 2.1) as well as passive mobile network characterization (Section 2.2). Furthermore, I break down the works of bandwidth estimation into sub-categories based on the specific metrics used. For mobile network characterization works, the review is classified into different application scenarios and purposes. At the end of the chapter, I will further outline the primary techniques and the major contributions in this dissertation in contrast to the existing works.

2.1 Proactive Bandwidth Estimation

The problem of network bandwidth estimation has been extensively explored in the past decade [36, 86]. I start with reviewing the existing works on the general area of bandwidth estimation in computer networks. Based on the metrics used, there are primarily three properties related to bandwidth: capacity, achievable throughput and available bandwidth. Table 2.1 summarizes the related works in this section based on metric used.
2.1.1 Bandwidth Metrics Overview

**Capacity**

Capacity is the maximum transmission rate one can achieve on a link. Generally, the term link capacity particularly refers to the measurement reflected on at the IP layer which is slightly less than the data rate on the data link layer (Layer 2). We assume the maximum transmission rate on L2 is $C_{L2}$ and the header on L2 is $H_{L2}$. When the payload size of the upper layer is $P^1$, then we have the upper layer capacity $C$ as:

$$C = C_{L2} \cdot \frac{P}{P + H_{L2}} \quad (2.1)$$

When applied to an end-to-end path, the capacity usually speaks to the minimum

---

*WiFi, +Cellular

---

The packet size here normally is the maximum transmission unit (MTU).
capacity along the path, such as:

\[ C = \min C_i, i = 1, \ldots, H \]  \hspace{1cm} (2.2)

Where \( C_i \) is the \( i \)-th hop on the path. The link with the minimum capacity is called the **tight link**.

*Cprobe* [19] took the first attempt to use the fixed-interval packet pairs to infer per-hop capacity by observing the packet dispersion. The validation experiments revealed many practical issues that inspired many later works, such as [16, 31]. An alternative approach of packet dispersion is to use RTT measurement. *Pathchar* [109] sent a series of probes by varying values of TTL (time-to-live) and packet sizes. *Pathchar* then used the RTT to infer the latency and bandwidth of each link in the path, the distribution of queue times, and the probability that a packet was dropped. Similarly, *clink* [31] estimated the latency and bandwidth of Internet links by sending UDP packets. To extend the estimation from per-hop to a path, *bprobe* [18] adopted the same philosophy to measure the capacity of an end-to-end path. As the previous methods usually leveraged consistent router message (e.g., ICMP), to reduce the time and data consumption, *tailgating* [53, 54] changed this behavior. Instead, it sent a large packet with a TTL set to expire at that link followed by a very small packet that will queue continuously behind the large packet. By doing this, *tailgating* sent an order of magnitude fewer packets than previous techniques while maintaining similar accuracy. From an analytic perspective, *pathrate* [30] examined the distribution of the conventional packet dispersions and developed a capacity estimation methodology that was robust to cross traffic effect. More recently, *CapProbe* [50] combined packet delay as well as dispersion measurements of packet pairs to filter out samples distorted by cross-traffic.
Achievable Throughput

For the most common metric, achievable throughout captures what a TCP flow(s) can obtain most from a network. As one of the pioneering works, cap [8] and Treno [67] implemented canonical frameworks to facilitate Bulk Transfer Capacity (BTC) measurement. Targeted at TCP performance, sting [96] used TCP to measure the packet loss rates between a source host and some target host(s). Later, the throughput test suits (e.g., ttcp [72], netperf [78], iperf [2]) were widely adapted for TCP performance measurement.

Today, many popular tools, e.g., SpeedTest [76] by Ookla, offer convenient measurement via web. There are many active research works that are looking for improving the estimation/prediction accuracy. PathPerf [71] adopted a machine learning approach (Support Vector Regression) to predict TCP throughput by using a combination of prior file transfers and measurements of simple path properties. On mobile networks, MobiPerf [41] and Mobilyzer [74] tried to tackle the problem by leveraging the power of crowdsourcing.

Available Bandwidth

Due to the intrusiveness and expensive time consumption of achievable throughput tests, there is a large body of research that attempted to divine network performance by using the metric of available bandwidth. Recalling the definition of available bandwidth at Equation 1.1, AB carefully calculates the portion of network bandwidth which is not occupied. To measure AB, it typically requires the active injecting of probe traffic. Among the various approaches, AB estimations can be largely classified into two categories: PRM (Packet Racket Model) and Packet Gap Model (PGM).

The Packet Gap Model (PGM) exploits the receiving packet gaps of successive packets as a reflection of the queueing effect experienced by the probe traffic. Then,
As one of the flagship works, IGI [105] developed a single-hop gap model that captured the relationship between the competing traffic throughput and the change of the packet pair gap for a single-hop network. Then, IGI used the model to propose packet pair techniques that estimated the competing bandwidth on the tight link of the path. Delphi [90] leveraged statistical knowledge of network dynamics provided by a versatile traffic model, *multifractal wavelet model* (MWM) [92], to improve the estimation accuracy. Similar works in the category of PGM include Abing [73], Spruce [106], *DietTopp* [49], [66] by Maryni and etc.

In contrast, *PRM (Packet Racket Model)* utilizes the comparison between the sending rate and receiving rate to compute the available bandwidth (shown in Fig 2.2). Informally, if one sends probe traffic at a rate lower than the available bandwidth
along the path, then the arrival rate of probe traffic at the receiver will match their rate at the sender. By finding such a tipping point that the receiving rate or the queueing delay starts to be less than the sending rate, one can approximate the available bandwidth. *PathLoad* [46] leveraged the one-way delay of probing packets. While increasing the probing rate, it looked for the observation of increasing delay as the indication of exceeding available bandwidth. Another work, *PathChirp*, improved the probing pattern to make the rate increasing as an exponential function thus reduce the search time. Similar works in this category include TOPP [68], PTR [105], SloPS [47] and etc.

2.1.2 Available Bandwidth Estimation on Mobile Networks

As this dissertation primarily focuses on the bandwidth characterization on mobile networks, in this part, I will give an overview of the AB estimation on mobile networks, including WiFi and cellular. Due to the highly dynamic bandwidth condition of the wireless channel, there are not many AB estimation approaches on mobile networks. Therefore, I will also look into the related works involving other forms of characterization beyond available bandwidth estimation.

**WiFi**

On WiFi networks, data transmission on a wireless channel can experience signif-
icant dynamics, e.g., channel contention, fast channel fading from movement, multi-rate transmission rates and etc. To capture these differences, *EXACT* [97] was one of the earliest works that considered the MAC layer overhead when conducting available bandwidth test. By assuming the MAC layer cost as constant for every single packet, *EXACT* varied the packet size to create different transmission time. Then it inferred the available bandwidth by eliminating the constant cost from the MAC layer. Furthermore, as an application case study, *EXACT* used the measured bandwidth to dynamically allocate the channel resource for a multi-hop mesh network to improve the bandwidth utilization. To deal with the different link behavior on wireless networks, another work *ProbeCap* [55] considered a more realistic wireless link model for bandwidth characterization. In this model, 1) the packets can be received in non-FIFO order and 2) the link capacity follows a multi-rate function. Technically, *ProbeGap* estimated the fraction of time that a link is idle by probing for the “gaps” in the busy periods and then multiplying by the capacity to obtain an estimate of the available bandwidth. Regarding the packet behavior on WiFi, Bredel et al. [15] conducted an in-depth analytic study to investigate the impact of several factors on WiFi, e.g., DCF (Distributed Coordination Function), protocol overhead and so on.

To revamp the traditional probe packet train designs on WiFi networks, Wbest [59] exploited the PRM to estimate the link capacity (using CapProbe [19] like approach) and then computed the available bandwidth. The works of exploring available bandwidth estimation on WiFi further includes [99], [52], and etc. The transmission scheme on WiFi networks has been gradually updated to improve the speed over years. Modern WiFi has adapted a new scheme called *frame aggregation* to assemble multiple packets into a single aggregated packet. This change fundamentally troubled the performance of many previous AB estimation methods, as conventional AB estimation works assumed a non-batch based scheduling. Recently, WBest+ [34] updated the previous work WBest (an AB test on WiFi) to mitigate detrimental
effect of frame aggregation on WiFi estimation. Compared with our work on WiFi–FMNC-WiFi \[104\], rather than engineering around frame aggregation, the proposed work leverages the embodied information from frame aggregation to achieve accurate AB estimation on modern WiFi.

**Cellular**

Due to the difficulties of conducting controlled experiments, the problem of bandwidth estimation on the cellular network has not been extensively studied compared to the WiFi networks. As a general open-source tool, MobiPerf \[41\] offered a comprehensive characterization of the quality of user experience (QoE) regarding many performance metrics, such as latency, data rate and so on. A recent work MONROE-Nettest \[70\] provided a configurable tool to study the influence of different parameters when measuring the speed on mobile broadband networks. Moreover, targeting on the available bandwidth estimation, Koutsonikolas \textit{et al}. \[52\] tested the traditional tool Wbest \[59\] under the CDWA x1 EVDO network. The result showed that the AB tool was infeasible in EVDO cellular network due to the short-scale dynamics in this type of networks. Later on, several works \[77, 82, 83\] aimed to revamp the classic PathChirp \[91\] on the cellular networks. NEXT \[82\] and NEXT-V2 \[83\] re-designed the chirp pattern to cope with the high dynamics on the mobile networks. To deal with the packet aggregation on LTE networks, a couple of works, PathQuick3 \[77\] and NEXT-FIT \[84\] tried curve fitting approaches to improve the accuracy of \textit{PathChirp} on the cellular networks. A recent work FLARP \[93\] further changed the chirp pattern into more aggressive form to fill up network buffer thus tease out the available bandwidth.

**Crowdsourcing**

One alternative of bandwidth estimation on mobile networks is to defer such
efforts to the crowd. Given the widespread prevalence of WiFi on mobile devices, crowdsourcing network performance would seem to be an excellent fit. In [35], the authors proposed MCNet, a tool that allows users to crowdsource Wi-Fi performance measurements. The authors in [35] proposed a similar study at the city scale and revealed several problems in WiFi deployments in public spaces. Furthermore, in the context of MPTCP, the works of [101] and [29] explored characterizing WiFi versus cellular networks. To the best of our knowledge, our paper is one of the first works to present a large-scale crowd-sourced deployment involving available bandwidth. We intend following the publication of the paper to anonymize and release all data via CRAWDAD which includes both our core FMNC AB test as well as ground truth AT observations.

2.2 Passive Mobile Network Characterization

In the previous section, I primarily focus on the proactive approaches on bandwidth estimation. In this section, I will shift the emphasis on reviewing the existing works on passive bandwidth/traffic characterization, including WiFi and cellular networks. I start with the works on WiFi networks. Since the method of passive characterization is widely applied in various scenarios and applications, I will iterate the related works under different sub-categories.

2.2.1 WiFi

WiFi traffic characterization can be tackled from multiple layers, including the physical layer, the MAC layer, or even the transport layer. At the bottom layer—the physical layer, there are works [35, 117, 121] that particularly target sensing the power on the WiFi spectrum. For example, Airshark [89] exploits commodity WiFi commercial-off-the-shelf (COTS) device to detect non-WiFi interference (e.g., microwave oven). Different from the physical layer works, the work in this dissertation
will explicitly focus on MAC-layer traffic.

Among the MAC-layer characterization works, based on the different standpoints to perceive the problem, we can further classify them into two groups: 1) AP side approaches and 2) client-side approaches. The AP side works [13, 81, 107] normally reply on the extensive infrastructure deployment, and the characterization is conducted from a point view of the controller. WiSe [81] collected wireless performance traces from 30 homes for a period in excess of 6 months to analyze the characteristics of the home wireless environment. More recently, [13] studied the data collected from approximately ten thousand radio access points, tens of thousands of links, and 5.6 million clients from one-week periods to provide a deeper understanding of the real-world WiFi traffic behavior.

The AP side works are expensive in terms of deployment and the characterization can only be restricted to the controlled vicinity. In this sense, client-side characterization can take advantage of the nature of ad-hoc and pervasive to provide a more realistic view of WiFi environment. The passive client monitor works have been extensively adopted for different purposes. [24] manipulated smartphones to monitor WiFi traffic in order to infer the human activities. Another smartphone-based work Pazi [88] uses indoor WiFi monitor fused with mobile crowdsource to achieve better localization. With large-scale studies, [62] explicitly seek the co-location between mobile users via Bluetooth information combined with WiFi scan. In addition, for security purpose, [22, 98] attempt to exploit WiFi monitor to detect rogue AP and potential attack. For the general purpose of traffic characterization, prior works [27, 63, 64, 119] in this category have different emphasizes. [63, 119] exploit multiple dedicated WiFi monitors to output merged view of WiFi traffic. They primarily focus on how to merge the measurements from different monitor to infer the uncaptured traffic. Those methods preassume that data packets can be easily captured without marginal loss. However, with the evolved WiFi, high-speed beamformed data trans-
mission can be hard to capture. Our method is particularly developed in the context of modern WiFi environment to give efficient controlled packet only characterization approach.

When targeting on available bandwidth estimation, many works leveraged the passively collected traffic trace at the MAC layer to infer bandwidth condition. For example, cPEAB \[108\] estimated available bandwidth by collecting i) the proportion of waiting and backoff delay, ii) packet collision probability, iii) acknowledgment delay, and iv) channel idle time compared to measurement period. cPEAB carefully considered the existences of potential hidden / exposed node, and yielded better simulation result than the similar prior works \[28, 95, 122\]. These type of works further include \[79, 95, 123\] and etc. As one of the most useful applications of bandwidth estimation, multimedia streaming usually depends on accurate bandwidth estimation in order to make judicious bitrate adaptation. IdleGap \[56\] was one of works that focused on bandwidth estimation on WiFi to facilitate streaming services. IdleGap took a cross-layer approach to leverage the NAV (Network Allocation Vector) information at the MAC layer to calculate the link idle rate to further infer the available bandwidth. Similarly, another work iBE \[120\] tried to improve estimation accuracy on 802.11 networks by considering the transmission delay at the MAC layer. When calculating the queue delays from the receiving probe packets, iBE counted the static cost from the MAC layer, including DIFS (DCF Interframe Space), RTS (Request To Send), CTS (Clear To Send) and etc.

### 2.2.2 Cellular

On cellular networks, due to the concern of intrusiveness of injecting traffic, many works \[20, 43, 69, 85\] focused on collecting/monitoring traffic trace passively and then inferring bandwidth condition. A popular passive characterization method is to utilize the traffic observations on the transport layer \[43, 69, 85\]. Huang et al. \[43\]
conducted an in-depth study of LTE network performance by deploying a large-scale monitor system. They investigated several traffic characteristics on the transport layer, such as flow duration, flow size and etc. Particularly, the TCP flow timing information was used to examine the available bandwidth. Recently, [69] tried to improve the bandwidth estimation accuracy by considering the impact of transport block allocation in LTE networks. In addition to transport layer observation, several works even looked deeper to leverage the physical layer information to infer bandwidth [20, 116]. For example, LoadSense [20] used physical layer power variations (e.g., RSPQ / RSRP in LTE) to infer the background traffic load on a serving cell. Although passive characterization is non-intrusive, it inevitably requires collaboration with network carriers or deploying and maintaining the monitor infrastructures.

2.3 Our Approaches and Contributions

To end the literature review in this chapter, in this section, I will discuss the similarities and differences of our proposed works compared to the existing works. In other words, I will outline the primary technologies I adopted as well as the major contributions I made in this dissertation. Similarly, the discussion will be divided into two subsections: proactive available bandwidth estimation on mobile networks (Section 2.3.1) and passive traffic characterization on WiFi (Section 2.3.2).

2.3.1 Proactive Available Bandwidth Estimation on Mobile Networks

In this dissertation, the first two out of three of our proposed solutions, FMNC-WiFi and FMNC-LTE, both focus on characterizing the bandwidth using the metric of available bandwidth. The intuition of adopting available bandwidth (AB) rather than achievable throughput (AT) is that AB estimation can be finished in a fast and efficient manner while consuming significantly less bandwidth and time than AT. The fast and efficient properties can enable many applications where AT estimation
is unfeasible to perform due to its cost. For example, for longitudinal network bandwidth monitoring, doing the AT test periodically can generate excessive traffic load on networks and potentially suppress the throughput of existing users. In addition, in the cases of bandwidth estimation driven adaptive services (e.g., video streaming), the bandwidth test is required to return the result quickly to drive the adaptation in a small time scale (e.g., 1~2 s). AT tests, which normally take around ten seconds (iperf3, SpeedTest), are inadequate for this purpose.

Among the many AB test models, I choose Packet Rate Model (PRM) because of its lightweight advantages on both data consumption and time cost. In contrast, most of the Packet Gap Model (PGM) works (except for Abing [73]) usually require collecting packet traces in a fairly long time window in order to discern certain distribution. While PRM can finish a test in one or several short packet trains. For example, as one of the flagship PGM works, IGI [105] can last 4~15 seconds for one test depending on the traffic load (studied in [37]). Wbest [59], as a typical PRM work, only costs less than one second. Note that the PRM works are not always fast. When the probe traffic is designed as a sequence of iterative streams, the time cost can be significantly high (*PathChirp* takes 15 seconds). Therefore, I adopted an “one-shot” design, where I only send one packet train consisting of multiple (hundreds of) packets to estimate the bandwidth.

The primary focus of the previous works on AB estimation was to design novel probe patterns for the wired networks. However, with mobile networks, the methods designed for wired networks may not be applicable to mobile networks due to the different link model. With the fundamental changes of data transmission behavior, many link properties will not be held on modern mobile networks, e.g., per-packet schedule, FIFO order and etc. A key feature that drives the current high-speed mobile networks is the *aggregation/batch*, where packets are no longer sent individually but are bundled together. The motivation of the proposed solutions is to revisit the
problem of AB test under the impact of aggregation. In next chapter, I will unfold the principal designs of the system as well as the detailed workflow of this system.

2.3.2 Passive WiFi Traffic Characterization

WiFi networks adopt a distributed manner to manage the channel access that the channel resource is competed by all nodes via CSMA/CA. The lack of the central controller makes network management inevitably challenging. To avoid the expensive infrastructure cost of an AP side solution, I manipulate the mobile clients as sensors to conduct the task of network characterization. In today’s high node density WiFi environment, the client-based approach can effectively leverage the power the crowdsourcing to deploy the characterization to a large scale setting. Similar to the existing client-side works, the capacity of passively characterization/estimation is benefited from the channel monitor functionality of WiFi clients. The general methodology is to exploit the traffic traces observed/eavesdropped in the vicinity of WiFi client(s) to infer numerous network statistics. Many of the existing works in this field were designing a characterization method for a particular purposes, e.g., security, video streaming, human sensing and etc. In the dissertation, I focus on providing a general framework for accurately characterizing WiFi traffic in an efficient manner.

Passive traffic characterization on modern WiFi networks faces a major technical obstacle: capturing all if not most data traffic in the passive mode is impractical now. With the beamforming and ultra-high data rate, a WiFi client can hardly eavesdrop most of the data traffic. Rather than observing the data traffic, in this dissertation, I utilize the control packets as a reference for instant network condition. Notably, I find one type of control packet – Block Acknowledge – embodies a rich set of information regarding the ongoing data transmission. In general, this method

\[\text{We have explicitly discussed the causes of traffic capture loss in Chapter 1.3.2.}\]
can be adapted into various application scenarios to provide accurate network traffic estimation. As a study case [103], I employed this method into a WiFi video streaming to help significantly alleviate the stalling under congested WiFi. Furthermore, our solution piggybacks the characterization into the existing WiFi scan operation, which barely adds any extra energy cost. As far as we know, those designs have never been attempted before. Moreover, I implemented the whole system onto commodity devices to conduct real-world performance validation. This works serves as a preliminary exploration towards the frame aggregation based WiFi characterization, which can be applied broadly into various WiFi channel-aware applications.
CHAPTER 3

FMNC: IMPLEMENTATION SYSTEM OVERVIEW

This chapter will introduce the implementation system, FMNC, as the platform where the two active bandwidth estimation methods are built upon, i.e., FMNC-WiFi (Chapter 4) and FMNC-LTE (Chapter 5). We will start this chapter with the requirements and goals of the system. Then we will introduce the FMNC architecture which is used to generate what we called a sliced, structured, and reordered packets. At last, we will briefly iterate on the code structure of the proposed system, including the key functional components and the data/call paths.

3.1 Goals

At a high level, the goal of FMNC is to provide a system that transmits a certain pattern of packet sequence. This task is normally done by creating dedicate sender and receiver as all of the existing works have done. Beyond this conventional approach, we aim at designing a generic system that suits to facilitate all types of packet train tests with several critical requirements as follows:

- **Client free:** The dependence on the dedicated application largely precludes the AB test tools from scaling over a large number of users. The designed system targets at achieving a client-free implementation. The feature is crucial to leverage the power of crowdsourcing.

- **Scalable to serve many clients:** The traditional bandwidth test tools are largely built for serving one client only at a time. We aim at a many-to-one solution that a server can serve multiple clients simultaneously with proper coordination.
• **Resolve quickly with one shot:** Path characterization must happen on the order of hundreds of milliseconds, not tens of seconds in order to be actionable. Particularly, we select to form the probe traffic as one-time direct measurement, rather than iterative streams.

### 3.2 FMNC Architecture

![](image)

Figure 3.1. FMNC work flow.

#### 3.2.1 A Customized HTTP Server

To satisfy the requirement of client-free design, the system must confine to working in the existing network stack. Therefore, the system disguises as a HTTP server that handles web request. In this sense, the network characterization can be embedded into the web content delivery. As shown in Fig 3.1, the FMNC process begins when a client initiates a TCP connection to make an HTTP request to the FMNC server. The normal TCP 3-way handshake occurs followed by an HTTP GET mes-

---

1 The usage of a non-standard port may be desirable to avoid proxying effects
sage which contains settings for FMNC embedded as parameters in the request. Then the server arranges the packet train to send back as the web content. From the client’s perspective, there is no difference from simply viewing a website. After transmitting the probe packets, the FMNC server will close the connection by initiating 4-way FIN handshake. Then the server will use the packet logs collected during the test to estimate the network bandwidth. Fig 3.2 demonstrates the screenshot of an FMNC test view on the client side. Note that we embed all the packet train description parameters in the url.

Since the normal web server can not control over how packets are scheduled in the lower layers (i.e., the transport layer), we have to build a customized transport layer. This is done by utilizing libpcap and raw socket in C++. As shown in Fig 3.3, we use libpcap to get a copy of any inbound traffic. To transmit outbound traffic, we construct packets from scratch and write into the network adapter directly by using the raw socket. Note that in order to avoid the operating system kernel intervening the probe traffic, we intentionally block out the traffic on the port used by FMNC (by configuring the firewall iptables).
3.2.2 Sliced, Structured, and Reordered Packet Sequence

Once the test request is parsed, the server will construct the probe traffic to send back. To shape the TCP traffic into a certain pattern of packet sequence, we design this *slicing and reordering* mechanism. Given the flexibility of fully controlling the transport layer, we can compose any form of packet sequence and schedule to send out as desired. Figure 3.4 presents an overview of the packet train as constructed. In conjunction with the figure, we expand on what we mean by the terms, sliced, structured, and reordered. We first slice the web content into different size of packet data payload. We then use the TCP ACK responded from the client to infer the probe packet reception statistics. However, such an approach would result in an insufficient number of observations, because typical TCP behavior will produce a ratio of data to ACK packets on the order of 2:1 or beyond. To augment this number, we apply the reordering from TCP Sting [96]. The sliced packets within a given window are reordered whereby the first packet is shuffled to the end of the window. This effect as originally identified by TCP Sting leverages TCP Fast Retransmit to yield a 1:1 ratio of data to ACK packets. In short, TCP Fast Retransmit engages if the same ACK number is seen three times implying a missing packet. By shuffling the packet to the end of the window, the first packet appears to be missing (lost) and hence we tease out additional ACKs thereby dramatically increasing our observation opportunities.
As we control the server via *libpcap*, we are free to ignore the spurious ACKs.

### 3.2.3 Bandwidth Estimation

The FMNC server includes a well-designed packet logger that organizes and stores the packet traces occurred within a test. Since only the information at the server side is gleaned, we have to infer the packet reception statistics in an oblique way. The client-side information can be inferred from two major resources: 1) TCP ACK receiving timestamp and 2) *TsVal* from TCP Timestamp Option. If the uplink (the link from client to server) is uncongested, then the TCP ACK receiving timestamp is a good approximation of the probe packet receiving timing. On WiFi networks, the wireless channel is shared by both uplink and downlink. The traffic on WiFi experiences a relatively symmetric channel. Under this circumstance, it is fair to use TCP ACK receiving time to infer the packet reception information at the client. While the uplink is significantly different from the downlink, e.g., the cellular network divides the uplink and downlink to control their channel resource separately, we are...
left with a secondary choice of using TCP Timestamp Option values to infer the client side information. The TCP Timestamp Option provides a coarse clocking scheme to measure the packet intervals. The granularity ranges from 10 ms/tick to 1 ms/tick. In the later chapter (Chapter 5), we will elaborate how to use this timestamp in the case of cellular network bandwidth estimation. Once the packet timing information is calculated, the server can proceed to run estimation algorithm to tease out bandwidth result. The algorithm designs are specific to different probe trains under different networks. We will unfold the detailed estimation algorithms on WiFi (Chapter 4) and cellular networks (Chapter 5).

3.2.4 Test Coordination

When handling multiple test requests from clients, we adopt the design akin to CSMA/CD (Carrier-sense Multiple Access with Collision Detection). Whenever the server needs to send out a probe packet train, it requires to sniff the traffic on the popular ports (e.g., HTTP 80, HTTPS 433). If there is ongoing traffic, then the server defers the transmission and waits for a certain period (i.e., 1 s) to retry again. Otherwise, if no traffic is detected, the server will proceed to send the probe packets. This simple design will eliminate test collision and undesired interference.

3.3 Code Structure

The FMCN server code is written in C++ which consists of 16,000 lines of source code. Among the code, there are 4,000 lines dedicated to composing IP/TCP packet from scratch bit by bit. The rest handles the core functionality of handling traffic and generating packet train. In Fig 3.5 we plot the overview diagram of the timer-based code structure of the server. The FMNC server receives incoming traffic from the libpcap and the TCP_Handler parses the traffic to decide further actions. All the traffic is handled by a unit of a TCP connection. If the TCP_Handler detects the
incoming traffic is for initiating a new connection (i.e., receiving TCP SYN), it will prepare the response packets (i.e., SYN-ACK and ACK) to complete the three-way handshake protocol. Otherwise, if the TCP_Handler realizes the incoming traffic is for an existing connection, it will query the timer-based event manager to further complete the probe transmission.

We use the timer-based event manager to 1) separate the processes of transmitting probe traffic into several stages and 2) carefully achieve high time resolution when scheduling packet transmissions. Overall, to generate a properly formatted packet sequence, the event manager first call the do_StartSlicing to prepare the packet train parameters which is used to describe a packet pattern. Basically, this function takes the web request (a.k.a. HTTP GET) as input to decide the following terms: packet length, packet size and packet gap for each packet. After finishing the parsing, the event manager will go into next step do_Slicing. In this step, the server will carefully pack the web content into several packets with desired packet sizes. To control the packet gaps, the event manager will assign a nanosecond-level timer to send each of
the packets. The timer is a busy-waiting thread that consistently checks whether the current system time matches any scheduled event start time. If yes, it calls another thread to finish the scheduled event.

Figure 3.6. A snapshot of FMNC packet log record.

It is noteworthy that, unlike the regular TCP stack, our server doesn’t handle loss and retransmission. There are several reasons we choose not to do this: 1) as the packet train has to follow precise timing, any retransmission in the middle of sending probe packets can break the pre-defined timing pattern; 2) the loss information per se is meaningful to reveal network performance. After sending out all packets, the event manager will schedule to send a FIN packet with certain fixed delay. While sending/waiting packets, the FMNC server records the timing information regarding every single packet transaction occurred in a test. Unlike the general packet dump tool (e.g., tcpdump), the packet logger can carefully categorize the packets into differ-
ent groups, e.g., sending/receiving, data/control and etc. We use the tree structure to store the packet traces into xml file. Fig. 3.6 is an example snapshot of one connection/test. In the figure, we also indicate the sequence number (SN) of the sent packets where we see how the packets are reordered (in this case, they are reordered every five packets).

As a summary, we introduced the implementation details and workflow of the packet probing system FMNC. The system provides a generic platform to implement any packet train/sequence based design without installing a dedicated client application. In the next two chapter (Chapter 4 and Chapter 5), we will introduce the two available bandwidth estimation methods for modern mobile networks that are built upon this system (i.e., WiFi and cellular).
CHAPTER 4

FMNC ON THE WIFI NETWORK

In this chapter, we will introduce the bandwidth estimation solution on WiFi networks. By understanding and further exploiting the rich information embedded in frame aggregation, we design a novel approach to utilize the aggregation intensity as an effective link congestion indicator. Together with the probe packet train technique, the designed AB estimation approach conquered the problem of estimating AB in the presence of frame aggregation which fails all of the previous AB methods. We implement the system AIWC (Aggregation Intensity based Wifi Characterization), which can achieve rapid and efficient available bandwidth estimation for the modern WiFi. With the proof of concept system, we demonstrated that AIWC can achieve high accuracy and robustness across various network scenarios.

The rest of the chapter is organized as follows. To start, Section 4.1 demonstrates the problem occurred on modern WiFi when applying the conventional AB estimation method. Followed by Section 4.2 we introduces the new metric to facilitate AB estimation under frame aggregation. In the Section 4.3 we elaborates the probe packet train format design coupled with the newly-developed metric. Then in the Section 4.4 we iterates on the extensive experiments we conduct for performance evaluation. Finally, in the Section 4.5 we demonstrate the empirical results we collected over hundred mobile users in the wild.

\[1^*\]This works has been published in IEEE SECON 2017 [104], ACM MobiCom (demo) [102].
4.1 Problem Demonstration

Before we discuss the details of the proposed work, we present an overview of frame aggregation and continue our motivation by demonstrating how frame aggregation affects prior AB methods.

4.1.1 Frame Aggregation Overview

The technique of frame aggregation was proposed in 802.11e for the purpose of increasing the WiFi throughput. By assembling multiple packets for transmission as one aggregated frame, one can reduce the overhead of physical header as multiple packets share one header. In tandem with Block Acknowledgement (BA), only one ACK at the MAC layer is required to respond per aggregated frame which improves the MAC layer efficiency. Frame aggregation can be operated at two levels: at the aggregate MAC protocol service unit (A-MSDU) and at the aggregate MAC protocol data unit (A-MPDU). The general principle is to allow multiple data units that are destined to the same receiver to be assembled and sent out as one aggregated frame. As an example, Fig 4.1 breaks down the components of an A-MPDU frame. The same principle is applicable to A-MSDU with a different unit (MSDU) as the subframe.

Figure 4.1. Illustration of frame format via A-MPDU.
The two aggregation mechanisms are operated at different levels with the A-MSDU near the top of the MAC layer and the A-MPDU near the bottom of the MAC layer. These two mechanisms may be employed together as specified in 802.11n/ac [51]. Overall, when forming an aggregated frame, the two-level aggregation process first needs to wait for the transmission of previous packets in the queue if any; then hold for an additional pre-defined delay to wait for any incoming traffic that is destined for the same address. The aggregation is complete if any of the three conditions occur: 1) the size of aggregated frame reaches the maximum ($P_{\text{max}}$); 2) the estimated transmission time of the aggregated frame reaches the maximum ($T_{\text{max}}$); or 3) timeout of the pre-defined delay ($D_{\text{ag}}$).

4.1.2 AB Estimation under the Impact of Frame Aggregation

Unfortunately, under frame aggregation, the assumption of FIFO per-packet based scheduling does not hold. The aggregated transmissions behave like a batch scheduler which dramatically distorts the timing characteristics of received packets. To understand how frame aggregation affects AB estimation, we conduct the following experiment.

Consider a traditional probe sequence with a fixed packet size and packet gap transmitted over a congested WiFi link. After the probes pass through the WiFi link, one can calculate the packet gap and packet rate at the receiver. Particularly, the packet rate can be computed with two methods: naive (dividing packet size by packet gap) and jumbo-based [34]. Wbest+ [34] suggested considering aggregated packets as a single jumbo packet for calculating packet rate$^2$.

As shown in Figure 4.2, we can contrast the impact of FA from 802.11n case (Fig 4.2(b)(d)) with the case without FA under 802.11g (Fig 4.2(a)(c)). With iden-

$^2$A jumbo packet can be recognized by finding a series of consecutive packets whose packet gaps under certain threshold (e.g., 300 $\mu$s).
Figure 4.2. The observed receiving packet gap and packet rate in a packet sequence under 802.11g v.s. 802.11n. The dashed line indicates the probe rate.

tical probes, the received packet gap under 802.11n (Fig 4.2(b)) presents a much more bursty and bimodal pattern than under 802.11g (Fig 4.2(a)). The reason is that, when frame aggregation is applied, the multiple packets that were assembled in an aggregated frame arrived at the receiver at the same time. These packets appear to have negligible packet gaps between each other. When computing the packet rate, those extremely small packet gaps are translated into high packet rates. Therefore, Fig 4.2(d) shows a similar bursty pattern for the naive method. Although the jumbo idea from Wbest+ helps smooth the spiky pattern, the calculated packet rate presents high dynamics with unacceptably large variation. According to PRM, when passing through a congested link, the received packet rates should be consistently less than
the probe rate as in Fig 4.2(c). However, this principle breaks with frame aggregation. As shown in Fig 4.2(d), the received packet rate hovers with over half of the received packets having a packet rate greater than the probe rate. Also, from the perspective of PGM, the majority of compressed packet gaps can not be used to infer cross traffic anymore.

4.2 Exploiting Frame Aggregation for Link Characterization

While the notion that FA breaks AB estimation has been explored in the literature [34], the prior approach has been to work around FA. In contrast, by carefully studying the characteristics of FA regarding probe traffic (Section 4.2.1), we show that the frame depth of FA can be manipulated to detect link congestion (Section 4.2.2).

4.2.1 Frame Aggregation Characterization

Consider a sequence of probe packets with packet size $P$ and packet gap $G_{snd}$ that are sent from a server to a client via a WiFi link with frame aggregation enabled and a data rate of $R$. We start with considering the simple case when no cross traffic is present. According to the aggregation process mentioned in earlier Section 4.1.1, the first packet that arrives in the AP queue will be held for a pre-defined delay time $D_{ag}$ to wait for more traffic. Any probe packet that arrives during this time will be assembled into the same aggregated frame. In order to describe the degree of the aggregation, we introduce a metric—Aggregation Intensity (AI)—to denote the number of packets assembled in an aggregated frame on a WiFi link. The AI in

\[^3\]It should noted that, instead of the physical layer transmission rate, the data rate is a transport layer measurement.
this case can be expressed as:

\[ AI = \left\lceil \frac{D_{ag}}{g_{snd}} \right\rceil \]  

(4.1)

When the cross traffic is injected to compete with the probe packets, the cross traffic will introduce additional queuing delay for the probe packets. Therefore, Eq (4.1) can be further extended as:

\[ AI = \left\lceil \frac{D_{ag} + T_Q}{g_{snd}} \right\rceil \]  

(4.2)

\[ T_Q = f(u_x) \]

subject to  \[ AI \cdot P \leq P_{max}, \quad \frac{AI \cdot P}{R} \leq T_{max}. \]

where \( T_Q \) is the extra queuing delay resulting from the cross traffic. \( T_Q \) is a monotonically increasing function of the link utilization \( u_x \) occupied by the cross traffic. \( AB \) can be calculated as \( AB = C(1 - u_x) \) with the link capacity of \( C \). As the cross traffic load increases (\( AB \) decreases), \( T_Q \) will grow as well. This will lead to more probe packets being aggregated in a frame which causes the growth of \( AI \). Furthermore, \( AI \) has an upper bound by the maximum size \( P_{max} \) and maximum transmission time \( T_{max} \). The function \( f(\cdot) \) helps transform the variation of \( T_Q \) to the impact reflected on the load of cross traffic. Ultimately, the relationship between \( AI \) and \( AB \) is determined by the characteristics of the function \( f(\cdot) \) between \( u_x \) and \( T_Q \).

Under the batch scheduling of frame aggregation, the function \( f(\cdot) \) does not follow a linear pattern as under FIFO scheduling. To explain the difference, we show a simple example in Fig 1.3 of two groups of packets to be sent to two clients on WiFi link. The \( A_i \) packet denotes the \( i \)-th packet destined to client \( A \), and \( B_i \) is destined to client \( B \). In this particular case, since the packet at the head of queue is addressed to \( A \), all of \( A \)'s packets will be sent as an A-MPDU before \( B \)'s packets. This behaves
like prioritizing the A’s packets over B’s packets.

For the case when the probe traffic competes with the cross traffic, if one assumes the queue is not empty when a probe packet arrives, we can apply the case in Fig 4.3 to consider the cross traffic packets as the A’s packets and the probe packets as the B’s packets. Following this intuition, the function $f(\cdot)$ can be approximated as the queuing delay of the probe packets (low-priority) with regarding to the cross traffic (high priority) load. Notably, previous works on priority queueing system [23, 111] have characterized the relationship between packet queuing delay of low-priority packets (i.e., the probe packets) and the traffic load of high-priority packets (i.e., the cross traffic). Due to space limitations, we do not describe the details of the derived formulation. The general pattern of this function can be depicted as the curve in Fig 4.4.

The curve in Fig 4.4 can divided into two zones: the ① zone where the probe traffic load plus the cross traffic load is less than the link capacity (the probe rate $< C(1 - u_x)$), and the ② zone where the probe traffic load plus the cross traffic load is greater than the link capacity.

Figure 4.3. An example of scheduling under frame aggregation.
load is greater than link capacity (the probe rate $> C(1 - u_x)$). Notably, in the first zone, the increasing slope of the packet delay with the cross traffic load is quite low. However, when the link starts to be saturated in the second zone, the queuing delay dramatically rises. By applying this pattern of $T_Q$ into $AI$ in Eq (4.2), one can expect that the AI would stay relatively consistent when link is uncongested; and it will grow quickly once link congestion occurred.

**Experimental Verification**

We conduct an experiment to verify the conjecture. By sending packets from a sender to a receiver via a WiFi link with a fixed packet rate, we can capture the observation of the receiver. The calculation of $AI$ follows a threshold-based method: if two consecutive received packets have a negligibly small packet gap ($G_{rvd} < \theta$, i.e., $\theta = 400 \, \mu s$), it implies the corresponding probe packets were aggregated. The
$AI$ can then be measured by counting the number of consecutive packets that are aggregated.

Without any cross traffic, we can vary the probe packet gap from 100 $\mu$s to 1100 $\mu$s. The probe rate is fixed to 10 Mb/s which is less than the link capacity. In order to fix the probe rate, we adjust the packet size proportionally according to the packet gap. We ran this experiment under 802.11n and 802.11ac with different types of AP WiFi chips (Atheros, Broadcom, Ralink). For each setting, we repeat the experiments 10 times and record the mean value and standard deviation. As the general pattern of curves hold for all the WiFi chips, we plot the result from Atheros. Fig 4.5 shows that, when the probe packet gap $G_{snd}$ decreases, $AI$ rises even with no link congestion. We also plot the curve of the theoretical result by setting the value of $D_{ag}$ to 1000 $\mu$s in Eq (4.1). As shown, both of the curves of 801.11n and 802.11ac match well with the theoretic result.
In order to add cross traffic, we injected constant bit rate (CBR) UDP traffic from the server to another WiFi client on the same AP (802.11n in this case). The probe rate of 10Mb/s consumed roughly about 10% of the link. In Fig 4.6, we can see that $AI$ monotonically increases with the cross traffic link utilization $u_x$. Particularly, when $u_x$ approaches 0.9 where the probe traffic starts to saturate the link, $AI$ rises quickly with a steep slope. This observation matches well with the tendency of function $f(\cdot)$ in Eq (4.2). Notably, with a larger packet gap setting, the zone pattern is clear. The reason is that under the WiFi environment, the queuing delay $T_Q$ is further affected by the backoff, physical layer losses, and other wireless transient impacts. The small packet gap is quite sensitive to the dynamics which results in a fast growing $AI$. 

Figure 4.6. Aggregation Intensity v.s. cross traffic load w.r.t. probe packet gap ($G_{snd}$).
4.2.2 Manipulating Frame Aggregation to Detect Link Congestion

Inspired by this observation, we explore how to manipulate $AI$ to detect link congestion. Intrinsically, by sending a probe packet sequence with packet size $P$ and packet gap $G_{snd}$, the observed $AI$ would lie in the first zone if the current $AB$ is greater than the probe rate $\frac{P}{G_{snd}}$. When the probe traffic exceeds the current $AB$ and triggers link congestion, the $AI$ would enter into the second zone and increase dramatically. Therefore, by simply checking if the observed $AI$ increases by a certain threshold ($AI_{con}$) from the baseline value $AI_{base}$, we are able to tell if the AB of a WiFi link is less than the probe rate $\frac{P}{G_{snd}}$. The expression can be summarized as follows:

$$AI - AI_{base} \begin{cases} > AI_{con} & \text{if } AB < \frac{P}{G_{snd}}, \\ \leq AI_{con} & \text{otherwise} \end{cases}$$

(4.3)

We choose to set packet gap $G_{snd}$ to be a large value such that packet gap $G_{snd} > D_{ag}$. The large gap setting helps gain robustness against wireless dynamics to better utilize the pattern from Fig 4.6. With $G_{snd} > D_{ag}$, we intentionally force the aggregation baseline to 1, which means the packets should be separated ideally when no cross traffic is present.

Extending High Probe Rates

In order to explore different values of $AB$, we must change the probe rates. With a fixed packet gap, we must vary the packet size $P$ to tune the probe rate. However, due to the constraint of the maximum transmission unit (MTU) for packet size, we are prevented from generating a higher probe rate in a single packet. To overcome this limitation, we propose a “concatenating” method: transmitting multiple ($N$) packets with a negligible packet gap $G_c$. With $G_c \ll D_{ag}$, we deliberately force the $N$ packets to be aggregated at WiFi link. Then, the intentionally concatenated packets
can be considered as a large packet with a packet size of $N \cdot P$ and the probe rate is $\frac{N \cdot P}{G_{\text{snd}}}$. In this case, the aggregation baseline $AI_{\text{base}}$ becomes $N$. Note that this design presumes the cross traffic on wired link(s) would not break the concatenated pattern. As we assume the bottleneck link is the last hop WiFi, the concatenated packets should not be coalesced with the cross traffic on wired link(s).

**Experimental Verification**

With the same experimental setting as before, we evaluate the principle of $AI$ as a link congestion indicator. Utilizing a fixed packet gap $G_{\text{snd}} = 1100 \mu s$, we generated three levels of probe traffic with probe rates of 10 Mb/s ($N = 1$), 40 Mb/s ($N = 4$) and 70 Mb/s ($N = 7$). For each level of probe traffic, we also varied the cross traffic load. Fig 4.7 shows the results of the observed $AI - AI_{\text{base}}$ under the different loads of cross traffic. Link congestion occurs when the probe rate plus cross traffic load
is greater than the link capacity. The observed $AI - AI_{\text{base}}$ matches perfect with the theory of Eq 4.3. For the high $AI_{\text{base}}$ with the high probe rate, the $AI$ can go higher than the lower probe rates. To further explore the congested and uncongested cases, we collected the trace from the experiment and plotted the empirical CDF of $AI - AI_{\text{base}}$ according to whether link congested or not. When AB is less than the probe rate, the link is congested; otherwise, the link is uncongested. As shown in Fig 4.8, there is a clear separation between the congested and uncongested cases. This implies that by sending a probe sequence with probe rate $R$, one can detect if the current AB on a WiFi link is less or greater than $R$ based on $AI$ observation.

4.3 Probe Packet Train Design

To utilize $AI$, we use the Packet Rate Model (PRM) to design a probe packet train for the purpose of AB estimation. In this section, we elaborate on the designed
probe packet train and the corresponding estimation algorithm.

4.3.1 Probe Packet Train Format

We design a probe packet train consisting of multiple sub-trains with monotonically increasing probe rates across the sub-trains. The intuition is that the problem of AB estimation can be transformed into finding the maximum probe rate sub-train that does not experience link congestion. We employ a linearly increasing pattern for the sub-train probe rates. By setting a minimum $R_{\text{min}}$ and maximum $R_{\text{max}}$ as the probe rate of the first and the last sub-train, we are able to search the potential AB in the range of $(R_{\text{min}}, R_{\text{max}})$. Fig 4.9 plots the basic format of the probe packet train with the labelled parameters. Table 4.1 summarizes the the notation and description of the design parameters. Notably, the sub-train $i+1$ depicts the case where the high probe rate requires concatenating multiple ($N = 2$ in this case) packets.

![Figure 4.9. Probe packet train format with parameters labeled.](image)
TABLE 4.1

PROBE PACKET TRAIN DESIGN PARAMETERS DESCRIPTION.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{snd}$</td>
<td>The packet gap between non-concatenated packets</td>
</tr>
<tr>
<td>$G_c$</td>
<td>The packet gap between concatenated packets</td>
</tr>
<tr>
<td>$G_{tr}$</td>
<td>The packet gap between neighbour sub-trains</td>
</tr>
<tr>
<td>$M$</td>
<td>The total number of sub-trains</td>
</tr>
<tr>
<td>$L_i$</td>
<td>The number of packets in the $i$-th sub-train</td>
</tr>
<tr>
<td>$P_i$</td>
<td>The packet size for the $i$-th sub-train</td>
</tr>
<tr>
<td>$R_i$</td>
<td>The probe rate of the $i$-th sub-train</td>
</tr>
<tr>
<td>$N_i$</td>
<td>The number concatenated packets to achieve $R_i$</td>
</tr>
</tbody>
</table>

4.3.2 Parameter Setting

We classify the design parameters into two groups: **system-defined** and **customized**. The first group ($G_{snd}$, $G_c$, $G_{tr}$) are set in order to satisfy the design principles. The second group ($M$, $L_i$) can be tuned by the users for their own goals, e.g., adjusting data cost and tuning estimation resolution.

**System-Defined**

We set $G_{snd} = 1100 \mu s$ to maximize the effectiveness of AI with a minimal time cost. $G_c$ is set to $20 \mu s$ to make sure the intentionally concatenated packets can be aggregated at WiFi link. In addition, due to the dynamics of WiFi links, the transient “hiccups” on a sub-train may contaminates the subsequent sub-trains as the temporal queuing effect may not be relieved immediately. In order to mitigate this impact, we add an inter-sub-train gap $G_{tr} = 2000 \mu s$ to reduce the queuing influence of a sub-train on subsequent sub-trains.
Customized

The setting for the number of packets in a sub-train $L_i$ is a trade-off between accuracy and cost. A large $L_i$ can achieve high accuracy with more sample points. It also potentially injects more traffic that may disturb the existing traffic. We recommend setting $L_i$ proportional to the probe rate which uses more probe packets for the higher probe rate. The number of sub-trains $M$ decides the granularity of the test result by determining the searching step $\Delta R = \frac{R_{\text{max}} - R_{\text{min}}}{M-1}$. Given a certain $L_i$, a large $M$ offers good estimation resolution but at expensive data cost.

4.3.3 Estimation Algorithm

Given the observed average $AI$ for each sub-train, $AB$ can be approximated by finding the maximum probe rate sub-train which did not experience link congestion. Unfortunately, the WiFi link in practice may experience some transient effects that cause $AB$ to drop for a short period of time. In order to compensate for this noise, we designed a simple scoring-based algorithm. The intuition of the algorithm is to seek consensus across sub-trains for the observation of the link.

For the $i$-th sub-train, based on the Eq (4.3), if $AI_i - N_i < AI_{\text{con}}$ (where we substitute the $AI_{\text{base}}$ with $N_i$), then the sub-train $i$ does not encounter link congestion so that we know $AB > R_i$. For all the sub-trains ($i$) with $AB > R_i$ detected, we score them based on the observations of their previous sub-trains $j$ ($j \in [1, i-1], R_j < R_i$). If the sub-train $j$ detected $AB > R_j$ which agrees with the sub-train $i$, we accumulate a positive score of $\frac{1}{(i-j)}$ to sub-train $i$; otherwise, a same value of negative score is added. By assigning the weight as reciprocal of sub-train index difference, we value the consensus of the closely transmitted sub-trains more than of sub-trains far apart. Finally, $AB$ can be approximated with the probe rate of the sub-train with the highest score. If $AB$ is not in the range of $[R_{\text{min}}, R_{\text{max}}]$, the result will be given as a classification: 1) if all sub-trains have $AB < R_i$, then it returns $AB < R_{\text{min}}$; 2) if
$AB > R_i$ observed for all sub-trains, it returns $AB > R_{\text{max}}$.

**Backward Compatibility**

When the detected $AI$ is consistently low, we assume frame aggregation is not present on the WiFi link (e.g., on 802.11g link). Therefore, the algorithm falls back to a conventional approach using the received packet rate to detect link congestion. The *scoring*-based algorithm can be still applied upon the sub-trains.

### 4.4 Experimental Evaluation

Our experimental environment was set up as follows. The AIWC client was written using a simple shell script (ex. *curl* requests) and was executed on a laptop running Ubuntu 14.04 with multiple WiFi adapter options (Ralink RT3950 for 802.11g/n SISO, EdiMax EW-7822UAC for 802.11n/ac MIMO). The same setting was used for a competing client to generate cross traffic. The AIWC client as well as the competing client were connected to a TP-LINK Archer C7 AP (802.11ac capable AP) with OpenWrt installed. The AP was connected through a local Gigabit Ethernet switch to a computer providing NetEM-based[39] emulation for link control. The NetEM box was used to emulate many network settings with the *tc qdisc*, such as delay, packet loss, rate limitation and etc.

The AIWC server was run on a separate laptop using the aforementioned *libpcap*-based C++ server and was also connected to the NetEM box via Gigabit Ethernet. To emulate a real-world environment, a 40 ms round trip time was added on the path from the server to the clients. Unless otherwise noted, the probe packet train parameters were set as $M = 10$, $L_i = \text{max}(20, 5 \cdot N_i)$. Based on experiment in Fig[4.8] we set $AI_{\text{con}} = 3$ for the estimation algorithm.

The cross traffic through the experiment is originated at the server in order to introduce congestion across the WiFi link. Cross traffic was generated by the *Dis-
Figure 4.10. 802.11g (2.4GHz) without frame aggregation.

Distributed Internet Traffic Generator (D-ITG) [14] for a fine granularity of control of packets. In order to obtain the ground truth of AB, an iperf3 UDP flow was used to measure the WiFi capacity. Then AB can be calculated by subtracting the throughput of cross traffic from the link capacity.

4.4.1 Cross Traffic

We begin by comparing the available bandwidth accuracy of AIWC versus prior AB estimation works, including PathChirp [91], Spruce [106], and WBest+ [34]. PathChirp is a typical PRM (Packet Rate Model), and Spruce is a typical PGM (Paket Gap Model). WBest+ derived from Wbest [59] is the only prior work that is aware of frame aggregation and is particularly designed for WiFi. All three methods were configured as specified in the papers (e.g., Spruce requires being informed of the bottleneck capacity).
Figure 4.11. 802.11n (2.4GHz) MIMO with frame aggregation.

By generating different loads of CBR UDP cross traffic from the server to the competing client, we are able to vary the AB ground truth. We begin with a direct comparison by sweeping the available bandwidth across 802.11g, 802.11n and 802.11ac. Each method was run 20 times for each point. For WiFi links with different capacities, we used different AB search range for our probe packet train with $(R_{\text{min}}, R_{\text{max}})$: (1, 20) Mb/s for 802.11g SISO with 20 Mb/s link capacity, (10, 80) Mb/s for 802.11n 2x2 MIMO with 80 Mb/s link capacity, and (10, 100) Mb/s for 802.11ac 2x2 MIMO with 140 Mb/s link capacity.

In Fig 4.10, 4.11 and 4.12, we plot the results regarding to the ground truth for each of the methods. Note that the diagonal dashed line in the figures indicates the optimal case where the estimation result is equal to the ground truth. Starting from the case of 802.11g with no frame aggregation, in Fig 4.10 AIWC and Wbest+ show better performance. Since Spruce and PathChirp are designed specifically for
wired links with targets of large bandwidths on the order of hundreds of megabits, the approaches struggle to sense the subtle AB variations on the WiFi link. For AIWC, when consistently low $AI$ was detected, AIWC adopts the backward compatible algorithm mentioned in Section 4.3.3. As the target range was set to 1-20 Mb/s, AIWC achieves a fine granularity with a resolution of 2 Mb/s that helps outperform Wbest+. Notably, it is difficult for all the methods to achieve high accuracy when AB is very low, e.g., 0-4 Mb/s. The normal fluctuations of WiFi link capacity makes the AB in this low region difficult to be consistent.

In Fig 4.11, in the presence of frame aggregation on the 802.11n MIMO link, Spruce and PathChirp are largely blind to the variations of cross traffic. Spruce as a PGM method looks for the increasing of packet gap as an indicator of link congestion. However, as the aggregation process compresses many of the packet gaps, it misleads Spruce to believe there is no cross traffic present. For PathChirp, it expects the
Figure 4.13. Queuing delay observed by PathChirp without (a) and with (b) frame aggregation.

The queuing delay of probe packets should be mostly above zero as shown in Fig 4.13(a). PathChirp considers the negative queuing delay as coalesced impact from severe cross traffic. When frame aggregation is adopted, it makes the majority of queuing delay show negative as shown in Fig 4.13(b), which misleads PathChirp to think the cross traffic load is significant. Therefore, PathChirp consistently returned a low AB estimation around 10Mb/s. For WBest+, it barely reacted to the variation of AB under 802.11n/ac. This occurs due to the fact that frame aggregation heavily distorts the received rate as discussed in earlier Section 4.1.2. As a method that relies on the received packet rate to detect cross traffic, WBest+ can not effectively overcome the disturbing from FA even with increased number of probes. Compared to the other methods, AIWC is the only approach that reveals the changes in AB. AIWC shows the best accuracy by following closely the pattern of the ideal case (dashed line). Notably, due to the conservative design of the scoring-based algorithm, AIWC prefers underestimation rather than overestimation.

**TCP Cross Traffic**

The purpose of using UDP cross traffic was to generate a consistently stable AB. However, in practice, TCP traffic is likely the major driver of cross traffic. Hence,
we continue to conduct a TCP cross traffic evaluation. In contrast to the UDP CBR case, the ground truth of AB will present an on/off pattern such that when TCP traffic is on, the AB should be zero; when TCP traffic is off, the AB should equal to the link capacity. Further, the AB will be between zero and the link capacity during TCP slow start. With the same experiment settings above, we transmitted back to back TCP flows with different flow sizes. We used an 802.11ac link with 40 AB tests repeated for each flow size.

As shown in Fig 4.14, we plot the AB test result as class distribution by breaking down the result into five classes. Generally, the flow size will decide the time fraction of the link being saturation. When the flow size is small, the WiFi link remains largely unconsumed which leads to 80% of the AB test returning $>100$ Mb/s for a 1KB flow size. When the flow size increases, the AB result moves towards the lower regions. As the flow size reached to 160MB, 60% of tests returns AB $<10$ Mb/s, which
**TABLE 4.2**

COST COMPARISON ACROSS DIFFERENT METHODS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time(s)</th>
<th>Traffic(KB)</th>
<th>Pkts #</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIWC ((R_{\text{min}}, R_{\text{max}}))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>((1, 20))</td>
<td>0.18</td>
<td>167.83</td>
<td>210</td>
</tr>
<tr>
<td>((10, 80))</td>
<td>0.10</td>
<td>314.10</td>
<td>260</td>
</tr>
<tr>
<td>((10, 100))</td>
<td>0.10</td>
<td>373.14</td>
<td>300</td>
</tr>
<tr>
<td>Wbest+</td>
<td>0.95</td>
<td>405.14</td>
<td>309</td>
</tr>
<tr>
<td>Spruce</td>
<td>19.16</td>
<td>766.24</td>
<td>734</td>
</tr>
<tr>
<td>PathChirp</td>
<td>10.36</td>
<td>305.21</td>
<td>222</td>
</tr>
<tr>
<td>iperf3 (throughput test)</td>
<td>10</td>
<td>11,851.90</td>
<td>8,940</td>
</tr>
</tbody>
</table>

implies that the link was largely saturated. The result shows AIWC can effectively reflect the impact of TCP traffic upon AB of the link.

**Cost Comparison**

By collecting measurement traces from the experiments above, we summarize the cost of each method in terms of time and traffic in Table 4.2. With different target range settings, we further break down the cost of AIWC into multiple rows corresponding to the settings for 802.11g/n/ac. From the perspective of time cost, AIWC shows up to 9× improvement compared to the second fastest method Wbest+ and over 100× speed up compared to the traditional methods of Spruce and PathChirp. For the data cost, AIWC costs less than 400 KB across all cases. Although PathChirp consumes the least traffic, the 10 seconds time duration makes it unfavorable. To measure higher ranges of AB, AIWC increases the traffic cost while reducing the time duration. As a reference, we also put the cost of a default `iperf3` test in the table.
Notably, the AT test shows significantly expensive cost with over 11 MB of data and 10 seconds of time duration.

4.4.2 Interference

Due to the unlicensed nature of WiFi, AB on WiFi is decided not only by the competing traffic on the same AP, but also by the interference on the same or overlapped channel. We continue by evaluating the performance of AIWC under interference. We set up another AP (interference AP) to run on the same channel as the major AP in order to generate interference. We also connected the competing client to the interference AP. By varying the traffic on the interference AP, we are able to vary the AB on the link between the major AP and the AIWC client. Similar to the case of cross traffic, we take the AB ground truth as the link capacity minus the throughput of interference traffic. The experiment was conducted on an 802.11n MIMO 2.4GHz link.
The experiments were run for all methods but we only plot the results of AIWC in Fig 4.15 as the other methods show no reactions to the changes of interference. In comparison with the cross traffic scenario, we also combine the AIWC result from Fig 4.11 into Fig 4.15. As shown in Fig 4.15, the AB drop resulting from interference traffic is more serious than from cross traffic. This occurs because interference traffic can 1) increase the backoff waiting due to CSMA/CA competition between two APs, 2) induce collisions incurring physical layer retransmissions. These effects lead to longer queuing time on AP thus lower AB on WiFi link. The result shows that AIWC can still distinguish the reduced AB resulted from interference.

Figure 4.16. $AI - AI_{base}$ of each sub-train w.r.t. rate limitation and link utilization.
4.4.3 Rate Limiting

With rate limiting in public WiFi (e.g., public guest WiFi), it is necessary to understand how AIWC would react to rate limiting. As most rate limiting is implemented by token bucket, we employed the `tc qdisc htb` tool to cap the maximum rate of outbound traffic of the AP. Under different settings of rate limits, we adopted the experiment setting of the cross traffic to vary the link utilization. The experiment was conducted under an 802.11ac link.

Fig 4.16 plots the $AI - AI_{base}$ for each sub-train in a (10, 100) Mb/s train. Without rate limiting, we can see the $AI - AI_{base}$ is consistently above zero, and the cross traffic can push this value higher by forcing more packets to be aggregated. With rate limiting, once a sub-train $i$ with probe rate $R_i$ is greater than the throttled rate, the $AI - AI_{base}$ drops negative, which implies the aggregated packets (according to the concatenating approach) were torn apart. For example, under a rate limit of 20Mb/s, the $AI - AI_{base}$ of the sub-trains whose probe rate is larger than 20Mb/s start to go negative. It is because the rate limitation prevents the high rate sub-trains from concatenating more packets to generate high probe rates. Therefore, the observed $AI$ stays low while the $AI_{base}$ increases with the probe rate. As the cross traffic and the probes share the tokens of rate limiting, the cross traffic can make the negative pattern occur on even lower rate sub-trains. This observation implies that $AIWC$ can recognize the rate limiting by detecting if the $AI - AI_{base}$ goes consistently negative.

4.5 Real-World Deployment

In order to evaluate real-world efficacy, AIWC was deployed on nearly one hundred mobile devices over a university break[^1]. Client applications were developed for both Android (Kit Kat+) and iOS (9+). Distribution was conducted via a closed beta in

[^1]: Full IRB approval and participant consent was received prior to deployment. Participants were incentivized for participation.
the Google Play Store\textsuperscript{[4]} and AirWatch\textsuperscript{[1]}.
In order to provide bandwidth reference for AIWC AB test, each app was configured to periodically fetch an object (approx. 7.2 MB) for a baseline TCP throughput (AT) followed by a AIWC test. Throughput objects were served via Apache web servers co-located in the same data center as the AIWC servers. The agents were user configurable to pick intervals between 2 to 8 minutes.

Table 4.3 summarizes the results of the high-level characteristics of the data. Notably, the data gathering recording more than 47,000 tests observing well over 6 million packet pairs. The tests came from a wide variety of network environment ranging across 992 different APs (unique BSSIDs). Generally, throughput tests tended to be successful with 91.3\% (43,263 out of 47,380) of the throughput tests completing successfully (45 second timeout). With a median throughput of 20.38 Mb/s for the throughput tests, it is not surprising that nearly 71.2\% of the throughput tests yielded an AT of more than 10 Mb/s.

For AIWC, 60.7\% of the AB tests yielded an AB of over 10 Mb/s, and among the rest 39.3\% of tests, 7.2\% falls into the red classification where $AB < 1$ Mb/s. Impressively, 50\% of AIWC tests are resolved within 123.2 ms of the first SYN packet from the client with a potentially actionable AB estimation. Notably, this is done without any geographic co-location as all AIWC tests had to traverse back to the same data center.

Finally, we close our data summary with a brief comparison versus achieving the same equivalent measurements using \texttt{iperf3}\textsuperscript{[2]}\textsuperscript{5}. For AIWC, the total cost of all AIWC AB estimations was 4.1 GB as one would not execute the ground truth AT measurement in an actual deployment. In contrast, the equivalent \texttt{iperf3} results would require 1,100 GB data, which is over 250 times more downlink bandwidth.

\footnote{We approximate the bandwidth cost for \texttt{iperf3} based on the observation that, \texttt{iperf3} costs $X$ Mb data for measuring a link with AT$\approx X$ Mb/s.}
under the assumption that `iperf3` actively saturates the link for a full 10 seconds for characterization (using our `AT` measurement as the ground truth).

4.5.1 AB/AT Analysis

Unlike the lab environment where network conditions can be mostly controlled, the large scale deployment affords no such luxury. To that end, we use the throughput tests as a stand-in for `AT` followed directly by the AIWC test instance to provide the ground truth reference. Client-side throughput results (or throughput failures) were linked to specific AIWC tests as a parameter in the HTTP GET request, which helps us connect the corresponding AB to AT result.

For the purpose of a more detailed analysis, we start with the precondition of a successful `AT` test for the results in Table 4.3. Table 4.4 further breaks down the `AT` tests into results of `AT` greater than 10 Mb/s and `AT` less than 10 Mb/s. The side-by-side comparison between AB and AT will give us a high-level view of the performance of AIWC AB test.

\[ AT \in (10, \infty] \]

As shown in Table 4.4, among those `AT` returning larger than 10 Mb/s, over two-thirds (67.5%) of those tests also have `AB` tests in the green zone (more than 10Mb/s). In such cases, the `AB` test could viably replace the `AT` test. Particularly, from the third column of average `AT`, we can clearly tell the `AT` decreases along with `AB`. It is noteworthy that, in 9.1% of `AB` test which resolved in the range of red zone ([0, 1] Mb/s), the `AT` unexpectedly shows high value at 20 Mb/s. After in-depth study on those tests, we find out that, many of those test have unacceptably low Packet Delivery Ratio (PDR). It implies that, when packet loss occurs, our method intends to give conservative estimation. If one requires a PDR of even 0.9 (10% loss), the number drops drastically to 3.5% implying a reasonable mapping between
estimation quality and loss rate. Fortunately, most (82%) of our tests meets the PDR threshold of 0.9, which infers packet loss does not have detrimental impact on our technique.

\[ \text{AT} \in [0, 10] \]

In the lower portion of Table 4.4, the constrained case of AT less than 10 Mb/s is broken down. Under this low range of AT, majority of AB results is moved to the yellow and red zone where AB < 10 Mb/s. However, we find that AB actually classifies 23.9% of cases as green despite a clearly non-green AT measurement. In practice, such a case represents an instance where the WiFi network was rate limited as mentioned in Section 1.3.1. The lightweight nature of AIWC does not ‘trip’ the shaping mechanisms while the AT test is actively shaped. Hence, the link appears to be largely uncongested with ample available bandwidth but yet the reality is that said link is actively limited. Absent the acquisition of a much larger bandwidth footprint, AIWC would be incapable of detecting such rate limiting links. Occasional AT tests coupled with crowdsourcing could detect such cases but such a topic is beyond the scope of the paper. Generally, the AIWC AB test tends to be reasonably pessimistic as espoused earlier in the paper as one would expect when AT is less than 10 Mb/s.

Resolution Time Analysis

For the cases where AB is greater than zero and a reasonable PDR is observed providing appropriate confidence, the completion of the AB estimation and discernment of uplink bottlenecks represents when a viable, actionable value will be available at the server. Figure 4.17 plots the eCDF of the AB resolution time broken down into various RTT ranges. For cases of an RTT up to 60 milliseconds, the vast majority of resolutions are completed before 225 ms. Even for larger AB values (various mobile devices in the results were international), nearly 70% of the results are discerned.
before 250 ms.

4.6 Conclusion

In this work, we presented AIWC (Aggregation Intensity based WiFi Characterization) to achieve rapid and efficient available bandwidth estimation for the modern WiFi. By understanding and further exploiting the rich information embedded in frame aggregation, we proposed a novel approach to utilize the aggregation intensity as an effective link congestion indicator. Together with the probe packet train technique, the designed AB estimation approach conquered the problem of estimating AB in the presence of frame aggregation which fails all of the previous AB methods. With the proof of concept system, we demonstrated that AIWC can achieve high accuracy and robustness across various network scenarios.
<table>
<thead>
<tr>
<th>Overall</th>
<th>Period</th>
<th>9 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Instances</td>
<td>47,380</td>
<td></td>
</tr>
<tr>
<td>Packet Pairs</td>
<td>6,168,682</td>
<td></td>
</tr>
<tr>
<td>Unique Users</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Unique SSIDs</td>
<td>154</td>
<td></td>
</tr>
<tr>
<td>Unique APs</td>
<td>992</td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>43,236</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AT</th>
<th>(10, ∞) Mb/s</th>
<th>71.2%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[0, 10] Mb/s</td>
<td>28.8%</td>
</tr>
<tr>
<td>Estimated Throughput</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50th Percentile</td>
<td>20.38 Mb/s</td>
<td></td>
</tr>
<tr>
<td>75th Percentile</td>
<td>41.17 Mb/s</td>
<td></td>
</tr>
<tr>
<td>90th Percentile</td>
<td>57.89 Mb/s</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AB</th>
<th>(10, ∞) Mb/s</th>
<th>60.7%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[0, 10] Mb/s</td>
<td>39.3%</td>
</tr>
<tr>
<td>Resolution Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50th Percentile</td>
<td>123.2 ms</td>
<td></td>
</tr>
<tr>
<td>75th Percentile</td>
<td>172.5 ms</td>
<td></td>
</tr>
<tr>
<td>90th Percentile</td>
<td>239.8 ms</td>
<td></td>
</tr>
</tbody>
</table>
TABLE 4.4

RESULT ANALYSIS BASED ON AT CLASSIFICATION.

<table>
<thead>
<tr>
<th>AT ranges (Mb/s)</th>
<th>AB ranges (Mb/s)</th>
<th>% of Tests</th>
<th>$\overline{AT}$ (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10, $\infty$]</td>
<td>(10, $\infty$]</td>
<td>67.5</td>
<td>40.8</td>
</tr>
<tr>
<td>(7, 10]</td>
<td>(7, 10]</td>
<td>8.6</td>
<td>34.3</td>
</tr>
<tr>
<td>(4, 7]</td>
<td>(4, 7]</td>
<td>7.9</td>
<td>27.4</td>
</tr>
<tr>
<td>(1, 4]</td>
<td>(1, 4]</td>
<td>7.0</td>
<td>23.6</td>
</tr>
<tr>
<td>[0, 1]</td>
<td>[0, 1]</td>
<td>9.1</td>
<td>20.4</td>
</tr>
<tr>
<td>[0, 10]</td>
<td>(10, $\infty$]</td>
<td>23.9</td>
<td>5.3</td>
</tr>
<tr>
<td>(7, 10]</td>
<td>(7, 10]</td>
<td>8.4</td>
<td>5.0</td>
</tr>
<tr>
<td>(4, 7]</td>
<td>(4, 7]</td>
<td>14.8</td>
<td>4.6</td>
</tr>
<tr>
<td>(1, 4]</td>
<td>(1, 4]</td>
<td>28.3</td>
<td>3.5</td>
</tr>
<tr>
<td>[0, 1]</td>
<td>[0, 1]</td>
<td>24.6</td>
<td>4.3</td>
</tr>
</tbody>
</table>
CHAPTER 5

FMNC ON THE LTE NETWORK

As a sequel of the work presented in the previous chapter (Chapter 4), this chapter introduces a novel available bandwidth estimation solution on modern cellular networks, i.e., LTE networks. This work adopts the similar design philosophy as the previous work that leverages the lower layer transmission behavior to reveal network bandwidth condition. Specifically, we find the TBS consistency in the physical layer can serve as an effective network congestion indicator. Based on this intuition, we design a probe train with logarithmic-like increasing probe rate. Furthermore, to estimate the result from packet measurements, a Bayesian change point detection algorithm is designed to cope with the uncertainty resulted from the noisy measurements. The designed system is carefully evaluated under various real-world network scenarios.

5.1 Background and Motivation

In this section, we start with the basics of data transmission on the LTE network, particularly on the downlink. Then we will demonstrate why the existing available bandwidth solutions mix poorly with LTE networks.

5.1.1 Challenges of Estimating Available Bandwidth on LTE

LTE is designed to be highly flexible for radio resource allocation among multiple end users. The LTE downlink exploits OFDMA (Orthogonal Frequency Division
Multiple Access) to enable fine-grained channel resource allocation in the time and frequency domains. As shown in Fig 5.1, the channel resource is divided into a grid structure. The frequency domain (vertical) is divided into 180-kHz sub-channels. The time domain (horizontal) is divided into continuous 0.5 ms time slots. Thus, a minimal resource unit (i.e., one cell in the grid) is called Physical Resource Block (PRB). For every TTI (Transmission Time Interval, 1 ms), the base station (eNodeB) will make a scheduling decision to assign PRBs to different UEs. As Fig 5.1 shows, different UEs can obtain different shares of the channel over time.

When assigning PRBs, the eNodeB will also decide the proper Modulation and Coding Scheme (MCS) for transmitting the data. Combing PRB and MCS, the total size of data that can be transmitted to a UE during a TTI is defined as the Transport Block Size (TBS). From the perspective of upper layers (e.g., the transport layer), since a TBS (up to 9 KB) is usually greater than the size of a packet (e.g.,

---

1Note that the scheduling policy of how to assign PRB and MCS is vendor-specific that can adopt different principles [17].
≤ 1500 B), the packets transmitted often are done so in an aggregated manner. The result is that TBS-based transmission can make packet delivery on LTE behave in a batch or bursty manner.

**Experimental Study of AB Estimation on LTE**

We now show how this batched behavior can harm traditional AB estimation methods. In this experiment, we ran *PathChirp* \(^{[91]}\) over LTE on a phone (Huawei Nexus 6P). *PathChirp* sends packets with exponentially decreasing packet gap, and estimates AB by searching for a turning point where the packet queuing delay (i.e., packet sending gap minus the receiving gap) starts to rise. For the comparison case, we set up a local wired network with a wired link capacity of 70 Mb/s (as the typical capacity of a 10 MHz carrier with 2x2 MIMO under ideal RF conditions). To emulate traffic load, we first measured the throughput on LTE via *iperf3* (i.e., 40 Mb/s). Next, we created similar throughput conditions (AT = 40 Mb/s) on the
wired network by generating 30 Mb/s of cross traffic (using `iperf3`) from another wired client. In this case, we know the ground truth of AB should be less than or close to AT, according to their definitions.

In Fig 5.2, we plot the packet queuing delay in a packet sequence as output from `PathChirp`. The test probes vary from 10 Mb/s to 200 Mb/s. The rectangles indicate the packets that arrived at the same time. We start by looking at the Ethernet network. As Fig 5.2a shows, the queuing delays gradually increase when the probe rate reaches the available bandwidth. The estimated available bandwidth is around 30Mb/s which is close to AT (40 Mb/s). In contrast, in Fig 5.2b, the queuing delay pattern on LTE is dramatically distorted and shows bursty patterns. As a result, the estimated results are erroneously high (75 Mb/s) which is considerably above AT.

**Implication**

The aggregation of LTE breaks the expected correlation between the network conditions and probe traffic observations (e.g., consistently increasing delay once capacity is exceeded). While curve fitting as conducted in [77, 83, 84] can overcome this issue, we argue such an approach is inelegant at best and fundamentally broken at its worst. It is the exact the need for revisiting this relationship and designing a new AB estimation solution for LTE that motivates this paper.

5.2 System Design

In this section, we introduce the design of our proposed Available Bandwidth solution on LTE networks. The designed solution adopts the traditional methodology of probe packet-based measurement. However, in contrast to existing methods, we devise a new physical layer metric—`TBS Fluctuation`—to capture network congestion conditions when induced through probe packet observations. The designed metric serves as an effective network congestion indicator (Section 5.2.1). Based on this
metric, we design new probe packet train pattern to cope with the aggregation effect of LTE (Section 5.2.2). The proposed design is implemented on an HTTP-based packet probing system which requires no modification on the client side (Section 5.2.3). Particularly, we discuss how we can leverage the packet observations to infer the designed network metric. Along the discussions in this section, we conduct real-world experiments to validate the various design principles.

5.2.1 TBS Fluctuation as the Network Congestion Indicator

Conceptually, the goal of estimating available bandwidth (AB) is to measure the portion of bandwidth that a UE can obtain without inducing congestion to existing traffic. Thus the key problem of measuring (AB) can be reduced to following: Can a UE detect if he/she is experiencing congestion? If the question can be solved, the task of AB estimation can be reduced to searching for the maximum traffic rate for a UE that does not cause congestion. To discern congestion, we design a network congestion indicator—TBS Fluctuation. As we know that the TBS is adaptively scheduled for UEs according to their channel quality and traffic competition and that channel quality for a UE is likely to be relatively consistent within a small time window (e.g., 1 second), variations of TBS should be highly correlated with the traffic competition and congestion.

Namely, the TBS received on the UE is supposed to be relatively consistent when the bandwidth resource is ample and requires no or light competition. Otherwise, if a UE is experiencing serious bandwidth competition with other UEs (e.g., network congestion), the TBS assigned is likely to be variant. This is because when the eNodeB tries to maintain a proportional fair schedule among UEs, it will inevitably

---

2 As observed in our real-world experiment, the MCS (which is proportional to CQI–Channel Quality Index) is largely consistent over 1-2 seconds.

3 According to the survey, a practical scheduler should guarantees fair throughput distribution among users.
prioritize the need of different UEs over TTIs. As the consequence, a UE can receive inconsistent TBS assignment over time. Technically, by observing the variation of TBS for a UE over time (i.e., TTI), we are able to detect whether or not a UE is experiencing congestion. Therefore, we define \(TBS\) Fluctuation \(TF\) to describe the degree of TBS variation. For a series of TBS assigned to a UE \((TBS_i, i = 1, ..., N)\), we have

\[
TF_i = |TBS_{i+1} - TBS_i|, i = 1, ..., N - 1
\]  

(5.1)

Experimental Validation

To demonstrate the how \(TBS\) Fluctuation can discern congestion (traffic competition), we employed two LTE phones (Huawei Nexus 6P and Google Pixel running Android 7.1.1) operating on a US cellular network with both phones connected to the same cell. We used one phone as a test UE to generate CBR traffic on the downlink. The second phone caused cross traffic via iperf3. We first sent 200 packets as CBR with a 5 Mb/s traffic rate to the test UE without competing traffic. Then, we launched aggressive iperf3 traffic on the competing UE and repeated the same CBR traffic on the test UE. The PHY layer trace was collected from the test UE by using MobileInsight [60].

In Fig 5.3, we plot the PHY trace collected both without (Fig 5.3a) and with (Fig 5.3b) competing traffic. Since the two runs were conducted within several seconds of each other, the bandwidth condition on cellular were relatively consistent over this period of time [118]. The observed difference should be largely attributable to the generated competing traffic. As shown in the figure, the MCS under both cases looks similar as the channel quality was relatively consistent. When there was no competing traffic (left plots), the patterns of PRB and TBS show as smooth horizontal lines with few spikes. However, when in the presence of competition (right plots), the PRB and TBS patterns fluctuate with a large number of spikes. The different
observations on TBS clearly show promise for revealing potential traffic competition.

**Analytic Evaluation**

To further evaluate *TBS Fluctuation*, we extended the previous experiment to vary the send rate of the CBR traffic on the test UE from 1 Mb/s to 30 Mb/s. Following the similar setting above, for each send rate, we sent over ten 200-packet sequences under the absence and presence of the *iperf3* competing traffic. From the captured PHY layer trace, we calculate the *TBS Fluctuation* over all TBS and take the aggregated result of each 200-packet sequence. In Fig. 5.4, we plot the comparison...
of the measured $TBS\ Fluctuation$ with absence and presence of competing traffic. The points are the mean and the bars are the standard deviation. We see that the $TBS\ Fluctuation$ gradually increased when the send rate rose. As expected, the $iperf3$ competing traffic caused a higher $TBS\ Fluctuation$ compared to the $w/o\ iperf3$ case. It is noteworthy that the $TBS\ Fluctuation$ stays quite small in low rate range (i.e., 1-5Mb/s) and then quickly hikes up after 7Mb/s. Notably, this potentially reveals that the available bandwidth under that experiment environment was roughly 7 Mb/s. We arrive at this conclusion as the traffic merely experienced marginal competition when the send rate was less than 7Mb/s while the competition indicated by $TBS\ Fluctuation$ became severe after 7Mb/s. This observation informs our self-induced congestion probe packet approach which will be discussed in the following subsection.

5.2.2 Probe Train Design

Following the intuition of leveraging the $TBS\ Fluctuation$ as a congestion indicator, we adopt the concept of self-induced congestion to form the probe packet sequence
(a.k.a train). The general idea is to send a train with monotonically increasing probe rate to search for a “tipping point” where the network condition changes from uncongested to congested. Thus the available bandwidth can be approximated with the probe rate at the “tipping point”. To minimize the probe traffic cost, we design the probe train as a “one-shot” test that requires only injecting one packet train to finish a test, rather than doing iterative packet streams such as PathChirp [91] and IGI [105] do.

To generate different probe rates, we vary the packet gaps and fix packet sizes at the MTU (Maximum Transmission Unit) as different packet size may experience significantly different delays on cellular [42]. For our baseline, we employ a linear pattern to our probe rate. However, given the batched transmission of LTE, when the probe rate is high, a large number of packets may be aggregated together, thus generating only a few timing measurements at the receiver. To cope with this aggregation effect, we want the probe rate to grow more slowly at higher probe rates. This means that we should assign more packets to the higher rates to increase the number of measurements. Following this intuition, the probe rate can be designed as a two-stage model that 1) starts with linear increases and then 2) from a certain point, changes to grow in a logarithmic-like function.

Technically, the probe packet train is designed to search available bandwidth from zero to a pre-define maximum rate $R_{max}$. We denote the probe rate for the $i$-th packet in the train as $R_i$ ($i = 1, 2, ..., L$, $L$ is total length of the probe packet train). Then the packet gap between the $i$-th and $i+1$-th packet can be computed as $G_i = \frac{P}{R_i}$, where $P$ is the packet size which is set to $MTU$. Assuming the probe rate

\[ G_i = \frac{P}{R_i} \]

4Since it is impractical to generate a zero probe rate, we define the lowest probe rate as a small value (i.e., 250Kb/s).
increasing step is $\Delta_i = R_{i+1} - R_i$, we have

$$\Delta_i = \delta \cdot \min \left( \frac{G_i}{G_c}, 1 \right)$$

(5.2)

We define a cut-off gap $G_c$ as the point where the probe rate switches from linearly increasing to logarithmic-like. When the probe rate is small such that the packet gap is greater than this value ($\frac{G_i}{G_c} > 1$), the probe rate grows linearly with the slope $\delta$. When the probe rate reaches a certain value that makes packet gap less than the cut-off gap ($\frac{G_i}{G_c} < 1$), the increasing step $\Delta$ proportionally decreases along with the packet gap $\delta \cdot \frac{G_i}{G_c}$. By doing this, we ensure that the increasing step becomes small when the probe rate increases. In addition, this design forces the probe rate increases by $\delta$ for each time duration of $G_c$. Overall, once $G_c$, $\mathcal{L}$ and $R_{max}$ are set, the $\delta$ can be computed by using a search algorithm (e.g., binary search).

Relief Gaps

For LTE, short-term channel fading (e.g., fast fading due to movement) may result in transient congestion effects. This short-term congestion can confound the actual congestion that results from exceeding the actual Available Bandwidth. While the congestion caused by transient channel fading is often resolved fairly quickly, congestion effects due to exceeding the Available Bandwidth should persist as long as bandwidth is saturated. In order to mitigate the impact of short-term congestion, we intentionally insert fixed large gaps (pauses) into our probe packet train at a fixed packet interval that we dub *relief gaps*. These relief gaps alleviate short-term congestion and prevent early transient issues from distorting later higher rates in the probe packet train.

Although there is a valid concern that adding relief gaps can potentially slow overall probe rates, the designed traffic indicator—*TBS Fluctuation*—captures the
variation of TBS, not the average of TBS. As long as the probe traffic creates enough queuing pressure on eNodeB before inserting a gap, the TBS Fluctuation will not fundamentally change while the mean of TBS may drop by adding the gaps. The setting of relief gap can be defined by $N$ and $G_r$. Namely, we insert a gap of $G_r$ for every $N$ packets.

**TABLE 5.1**

<table>
<thead>
<tr>
<th>PACKET TRAIN FORMAT PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-Defined</td>
</tr>
<tr>
<td>$G_c$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$G_r$</td>
</tr>
<tr>
<td>Custom-Setting</td>
</tr>
<tr>
<td>$L$</td>
</tr>
<tr>
<td>$R_{max}$</td>
</tr>
</tbody>
</table>

** Packet Train Parameters**

Overall, a packet train can be described with the following parameters: $G_c$, $L$, $R_{max}$, $N$ and $G_r$. We divide the parameters into two groups: *system-defined* and *custom-setting* as shown in Table 5.1. The *system-defined* parameters are set to yield optimal system-wide performance regardless of bandwidth condition, including $G_c$, $N$ and $G_r$. The *custom-setting* parameters are adjustable based on user’s needs, including $L$ and $R_{max}$. In the next subsection, when introducing the implementation
system, we will discuss the settings of the system-defined parameters that work best for the system. The optimal setting of the custom-setting parameters will be explored in the later section of experimental evaluation (Section 5.4).

5.2.3 FMNC-Based Estimation

To obtain the traffic information at the client side, we further propose a TCP Timestamp-based solution to infer the TBS statistics of the client.

**Inferring TBS using TCP Timestamp**

We utilize the *TSval* carried in the TCP timestamp option to infer packet arrival timing at the client. As the timestamp option is enabled by default for almost all mobile phones (Android and iOS devices), timestamp-based measurement works have become quite popular [57, 58]. The modern mobile devices can provide at least 3.3 ms/tick resolution\(^5\) which reveals the packet arrival timing information at a fair granularity. As TBS describes the overall data size delivered in a TTI (1 ms), the packets in a TBS usually appear to arrive at the same time (recall the rectangles indicated in Fig 5.2). Intuitively, we can infer the TBS with the total size of the packets whose returned TCP ACKs have the same *TSval*. Since the time span of a *TSval* is not necessarily equal to TTI, the calculated result needs to be further divided by the timestamp resolution (e.g., 3.3 ms/tick for Android phones) to compute the average transmitted data size per TTI.

To estimate the timestamp resolution of a client, we exploit the first pair of packets in the probe train to measure the value. By assigning a constant large time gap\(^6\) (i.e., 40 ms) between the first two packets, we the receiving gap is largely equal

---

\(^5\) Android (after Android 6.0.0) and iOS devices have 3.3 ms/tick and 1 ms/tick resolution respectively.

\(^6\) The large gap forces the *TSval* to increase more than one.
to the sending gap due to the low data rate. Thus the resolution can be computed from the difference of the TSval values of the first two TCP ACKs.

We conducted real-world experiments to evaluate the TBS inference accuracy by using the timestamp. Due to space constraints, we do not show the detailed results. Overall, the inferred value tends to overestimate especially when the actual TBS is high as the coarse resolution of the timestamp can amplify the impact of aggregation. To overcome this issue, we apply a moving minimum filter (of window size 2) over the inferred series. The filtered result significantly improves the inference accuracy which helps control the estimation error within 2 Kb for all tested TBS values (ranging from 1 Kb to 30 Kb).

 Probe Packet Train System-Defined Parameter Setting

We set $G_c$ to $3.33$ ms as the maximum timestamp resolution for most smartphones. The setting guarantees that we can obtain at least one unique TSVal value for every time probe rate increases by $\delta$. For relief gap setting, we set the gap width $G_r$ to $20$ ms (which yields merely 500Kbps data rate). Under modern LTE networks with typical a downlink rate greater than 10Mbps [42], this time duration is sufficient to empty a user’s buffer at the eNodeB. For $N$, we have to ensure that the amount of probe traffic is sufficient to generate at least one TBS inference value. It means the size of the probe traffic before each relief gap should be at least $\text{max}(\text{TBS}) \times \text{max}(\text{Timestamp Resolution})$. Given the MTU as the packet size, we set $N = 20$ ($\approx \text{max}(\text{TBS}) \times \text{max}(\text{Timestamp Resolution})/\text{MTU}$).

5.3 A Bayesian Change Point-Based Estimation Algorithm

Given a series of TBS observations yielded from a probe packet train, in this section, we introduce the estimation algorithm to calculate the available bandwidth. Recalling the intuition of our probe packet train, as the probe rate increases along the
train, we leverage the TBS observations to capture the network condition changing from uncongested to congested. The available bandwidth can then be calculated by pinning down that particular change point. However, due to the high dynamics of the wireless channel on LTE, the TBS observations can be so noisy that a naive threshold-based approach is inadequate to pick a right change point. To cope with the uncertainty and noise involved in the observations, we design a probabilistic Bayesian change point detection algorithm similar to [115].

5.3.1 Data Preprocessing

Given a sequence of TBS fluctuations $TF$, we first transfer it into a binary expression with a threshold value $TH_{\theta}$. We denote the $i$-th TBS fluctuation as $TF_i$. Its binary expression $BTF_i$ will be $0$ (uncongested) if $TF_i < TF_{\theta}$ (congested) and 1 otherwise. To set a proper cutoff $TF_{\theta}$, we check the distributions of $TF$ under uncongested versus congested networks. Based on the empirical study (e.g., Fig 5.3), we find the $TF$ is usually below $3$ $Kb$ under uncongested networks and rises above $3$ $Kb$ when network congestion is experienced. Thus, we set the cutoff $TH_{\theta}$ to $3$ $Kb$.

5.3.2 Model Design

Once we transfer the sequence into binary expression, we can attempt to find a change point $\tau$ that divides the sequence of $\{BTF_i\} \ (i \in [1,N])$ into two segments: the first segment includes largely zeros and the second segment contains mostly ones. To formulate this problem, we assume $BTF$ follows the Bernoulli distribution with parameter $p$. Particularly, $p$ is a two-stage value that it takes $p_1$ in the first segment and $p_2$ in the second segment respectively. The change point is then located at $\tau$
(1 ≤ τ ≤ N). Overall, the BTF can be expressed as:

\[
BTF \sim \text{Bernoulli}(p), \text{where } p = \begin{cases} 
    p_1 & : 0 \leq i \leq \tau \\
    p_2 & : \tau < i \leq N
\end{cases}
\] (5.3)

The distribution of \( p_1 \) and \( p_2 \) can be generated by the conjugated prior distribution Beta with \( a_1, b_1 \) and \( a_2, b_2 \) as the hyperparameters.

\[
p_1 \sim \text{Beta}(a_1, b_1), \quad p_2 \sim \text{Beta}(a_2, b_2)
\] (5.4)

By assigning the proper hyperparameters, we can apply the prior knowledge of \( p_1 < p_2 \) to imply that the first segment usually contains less ones.

The goal of the model is to find a proper \( \tau \) to maximize the joint probability to get the maximum a posterior estimation:

\[
\hat{\tau} = \arg\max_\tau \int \int p(BTF, \tau, p_1, p_2)dp_1dp_2
\] (5.5)

5.3.3 Estimation \( \hat{\tau} \) via Gibbs Sampling

In order to solve this problem, we adopt the Gibbs sampling to estimate the latent parameters. One of the merits of using sampling method is that it reveals the complete posterior distributions of all the latent parameters \( p_1, p_2 \) and \( \hat{\tau} \). These distributions can help assess the quality of estimation (as will be discussed soon). Overall, the estimation process can be illustrated in Fig. 5.5. Note that in the second row when we transfer the TF to BTF, we apply a low-pass filter (LPF) upon the original data to help rule out some sharp spikes. In the figure, we show the two cases of applying the proposed estimation method upon the PHY layer ground truth TBS (Fig. 5.5a) versus the inferred TBS from TCP timestamp (Fig. 5.5b). We see that both
Figure 5.5. Step by step estimation algorithm illustration on real data. (a) and (b) are from the same test.
cases yield similar estimations with the inference approach slightly underestimating.

In the experiment part (Section 5.4), we will further evaluate the performance of using inference compared with the ground truth measurement.

5.3.4 Interpreting Estimation Result

Once sampling process converges, we can look into the estimated parameters to tease out the available bandwidth. Ideally, the probe rate at the change point $\hat{\tau}$ indicates the available bandwidth. However, we need to further assess the quality/confidence of such an estimation by checking $p_1$ and $p_2$. We define a congestion threshold $p_\theta$ to assess if a segment (i.e., the first or the second) experienced network congestion. We will explore the setting of $p_\theta$ in the next section (Section 5.4). Based on the different values of $p_1$ and $p_2$ compared with $p_\theta$, we have the following cases:

- $p_1 < p_\theta < p_2$: This is the desired case that the available bandwidth is indicated by $\hat{\tau}$.
- $p_1 > p_\theta$: It means the change point is selected in a position that even the first segment involves a relatively high degree of congestion. So we discard the result as it portends overestimation.
- $p_1 < p_\theta$ and $p_2 < p_\theta$: This means the entire probe packet train does not trigger network congestion. Thus the available bandwidth are above the maximum probe rate $R_{max}$.

Compounding Loss

To integrate another important aspect of performance impact factor – loss, we exploit the probe traffic loss to incorporate the loss impact on the estimation. Intuitively, given the estimation value returned from the change point algorithm, we proportionally discount the result to the probe traffic delivery rate. Therefore, the final AB estimation can be expressed as $\text{change\_point\_result} \times \frac{\# \text{ of received}}{\# \text{ of sent}}$. 

88
5.4 Performance Evaluation

In this section, we conduct extensive real-world experiments to evaluate the performance of the proposed method under various network conditions. Generally, the evaluation requires the knowledge of the ground truth available bandwidth (AB) on the cellular network. However, as we can not control the network setting, it is impractical to obtain this information. As a workaround, in this paper, we leverage the *Achievable Throughput* (AT) as a reference and evaluate AB by analyzing the relationship (e.g., difference, correlation) between AT and AB.

We start this section with exploring a valid TCP variant as the reference measurement. By using the valid reference, we then verify the effectiveness of the AB estimation (Section 5.4.1). Next, we investigate the impact of different parameter settings (Section 5.4.2). Followed by Section 5.4.3, we conduct a longitudinal test run to examine the correlation between AT and AB. At last, by doing a in-town driving test, we test the proposed method under multiple network cells. We also study the performance under different movement profiles.

5.4.1 Effectiveness Validation

Since the AT measurement can vary with different congestion control algorithms (i.e., TCP variants), we need to find a variant which works best to reveal bandwidth variations under the modern cellular network environment. In this case, we tested three popular TCP variants as the potential AT reference choices, including *Vegas*, *Westwood* and *Cubic*. In order to validate the effectiveness of different AT measurements as well as our AB estimation method, we intentionally caused bandwidth variation by putting the test phone in and out of a metal box. The phone got -97 dB (53%) and -107 dB (30%) RSRP when in and out of the box. We ran a TCP throughput test (by downloading a 5MB file) followed by an AB test every 10 seconds. For each TCP variant, the experiment lasted for 20 minutes with 10 minutes
out of the box and 10 minutes in the box. The AB probe train parameters were set as $L = 400, R_{\text{max}} = 30 \text{Mbps}$ and $p_\theta = 0.2$. We will further study the impact of parameter setting next subsection. Again, we used Mobile Insight to capture the PHY layer trace for every single AB test.

**Result and Analysis**

We first study the relationship between measured AB versus different TCP throughput. In Fig 5.6, we compare the AB estimation to various TCP variants measurement. In Fig 5.6a, we put the TCP throughput in the $x$-axis and the estimated AB from mobile insight PHY trace in the $y$-axis. Each point in the figure is the mean of the two tests. We see that the throughput measured from Westwood and Vegas are consistently low. This is because the two algorithms utilize delay observation to decide the throughput (i.e., $cwnd$), which can be ill-suited under dynamic LTE network.
environment. So the yield throughput was constantly suppressed and barely reacted to the bandwidth variation when the phone was in and out of the box. Fortunately, Cubic, as the default setting on Linux kernel, shows evident variation on the throughput. The overall Cubic throughput is greater than our AB test, which matches our expectation according to the definitions of AT versus AB. Particularly, in Fig 5.6a, we see the Cubic measurements scattered along the diagonal direction. This implies a good correlation between AB and Cubic test. To further reveal the variation of Cubic, we plot the time series in Fig 5.7 where we clearly see that the throughput starts to drop after the phone was put into the box. Combining both Fig 5.6 and Fig 5.7, we finalize Cubic throughput as the bandwidth reference. In later experiment, the AT result refers to the throughput measured from TCP Cubic.

On the other hand, in Fig 5.6b, we also compare the AB from PHY trace (y-axis) versus from TCP timestamp (x-axis). The result shows clustered dots around the dashed line (which indicates the ideal case). It means the TCP timestamp inference approach can yield similar estimation as the PHY trace provides. In the next part, we will further give the quantitative analysis regarding the accuracy of the timestamp-
Figure 5.8. The estimation difference distributions under different $L$.

based approach compared to the PHY trace-based method.

5.4.2 Parameter Setting

We start with exploring proper setting for the custom-setting parameters $R_{\text{max}}$ and $L$ (Section 5.1) in our probe packet train. Generally, a user can use $R_{\text{max}}$ to decide an upper bound she/he desires to probe. To better leverage the change point detection algorithm, we intend to create a probe train which can trigger network congestion for sure. Based on our empirical observations, the TBS often shows significant fluctuation when the probe rate reaches beyond 30Mb/s. So that we take $R_{\text{max}} = 30$ Mb/s for rest of the result.

$L$ decides the data budget a user wants to spend on running the test. For a certain $R_{\text{max}}$, the setting of $L$ is crucial for determining estimation accuracy. This parameter can decide how fast the probe rate increases (i.e., $\delta$ in Equation 5.2). To study its impact, we fixed $R_{\text{max}}$ (30Mb/s) and repeated the previous 20-min AB and
AT test for three different $L$ settings (i.e., 200, 400 and 600). In Fig 5.8, we plot the estimation difference distributions under the different $L$. From Fig 5.8a, we see that when $L = 200$ there is about 40% the AB tests greater than AT–overestimation. It is because due to the large buffer size on eNodeB, once the probe rate exceeds the available bandwidth, it requires additional traffic to reveal the congestion observation. When $L$ is small and probe rate increases quickly, this additional traffic can cause the estimated result to deviate far above the actual available bandwidth. When $L$ rises to 600, the overestimation reduces to 20%.

Fig 5.8b shows a small $L$ can even cause timestamp-based inference overestimates over the truth estimation from the PHY layer (see $L = 200$). It is due to that when the packet train is short, the sparse timestamp measurement is less sensitive to capture the actual TBS variations thus requires more dramatic changes on the PHY layer to trigger network congestion observation. Once we increase the packet number to 400, such effect can be largely mitigated. For the sake of both data cost and estimation accuracy, given $R_{max} = 30$ Mb/s, we set $L$ to 400.
We now continue to study the impact of \( p_\theta \). Given the estimated result from the proposed algorithm, the results returned with \( p_1 > p_\theta \) need to be discarded as they indicate overestimation. A large \( p_\theta \) may fail to capture the overestimations. While setting \( p_\theta \) too small could exceedingly discard too many estimations (including good results). By using the results collected from the experiments above, Fig 5.9 plots estimation difference distributions of the bad tests captured with different \( p_\theta \). The numbers inside the parentheses indicate the percentage of discarded results. The goal of setting \( p_\theta \) is to filter out the excessive overestimations. From the figure, we can see that when \( p_\theta = 0.1 \) we discard 36.54% tests (as indicated in the parentheses) and half of them even not overestimate. It is clearly an improperly low setting. However, if we set \( p_\theta = 0.3 \), we only discard a small amount (2.56%) of dramatically overestimated tests while leaves many other overestimations uncaptured. To balance the tradeoff between over filtering and discerning overestimation, we choose a middle point as \( p_\theta = 0.2 \).

5.4.3 Longitudinal Evaluation

In order to evaluate how the proposed method can capture the bandwidth variation in a long-term run, we conducted longitudinal overnight experiment. During this experiment, we periodically ran the AT test followed by two AB tests. Thus the throughput test serves as the reference for the following two AB tests. The experiment lasted for 12 hours starting from 9:00 PM to 9:00 AM. In total, we collected 702 tests with 88.19% of them valid.

In Fig 5.10, we plot the time series of the estimated results. The dots are the raw measurements and the lines are smoothed result with 20-min moving average window. We see that the curves of AT and AB show great match in the general tendency. To further analyze the degree of the matching, we calculate their Pearson correlation coefficient in a 5-hour window. We slide this window over time and plot the empirical
Figure 5.10. Time-series plot of AT and AB estimation from the longitudinal run.

Figure 5.11. Pearson correlation coefficient over the smoothed result.
CDF in Fig 5.11. Particularly, we plot both results of with and without compounding probe loss (as discussed in Sec 5.3). As shown in the figure, when considering loss into the AB estimation, it can achieve 0.7 correlation coefficient as the median. While without involving loss, the median value of the calculated coefficient is only 0.03. It implies that probe loss is one of the key points to reconcile the AB and AT estimation.

Another aspect to look at the estimation performance is the absolute estimation error between AT and AB. As shown in Fig 5.12a, about 80% of AB tests are less than the corresponding AT test, which matches our expectation as the AT test always tends to be more aggressive than AB test. Overall, there are about 75% of AB tests stay within $\pm 2Mb/s$ of AT tests (the shadowed area). To show how the difference distribute across different bandwidth condition, in Fig 5.12b, we take the AT estimation as the $x$-axis and plot the corresponding estimation difference (AB - AT) on $y$-axis. As the figure shown, most of the overestimation from AB tests occurs when AT is very low (e.g., 2Mb/s). This overestimation is mostly due to the different
Figure 5.13. Test time cost analysis.

sensitivities of the two kinds of tests reacting to the loss. As TCP (especially the Cubic) is utterly sensitive to loss, any noticeable loss can acutely dampen the AT test result. In addition, the AT test takes long time to finish which is more likely to encounter loss. It potentially exacerbates the impact of loss upon AT test. In contrast, our AB estimation merely takes linearly reduction with the loss rate. Noted that when the AT test grows out the low region, the AB estimation gradually falls under AT test as the impact of loss get diluted.

Test Cost Analysis

To examine the test cost with different metrics, we compute the time duration of the two tests by using the experiment results collected above. In terms of data consumption, the traffic volume used in AT and AB tests in our experiment are static: an AT test costs 5MB and an AB test costs 540KB ($\mathcal{L} = 400$). When using the popular AT test tools (e.g., iperf3), the traffic cost can be much high
and proportional to the bandwidth, e.g., when it costs $X$ MB to measure a network path with $X$-Mbps throughput. For time duration, the test time can vary with the bandwidth condition, especially for AT test. In Fig 5.13a, we plot the distribution of time cost of both types of test. We can see that our proposed AB estimation solution merely takes less than 1 second to finish (with the median of 800 ms). However, the AT test costs at least several seconds and up to tens of seconds to finish. As shown in Fig 5.13b, the time duration of AT test proportionally grows with the decreasing of bandwidth. In contrast, the time duration of the proposed AB test is constant. Benefited from being lightweight and non-intrusive, the AB test is a great choice to do longitudinal network monitoring.

5.4.4 In-Town Drive Evaluation

To evaluate the proposed method under different network environments, we took the test phone and drove to various places in a town. Similar to previous longitudinal experiment, the phone was set to run available bandwidth test and throughput test periodically in a back-to-back manner. We recorded GPS trace along with cell information provided from Mobile Insight. The test covers over 13 miles of distance and made 9 intentionally stops (each lasted for 20 minutes) to capture static measurements. Overall, the experiment lasted for 5 hours and we collected 1,360 tests from 12 cells. Fig 5.14 plots the drive route and the cell information, including cell ID (labeled by color) and RSRQ (indicated by the circle radius).

Cell-based Analysis

By categorizing the test results by the cell ID, we plot the measured AT and AB in Fig 5.15. In addition, we also plot the average RSRQ of the cell. Overall, we see that the estimated AB is close to AT and mostly below AT. However, when the channel quality is significantly ominous (RSRQ $\leq -14$ dB) the AB is noticeably

98
Figure 5.14. Drive test route map. Different colors denote different cells and the radius size indicates the RSRQ.
higher than AT under Cell 0 and Cell 1. The reason is that since AT test takes long time to finish, the bad signal power can consistently dampen the throughput. While the AB test is lightweight, the eNodeB may satisfy the short-term high bandwidth demand even under bad channel conditions [9]. This can lead to occasionally high estimations from the AB test. Under this circumstance, AB estimation can be higher than AT. Continuously performing multiple AB tests is a practical solution to resolve this discrepancy.

Movement Impact

To investigate the impact of movement upon our tests, Fig 5.16 plots the distributions of the estimation difference under different moving speed. To eliminate the impact of signal power, we choose the tests with RSRQ in the range of $[-10, -14]$ to plot this figure. We categorize the different moving speeds into three profiles: Sta-
tionary means no movement; Walk Speed refers to \([0, 10]\) mile/hr; Drive Speed refers to \([10, \infty]\) mile/hr. From Fig 5.16(a), we see that the movement can slightly reduce the difference between AT and AB. Combining the AT observation from Fig 5.16(b), we find that the reduction of \(AT - AB\) is mostly due to the fact that when speed increases, AT drops while AB stays relatively consistent. The reason behind the different reactions of AT and AB towards movement is similar to the previous case of RSRQ: the extremely short test duration of AB makes it less sensitive to the channel variations due to fading (e.g., path loss, Doppler spread, etc.).

5.5 Conclusion

The chapter introduced a novel available bandwidth estimation tool for LTE network. As the aggregation/batch effect observed on the modern LTE networks, the conventional packet-wise bandwidth solutions suffer from severe performance degra-
dation. This work leverages a physical layer transmission property - TBS Fluctuation - to reveal the potential bandwidth competition occurred on LTE network. By adopting a client-free probe system - FMNC, we design an “one-shot” available bandwidth estimation tool. The solution manipulates a Bayesian change point detection algorithm to cope with the noisy packet measurements obtained from real-world estimation. Through the extensive experimental evaluations, the results estimated from the proposed solution show a strong correlation with the traditional throughput test results, while with significant less test cost in terms of time duration as well as data consumption.
CHAPTER 6
PASSIVE WIFI CHARACTERIZATION VIA WIFI SCAN

The previous two chapters (Chapter 4 and Chapter 5) explored proactive approach of network characterization on mobile networks (WiFi and cellular). As a passive solution, in this chapter, we present an intriguing approach for passive WiFi traffic characterization. To solve this problem, we intrinsically need to answer two questions: 1) what traffic information required to conduct the characterization? 2) in what manner to obtain such information? We show 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 show 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. The implications for the work are considerable from both the end user standpoint, troubleshooting standpoint, and analysis/potential of cellular onto WiFi bands standpoint.

6.1 Preliminary Feasibility Study

Before diving into the details of our system, we discuss several key background concepts. First, in Section 6.1.1 we discuss the need for using control packets followed a basic primer on frame aggregation and block acknowledgement. Next, in Section 6.1.2 we describe how WiFi scanning typically operates and why it provides the potential for characterization.
6.1.1 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 [25]), 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. 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 non-directional manner. Secondly, 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 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 frame aggregation, we posit that the control packets observed on mobile devices can help deliver efficient traffic characterization.
Frame Aggregation

The general principle of frame aggregation is to assemble multiple data units to transmit as one aggregated frame\(^1\). The aggregation can be operated at two levels: aggregate MAC protocol service unit (A-MSDU) and aggregate MAC protocol data unit (A-MPDU). A-MSDU is on upper MAC layer which can be further aggregated again into A-MPDU when pushing into physical layer. Therefore, the frame transmitted in the air is eventually expressed in the form of A-MPDU. In this paper, we will focus on leveraging A-MPDU. As shown in Fig 6.1 in tandem with Block Acknowledgement (BA), each A-MPDU only requires one BA to notify the receipts of multiple MPDUs (i.e., packets\(^2\)). In order to support this one-to-many acknowledgement, a BA uses a bitmap field to explicitly indicate the failure or success of delivery of every single MPDU.

As frame aggregation has become the default manner of sending data on modern WiFi (802.11ac), data transmission will always invoke an exchange of a BA. These acknowledgements potentially provide opportunities to infer the data transmissions 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

\(^1\)Noted that the term \textit{packet} speaks to MAC and upper layers, while \textit{frame} refers to PHY layer.

\(^2\)Packet and MPDU are used interchangeably for the rest of the paper.
find that the information of how many MPDUs in an A-MPDU, dubbed Aggregation Intensity, 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. 6.2), we discuss the technical details about how we can manipulate the information to achieve accurate characterization based on control packets.

6.1.2 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 monitor mode. However, since monitor mode will suspend all communication, 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 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 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 our paper, 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 crowd sourced information, the humble WiFi now becomes an exciting new opportunity for network characterization.

6.2 WiFi Characterization via Control Packets

In order to characterize the WiFi channel, we define two primitive metrics: 1) the *Aggregation Intensity* 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.

6.2.1 Primitive Measurements

We begin with the first Primitive Measurement (PM)—*Aggregation Intensity*. Frame aggregation 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 *Aggregation Intensity* (AI) that counts the number of MPDUs (i.e., packets) within one aggregated frame. As an example in Fig 6.1, 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 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 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 bytes) and rarely reached, the AI is usually limited by the transmission
time. For a certain packet size $P$, we have $T_{\text{max}} \leq \frac{AIP}{R}$ where $R$ is the transmission rate. Thus the maximum AI can expressed as

$$AI_{\text{max}} = \frac{R \cdot T_{\text{max}}}{P} \quad (6.1)$$

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 6.2. Each bit on the bitmap represents the receiving status (success/failure) of a MPDU. The SSC includes a sub-
filed 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 of the corresponding A-MPDU (sent from B to A). For example, in Fig 6.3 we re-plot the case of Fig 6.1 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). When loss occurred as shown in the third A-MPDU, the 0 in the bitmap indicates the miss of the 1123 MPDU. Thus the loss count can be computed with the number of zeros in the bitmap.

Figure 6.4. (a) AI estimation error; (b) The maximum AI and loss rate across different transmission rate.
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 6.4a we plot the 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 Eq (6.1). 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 6.4b, the observation matches Eq (6.1) 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 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 the AI and loss across different network settings.

We now introduce the second primitive 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 primitive measurement — **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 6.1. 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, the larger the BA gap should be. The relationship can be expressed as:

$$G_{BA} \propto \frac{AI \cdot P}{R}$$  \hspace{1cm} (6.2)

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

$$G_{MPDU} \triangleq \frac{G_{BA}}{AI} \propto \frac{P}{R}$$  \hspace{1cm} (6.3)

Once AI is estimated with the method mentioned above, $G_{MPDU}$ can be computed.
From Eq. (6.2) and Eq. (6.3), 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 Mbps and repeated the TCP traffic on both 802.11ac 5GHz and 802.11n 2.4GHz. Fig 6.5a and Fig 6.5b 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 6.5a, $G_{BA}$ 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 6.5b 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 (6.2), (6.3).
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.

According to Eq (6.3), $G_{MPDU}$ should be inversely proportional to the transmission rate given certain packet size. We continue the validation by varying the transmission rates with fixed packet size (i.e., MTU). In Fig 6.6, we plot the empirical CDF of $G_{MPDU}$ under four different transmission rates including 33 Mbps, 144 Mbps, 433 Mbps and 866 Mbps. Because of the inconsistent behaviour under $AI = 1$ as discussed above, for the rest of the paper, $G_{MDPU}$ will be calculated by filtering out the gaps whose $AI = 1$. The result in Fig 6.6 matches the expectation of Eq (6.3) 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 next subsection, we will elaborate how to exploit the primitive measurements to deliver comprehensive traffic characterization regarding to various metrics.

6.2.2 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 subsection, we will elaborate how to derive each of these metrics from the primitive measurements.

Given the control packets collected during a certain time window $\omega$, 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 ($\sum_{\omega} AI$). Similarly, 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 $\omega$.

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

**Airtime**

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 [80]. 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 trans-
mision 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 transmis-
sion time of the A-MPDU data frames. Intuitively, given the BAs captured in a
time window $\omega$, summing up the $G_{BA}$ and dividing by $\omega$ 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

$$Airtime = \frac{\sum_{\omega} G_{BA}^{AI>1}}{\omega}$$

(6.4)

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

**Experimental Evaluation and Improvement:** Using the same experimental set-
ting before, we generated different sizes of TCP flow to cause different airtime cost
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 that once it completes we
re-start immediately. Each flow size ran for 10+ seconds. The ground truth airtime can be polled from the AP kernel via \texttt{iw} tool. With the control packets captured from the monitor node, we sliced them into continuous windows with a window size of $\omega$. Then we calculate the airtime for each window according to Eq (6.4).

In Fig 6.7 we plot the estimated airtime under different flow sizes with regarding to two window sizes ($\omega = 100$ \textit{ms} and $\omega = 20$ \textit{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 $AI = 1$ 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 $AI = 1$. 

Figure 6.7. Estimated airtime v.s. flow size w.r.t. window size $\omega$. 

![Figure 6.7](image-url)
Window Compensation: Comparing the results from different $\omega$, we notice the small window ($\omega = 20 \text{ ms}$) 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 6.1 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. So the airtime estimation will double count the first BA gap.

After applying the compensation, we re-plot the estimation accuracy $(1 - |\frac{\text{estimated} - \text{ground truth}}{\text{ground truth}}|)$ for different window size $\omega$ in Fig 6.8. 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., \( \omega < 20 \text{ ms} \)). 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 the compensation enabled, we can achieve at least 90% estimation accuracy when \( \omega \geq 20 \text{ ms} \).

**Transmission Rate**

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 \[12\]). But this information normally is not readily available at AP side as well as client side. Fortunately, with the properties of MPDU gap (in Fig 6.6), we can use it to make reasonable inference to the transmission rate. However, although 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 2.4GHz v.s. 802.11ac on 5GHz), interference and so on. In order to make $G_{MPDU}$ a reliable transmission rate indicator, it must be resistant to the influence from the other factors.

Through our empirical study from experiments, we find that flow size cause the most significant impact on MPDU gap distribution in addition to transmission rate. The influence from other factors is marginal. To demonstrate this impact, we use the trace from previous experiments to study the distribution of MPDU gap under different flow size. In Fig 6.9, we plot the CDF of $G_{MPDU}$ under 802.11ac 5GHz with fixed transmission rate (130Mbps). The flow size of heavy, medium and light refer to 160MB flow, 8MB and 400KB respectively. For each flow size, we collected over 2,000 BAs to generate the result. We see that when flow size decreases, the MPDU gap becomes larger. It is because that 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 6.9 reveals that even though the overall distribution varies across different flow size, 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 \( \theta \)-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 \( \theta \), we can alleviate the impact resulted from flow variation. The optimal setting of \( \theta \) will be explored through extensive experiments later. Once the effective \( G_{MPDU} \) is calculated, we can estimate the transmission rate by

\[
Rate = \frac{P}{G_{MPDU}^\theta}
\]  

(6.5)

where \( P \) is the packet size and \( G_{MPDU}^\theta \) is the \( \theta \)-percentile MPDU gap observed in a window.

**Inferring 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: MTU for data packets and 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\(^3\) 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 high degree of aggregation. To validate this assumption, we collected the experiment trace\(^4\) to plot the AI distribution from data or Ack stream under different flow sizes. In Fig 6.10, we

\[^3\]If only one direction traffic is detected, we assume it is the data stream.

\[^4\]The traces are selected from 802.11ac 5GHz under fixed transmission rate 325Mbps.
Figure 6.11. The impact of window size $\omega$ on the two components of estimating transmission rate.

see that for a certain flow, the AI of data link is always greater than of Ack link. This difference becomes more evident when flow size increases. Therefore, by selecting the link with 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 Ack link which make gaps fluctuate, we normally choose to the data link to estimate the transmission rate. Eventually, the rate can be computed from Eq (6.5).

**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 5GHz, since it is more challenging to estimation the rate with the wider range of values (up to 877Mbps) compared to 802.11n 2.4GHz. We start with investigating the impact of window size $\omega$. The performance of transmission rate estimation is decided by two components: inferring packet size and measuring $G_{MPDU}^\theta$. Fig 6.11 plots the impact of $\omega$ on the two aspects under different rates and flow sizes. We tentatively chose $\theta = 10$ for $G_{MPDU}^\theta$ 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 6.11a, regardless of transmission rate and flow size, the accuracy can reach above 90% once $\omega \geq 20 \text{ ms}$. The similar pattern is observed in Fig 6.11b that the measured gaps start to be consistent from $\omega = 20 \text{ ms}$. Combing the result from the previous experiment in Fig 6.8, it implies that the window size need to be at least 20 ms to yield reliable result. Moreover, from Fig 6.11b we see that the low rate (32Mbps) clearly has larger $G_{MPDU}$ than the higher rate (325Mbps) 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 $\theta$. In the following, we continue the experiments to search the optimal setting of $\theta$.

**$\theta$ setting:** From the observation from previous experiments in Fig 6.9 and 6.11b we see that the light flow is the most troubled case to obtain effective $\theta$-percentile $G_{MPDU}$. So we chose the light flow with $\omega = 20 \text{ ms}$ as the target to search $\theta$. Fig 6.12a shows the transmission rate estimation accuracy $(1 - \left| \frac{\text{estimated} - \text{ground_truth}}{\text{ground_truth}} \right|)$

Figure 6.12. Transmission rate estimation accuracy from different $\theta$ (a) and different flows (b).
from three $\omega$ settings. We see the estimation accuracy drops with transmission rate increasing. It is because when the MPDU gap is small under high transmission rate, it is more vulnerable to be bothered by the random noises (e.g., back off, time skew). From the three settings of $\theta$, 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 6.9).

With the $\theta = 5$, we evaluated the performance of transmission rate estimation across different flow size. As shown in Fig 6.12b 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/5GHz. In the low transmission rate range ($\leq$ 300Mbps), we can archive at least 75% accuracy for all flows. In 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.

**Queuing Indicator**

Queuing can affects network performance in various ways [11, 45]. To understand queuing information can help one diagnose or improve network conditions. This information requires access the kernel level on the device. It is difficult to access from the client side, especially with a passive approach. Fortunately, the primitive measurement 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

$$QI \triangleq \frac{AI}{AI_{max}}$$  \hspace{1cm} (6.6)
Figure 6.13. (a) Queuing Indicator performance compared with (b) beacon delay.

Since different transmission rates allow different $AI_{max}$ (Eq (6.1)), the same AI may indicate different queuing degree under different transmission rate. Therefore, we define QI as a normalized AI. The value is between 0 and 1 which describes the 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 Eq (6.1). By plugging in Eq (6.5) into Eq (6.1), we can compute $AI_{max}$ as

$$AI_{max} = \frac{T_{max}}{G_{MPDU}}$$

(6.7)

where $T_{max}$ is the maximum transmission time allowed. It is set by WiFi adapter manufacture (e.g., 4 ms in ath9k [33]).

**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 hardware queue. In order to evaluate the performance, we varied the TCP flow sizes to cause
different queue length. Similar to the setting in Fig 6.7, six flow sizes were used in this experiment ranging from 20KB to 160MB. The ground truth queue length can be polled from the Linux kernel. As the performance of QI largely depends on the robustness of $G^\theta_{MPDU}$, the impact of window size $\theta$ is similar with the previous case. Thus, we skip the $\omega$ impact study, and set $\omega = 20$ ms as the minimal feasible value.

In Fig 6.13, we plot the performance of QI contrasted with a prior work [110]. [110] uses the similar passive approach to infer queue length. But it exploits the delay of beacon frame from AP as the indicator. The designed QI (Fig 6.13a) monotonically increases with queue length regardless of transmission rate. However, the beacon delay (Fig 6.13b) is only responsive on low transmission rate (72.2Mbps). When transmission rate is high (144.4Mbps), beacon delay is not sufficiently sensitive to capture the subtle changes on queue length.

**Facilitating Transmission Rate Estimation:** In addition, QI can also serve as a confidence metric for facilitating transmission rate estimation. When serious queuing effect occurs, 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 re-plot the accuracy of transmission rate estimation as the function of QI from previous experiments. In Fig 6.14, 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.

6.3 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 paper, 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 scan, we are able to accomplish characterization on WiFi channels over time without triggering 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 \( \omega \) 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 behaviour.
6.3.1 Feasibility Study

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 setting. In order to understand the scan behaviour in the wild, we conduct the analysis on *scan interval* and *dwell time* from a real world dataset which involves high user diversity.

**Dataset summary**

We studied WiFi scanning by drawing from control packets captured in prior work [40]. 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 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 same device and set apart more than 1 second. The empirical CDF is plotted in Fig 6.15a. The observation shows that, about half of the scans have interval of less than 20 second and the 75-th, 90-th percentiles are 134.90 second and 390.73 second. The frequent scan behaviour in the dataset is due to the facts 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 2.4Ghz band. The intuition is that the packet transmitted on a channel (e.g., channel 2) can be heard on the adjacent channel (e.g., channel 1). For a probe request packet, the parameter *current channel* indicates its target channel. By measuring time gap between two consecutive probe requests (with consecutive sequence number) from 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 6.15b. 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 a 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 experiment results in previous section (Sec 6.2), 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 a few minutes. In the next section, we will evaluate the performance of this proposed mechanism under real world scenario.

6.4 Performance Evaluation

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 $\omega$ 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 crowdsourcing 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.
6.4.1 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 2.4GHz and 149 for 5GHz. By using the controller, we could obtain the ground truth information through SNMP (Simple Network Manage Protocol). We periodically (i.e., every 45 seconds 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 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 Tab 6.1. Over the 4-hour data collecting, we gathered total 8.6GB pcap files for only control traffic. There were over 51,000 unique devices\(^5\) sighted during the study. But many devices only showed up for a short time. Among them, 480 clients (350 on 5GHz and 130 on 2.4GHz) 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,

\(^5\)A device is identified with a unique MAC address.
TABLE 6.1

DATA STATISTICS SUMMARY AT THE TENT.

<table>
<thead>
<tr>
<th></th>
<th>2.4GHz</th>
<th>5GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Duration</td>
<td>4+ hrs</td>
<td></td>
</tr>
<tr>
<td>Data File Size</td>
<td>1.2GB</td>
<td>7.4GB</td>
</tr>
<tr>
<td># of Sighted Clients</td>
<td>19,116</td>
<td>33,134</td>
</tr>
<tr>
<td># of Active Clients</td>
<td>500</td>
<td>284</td>
</tr>
<tr>
<td># of Block ACK</td>
<td>181,164</td>
<td>2,636,317</td>
</tr>
</tbody>
</table>

the SNMP data should provide complete ground truth about the WiFi environment.

6.4.2 Evaluation with Ground Truth

Among the characterization metrics derived from our method, SNMP data only provide ground truth for two metrics: throughput and airtime. Therefore, we focus evaluation on the two commonly-used metrics. Since 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 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 6.16. Since the usage of 2.4GHz band is sparse in this case, only 5GHz band data is used for this evaluation. In Fig 6.16a, 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.
Figure 6.16. Time series of (a) airtime estimated compared with ground truth and (b) number of packets captured during dwell time for each scan.

Figure 6.17. Correlation between estimated value and ground truth w.r.t. number of devices used for generating result. All the results are from 5GHz band.
in Fig 6.16. 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 quantize the performance, we calculated the Pearson correlation coefficient\(^6\) between the estimated value and ground truth. A correlation value is calculated\(^7\) between the estimated value and ground truth. A correlation value is calculated

\(^6\)The figure is plotted after down-sampled to reduce visual clutter.

\(^7\)The value ranges from -1 to 1, where 1 implies perfect linear relationship while -1 implies negative linear relationship.
from a 90-minute window. By moving this window across entire time session, we can obtain over 200 estimation points. In addition, to explore the impact of crowdsourcing, we assume that only some clients 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 delivery accurate characterization through the crowdsource manner. We sort the sighted devices by their presence time. Thus the $x$ devices means the top $x$ sighted devices. In Fig 6.17 we plot the empirical CDF of the correlation coefficient on both airtime (Fig 6.17a) and throughput (Fig 6.17b). 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 10 devices contribute their characterization results, we can provide the accurate channel traffic characterization through crowdsourcing.

6.4.3 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 empirical study to analyze the WiFi traffic. In Fig 6.18 we plot the empirical CDF of the several metrics computed from our method. Particularly, we compare the different distribution from 2.4GHz and 5GHz band. Starting from the AI in Fig 6.18a since we had much more traffic load on 5GHz, the AI on 5GHz is higher than on 2.4GHz. Notably, there is about 20% of AI equal to 1 on 2.4GHz which could be due to the many light-flow traffic. The QI in Fig 6.18b reveals that with the higher capacity, although there were more traffic on 5GHz, it caused less backlog pressure compared to on 2.4GHz. At last, by filtering out
the results with $QI \leq 0.1$, we plot the estimated transmission rate in Fig 6.18c. The result matches our expectation that with 802.11ac supported on 5GHz, the transmission rate on 5GHz (with the median around 200Mbps) is greater than 2.4GHz (with median of 100Mbps). Overall, we see the rich set of characterization metrics from our method can help reflect insightful attributes about the WiFi traffic.

6.5 Conclusion

In this work, 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 the stadium, and various public venues.
CHAPTER 7

CASE STUDIES OF WIFI PASSIVE CHARACTERIZATION

As an extended evaluation of the passive WiFi characterization method proposed in the previous chapter (Chapter 6), this chapter further presents 1) an empirical case study in the wild (Section 7.1) and 2) a WiFi video streaming application case driven by the proposed method\(^1\) (Section 7.2). In Section 7.1, unlike the controlled environment in the previous chapter, we captured measurements from the uncontrolled network without ground truth. The purpose of this case study is to use the proposed method as a tool to reveal some interesting characterization observation in the wild. In Section 7.2 to demonstrate how to use the passive characterization to drive WiFi network adaptive services, we propose a cross-layer WiFi video streaming mechanism that manipulates the airtime information estimated from the passive characterization. The result shows that the proposed solution helps reduce the video stall rate significantly under congested WiFi environment.

7.1 Empirical Case Study in the Wild

In this study, we choose two real work scenarios to characterize the WiFi environment: 1) a university campus and 2) Chicago downtown. To capture raw data for each scenario, we use laptops equipped with multiple WiFi adapters (TP-Link WN772N and EdiMax 7822UAC) that each adapter is dedicated to monitor on one specific channel. We target on the non-orthogonal and non-DFS (dynamic frequency

\(^1\)This works has been published in ACM NOSSDAV 2018 [103]
TABLE 7.1

DATA SUMMARY OF THE REAL-WORLD STUDY.

<table>
<thead>
<tr>
<th></th>
<th>Campus</th>
<th>Downtown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.4GHz</td>
<td>5GHz</td>
</tr>
<tr>
<td>Time duration</td>
<td>2.5 hr</td>
<td>2.5 hr</td>
</tr>
<tr>
<td># of sighted APs</td>
<td>49</td>
<td>34</td>
</tr>
<tr>
<td># of sighted Clients</td>
<td>905</td>
<td>764</td>
</tr>
<tr>
<td># of active Clients</td>
<td>479</td>
<td>216</td>
</tr>
<tr>
<td># of Block ACK</td>
<td>1,739,760</td>
<td>2,378,076</td>
</tr>
</tbody>
</table>

selection) channels on both 2.4GHz (i.e., 1, 6 and 11) and 5GHz (i.e., 36, 40, 44, 48, 149, 153, 157, 161 and 165). Noted that, since we only collect the control traffic, the traffic transmitted on the un-monitored channel can be also heard on the primary channels we monitor on. Therefore, we can guarantee to cover all the channels. For the campus case, we choose a department building hall during a final week where many students gathered to prepare their exams. For the downtown case, we select a popular restaurant as the location where many customers used free WiFi while enjoying their food. With the densely deployed WiFi networks in the downtown area, we can not only capture the traffic inside this particular restaurant but also gathered the traffic from hundreds of APs nearby. The Tab 7.1 summarizes the statistics from our captured data. Noted that a sighted node in the tablet refers to a MAC address observed in any packet. While an active STA (station) particularly refers to a MAC address involved in data exchange (i.e., observed in Block Ack).

To parse the characterization result from the captured control traffic, different from the probe request based trace driven approach we used in the previous section, we perform a continuous analysis with 20 ms window size. We report the character-
ization result for every 20 ms window. By doing this, we emulate the ideal case that the WiFi scans are dense enough to cover every time point. Again, the purpose here is to use this generic method to WiFi environment in real world.

7.1.1 Result and Analysis

Starting from the airtime, we plot the average airtime of top 3 channels on each bands (2.4GHz and 5GHz) in Fig 7.1a. We see that although there are much more devices sighted in the downtown scenario than of the campus, the channel utilization of the campus is significantly greater than of the downtown. This difference can be also reflected on the number of the active STA as well as the number of Block Acks (refer to Tab 7.1). The reason is that, even with the large number of people sighted at the downtown case, most people rarely use the WiFi. Among the people that use WiFi, they just lightly surf the Internet with only about 551,000 Block Acks transmitted compared to the 4 million of campus case. For the campus scenario, we noticed that many students use their laptops to view online materials (text and video) for preparing final exams. It led to intensive channel utilization on WiFi. Another look at the difference on 2.4GHz and 5GHz is that, the utilization of 5GHz band at downtown is much less than 2.4GHz. As 5GHz suffers from severe signal attenuation, the business owners usually use 2.4GHz to gain wide coverage. While with the densely deployed APs at university campus, 5GHz band can be better exploited.

In Fig 7.1c we plot the CDF of the aggregation intensity observed in difference cases. Overall, AI mostly is in the low value that 80% of observed AI is less than 10 for all cases. Combining with the airtime result Fig 7.1a, we can see that the distribution of AI is related with the airtime consumption: high airtime (e.g., campus 2.4GHz) comes with relatively high AI value (e.g., the median of 3 at campus 2.4GHz). As we have discussed, AI is decided by both transmission rate and flow size. To further understand the cause for the low AI, we calculated the estimated transmission rate
Figure 7.1. Empirical observation of various characterization metrics from university campus and city downtown.

Based on our method, the transmission rate is computed as the peak transmission rate of a client during the 20 ms time window. To filter out the result with low confidence, in Fig 7.1b, we plot the estimated transmission rate for different scenarios when queue indicator is greater than 0.5. It is interesting to see that, although 5GHz can provide larger capacity than 2.4GHz with wider bandwidth, the overall transmission rate on 2.4GHz is higher than 5GHz. Zooming in on the distribution of transmission rate at high value (> 256 Mbps), we notice that the 5GHz band at the campus scenario is greater than other cases. It implies 5GHz still has its advantage on providing high
transmission rate. However, with the bad communication range, the full potential of 5GHz can not be exploited. In addition, we more isolated environment and exclusively controlled network, campus WiFi usually offers better transmission rate than public business WiFi.

At last, in order to investigate the scan behaviour under the different scenarios, we plot the CDF of scans rate (number of scans sighted in a one minute window). First, the scan rate is primarily decided by the client density. As shown in the Fig 7.1d, the scan rate is proportional to the number of clients referred in Tab 7.1. In addition, according to the study in [40], the scan rate can also be influenced by whether the WiFi is being connected. In the campus case where most devices were associated with WiFi, the scan rate is about 40 per minute given about 1,600 devices. While when many devices were not connected to WiFi in the downtown case, the median value of scan rate can reach up to 200 at 2.4GHz and over 400 at 5GHz.

7.2 Application: A Cross-Layer Stall-Free WiFi Video Streaming Mechanism

In this section, we introduce an application case of using the passive characterization method to facilitate the bitrate adaptation on WiFi networks. We begin with describing the motivation by demonstrating an experimental study to show the problematic behavior of existing works.

7.2.1 Problem Demonstration

We conduct a controlled lab experiment to demonstrate video streaming performance under busy WiFi. The experimental setting is as follows: we set up a client to stream video content from a local server via a WiFi link (i.e., 802.11ac on 5GHz). On the same WiFi channel, we generate competing traffic (i.e., long-lived TCP flows) from other WiFi clients. In this experiment, while streaming 100-second video content, we launch two competing clients at the 20 and 40 seconds respectively. Then,
competing traffic is stopped at the 80 seconds. We defer other detailed settings to the later section at Sec. 7.2.3. We show the stall rate (the percent of the time when video buffer is drained out) for several popular bitrate adaptation algorithms [32, 44, 48]. For reference, we also tested fixed bitrate approaches, i.e., the minimum (3Mbps) and maximum (52Mbps) used in Sec. 7.2.3.

Result

Table 7.2 lists the stall rate for the different algorithms. First, we see that the bandwidth is sufficient to support the minimum bitrate with a negligible 2% stall rate. The maximum bitrate causes significant stalling (73%) as expected. The three existing algorithms all experience unacceptably high stall rate (around 50%). Specifically, the highly variant throughput on WiFi makes the latest throughput approach [32] hard to predict. Although smoothed throughput [48] can help stabilize the bitrate, the stale reaction to bandwidth variation can still lead to severe stalling. The dynamic throughput leads to fast-changing buffer level which hampers the buffer-based approach [44].

As mentioned earlier, the throughput on the WiFi link can be largely captured by the airtime on the channel. Thus, we further look into the airtime behavior during

---

TABLE 7.2

STALL RATE FOR DIFFERENT ALGORITHMS.

<table>
<thead>
<tr>
<th>Algo.</th>
<th>min Bi-</th>
<th>Latest Thrpt. Buffer- max Bi-</th>
<th>Thrpt. mean* based Bi-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>Thrpt. mean*</td>
<td>based Thrpt. Bi-rate</td>
</tr>
<tr>
<td>Stall Rate(%)</td>
<td>2.12</td>
<td>51.85</td>
<td>45.50</td>
</tr>
</tbody>
</table>

*harmonic mean.

---
the streaming experiment described above. In Fig 7.2 we plot the time series of the latest throughput-based adaptation algorithm. The subplots are (a) the segment throughput/bitrate selection (b) the instantaneous video buffer (c) the WiFi channel airtime. Particularly, we break down the airtime consumption for competing traffic and video traffic. As shown, without competing traffic (i.e., 0-20 s and 80-100 s), the video can be smoothly played on the highest bitrate. When competing with other clients (from 20 s onwards), the WiFi channel starts to become crowded and the video client starts to experience a significant throughput drop. However, what causes the stalling is not the low throughput since we knew that the minimum bitrate is well-supported (Table 7.2). Instead, the root cause is due to the faulty bitrate selection when throughput fluctuates. The transient high throughput under channel
congestion consistently misleads the video client into over-selecting, which results in video stalling. In addition, frequently switching the bitrate further exacerbates the variation of throughput.

**Implication**

From the experimental observation above, we can see that predicting the highly-transient bandwidth under congested WiFi is impractical. Instead, under this circumstance, conservatively staying at a fixed bitrate (e.g., minimum) can potentially avoid over-selection and aggressive switching. Therefore, we argue that in order to avoid video stalling, a video client needs to *infer channel competition on WiFi and act accordingly*. In the next section, we will introduce the proposed channel-aware cross-layer bitrate adaptation mechanism.

### 7.2.2 SEWS: A Stall-Free WiFi Streaming Adaptation Algorithm

According to our discussion earlier, the bitrate over-selection caused by transient throughput spikes during channel competition is the culprit of video stalling. The goal of our algorithm is to avoid faulty bitrate increases under the congested WiFi channel. The proposed algorithm will use the latest throughput as the baseline. The primary task is to identify the problematic case when the throughput increase is a false signal for bitrate increases. By analyzing the airtime on WiFi, we find that the erroneous bitrate increasing usually occurs under *two conditions*: 1) *the channel is fully occupied* and 2) *the video streaming link is not the dominant link* (i.e., the highest airtime link). Namely, if the two conditions occur simultaneously, a video client should *not increase bitrate* regardless of the throughput.

When the channel is not fully consumed, the throughput-based approach normally has no issue. When the channel is fully occupied, the throughput on video link can still increase to suppress the other traffic *if and only if* the video link is the
Algorithm 1: Bitrate adaptation algorithm.

**Data:** cur_rate, Thrpt, A, R, α

1. not_increase = false
2. for i = 1 to N do
3.   if A_i > A_v and \( \sum A > \alpha \) then
4.     not_increase = true
5.     break
6. new_rate = 0
7. for j = 1 to M do
8.   if Thrpt > R_j then
9.     new_rate = R_j
10. if new_rate > cur_rate and not_increase then
11.   return cur_rate
12. else
13.   return new_rate

most aggressive link that dominates the airtime consumption. Therefore, when two conditions are triggered, the video client should be precluded from bitrate increasing.

Overall, the algorithm can be described as follows in Algorithm 1. For each video segment, the algorithm will take the following information as input: the current video bitrate `cur_rate`, the measured throughput for the previous segment `Thrpt`, the airtime `A` measured during the previous segment playback time. We estimate airtime `A_i (i = 1, ..., N)` for each active link on the channel. Particularly, we denote the video link airtime as `A_v`. In addition, we need the set of bitrate options `R` for all levels of rate `j = 1, ..., M`. At last, we define a fully-occupied channel when overall airtime is beyond `alpha`.

As described in Algorithm 1, the `not_increase` signal is used to control whether bitrate increasing is allowed based on the two criterion discussed earlier. In the following section, we will conduct extensive lab experiments to evaluate the performance of the proposed scheme under various realistic network settings.
7.2.3 Performance Evaluation

To conduct our experiments, we implemented the system on a Commercial off-the-shelf (COTS) device. The overall system work flow is shown in Fig 7.3. To capture the Block Acks for airtime estimation, the WiFi adapter needs to eavesdrop on the channel while maintaining communication. This requires a firmware modification which is impractical on commodity devices. Therefore, for evaluation purposes, we employ an extra USB WiFi adapter in monitor mode to conduct airtime characterization. This extra adapter is plugged into the video client laptop. The airtime information is fed to the video client through a messaging library zmq.

Experiment Settings

We set up a DASH server via Apache on a laptop (HP ProBook 4730s). The server was connected to a controller (another HP laptop) which was used to configure the network environment (e.g., round trip time—RTT). WiFi clients connected to the controller via WiFi AP(s). We used TP-Link Archer C7 V2 802.11n/ac as the AP(s), and HP 250 G6 notebook (equipped with Intel Wireless 3168 802.11ac chipset) as the clients. For the video client, we used an Edimax EW-7822UAC USB adapter for airtime characterization purpose. The experiments were primarily run on 802.11ac 5G channel (channel no. 149). We set up two identical APs on the same channel to create competition. The WiFi link can achieve up to 150Mbps throughput with

\[^2\text{zeromq.org}\]
20ms RTT. For the video streaming setting, the DASH server could provide 7 different bitrates, including 3, 7, 17, 27, 32, 37, 42, 52Mbps. The segment length was set to 1 second and the client buffer length was 2000 ms as the low latency setting. We set $\alpha = 0.86$ based on empirical observations. Due to space constraints, we will not present the detailed results of searching the setting of $\alpha$.

**Illustration**

To demonstrate the working principle, we repeated the experiment in Fig 7.2 with our proposed method. In Fig 7.2a, we indicate the *not increasing* in Algorithm 1 when bitrate increasing was prohibited under channel competition. In Fig 7.4c, we further break down the airtime cost on each link. We see that when competing with the heavy flow(s), the video client recognized 1) the channel was busy ($\sum A > \alpha$)
and 2) the competing links were more aggressive than the video link ($A_i > A_v$). Therefore, it purposely prohibited bitrate increases during this time regardless of the throughput bumps. Overall, the designed method successfully prevented the video buffer from drained out (Fig 7.4b).

**System Micro-Benchmark Test**

We continue to conduct system-level micro-benchmark tests and comparison with the existing flagship works. We selected five contrast DASH adaptive algorithms: BBA[44], FESTIVE[48], PANDA[61], GPAC[32] and minRate. BBA is a buffer-based approach that maps the instant buffer level to certain video bitrate adjustment. We adopted the BBA-0 implementation proposed in [44]. FESTIVE uses the harmonic mean of throughput history to guide bitrate selection. We set the smooth window to 5 seconds. PANDA behaves like TCP AIMD (Additive Increase Multiplicative Decrease): it keeps increasing bitrate with an additive term; once throughput decreases, it reduces the bitrate with a multiplicative term. The parameters are tuned based on [61]. GPAC leverages the latest throughput to select video bitrate. For reference purpose, we also ran the fixed bitrate by using the minimum bitrate.

**Evaluation Metrics:** Three metrics are employed to evaluate the performance, including *stall rate*, *smoothness* and *video bitrate*. *Stall rate* is the percent of the time when video buffer is drained out. *Video bitrate* computes the average of bitrate selected over entire video. *Smoothness* measures the intensity of bitrate switching which can be computed as the average absolute video bitrate change per segment. A good adaptive algorithm should try to stabilize the bitrate and yield a low smoothness value.

**Heavy Flow Stress Test**

We first evaluate the video adaptation algorithms under heavy-flow competition.
The experiment started with streaming a 100-second on a clear WiFi channel. At 20 second, the competing client launched a long-lived TCP flow by fetching a big file from the server. The competing traffic lasts for 60 seconds and stops. We varied the number competing clients from 1 to 5. For each adaptation algorithm under a certain client number, we repeated the experiment over five times.

**Results Analysis:** Fig 7.5 plots the performance comparison of all the algorithms. In the figures, the boxes are the means and the bars denote standard deviations. We selectively plot the results from 1, 3, and 5 competing clients. First, as one of the
most critical QoE metrics, we see that the stall rates in Fig 7.5a are unacceptably high (> 0.4) for all approaches except for the minRate and the proposed method. The stall rate worsens when the client number increases. Overall, our method can maintain the stall rate within 0.2 even under severe competition with 5 competing clients. Compared to the second-best solution (BBA), it achieves up to 20x (0.41 v.s. 0.02) performance gain under the 1 competing client. FESTIVE ranks right behind BBA. Its dull reaction to bandwidth variation, due to the smoothed throughput measurement, results in high stall rate. PANDA aggressively raises bitrate to probe bandwidth which leads to the highest under this congested network. GPAC is sensitive to throughput fluctuation which makes it utterly vulnerable to faulty bitrate over-selection.

We continue to compare video bitrate and smoothness. From Fig 7.5b, we see that, with the similar low stall rate compared to the minRate, our proposed method can offer 7x video quality gain. Although the other adaptation algorithms provide better bitrate, the cost of stalling is undesirable. Because video stalling is more detrimental than low video bitrate from the QoE standpoint. The high video quality is always running the risk of bad stall rate. For instance, PANDA has the highest overall bitrate due to its aggressive probing but it provides barely watchable video with up to 0.78 stall rate. With more competing clients, video quality decreases because of the less available airtime on the channel. Note that, the bitrate of our method is relatively consistent across different numbers of competing clients. It is because once a certain level of competition is detected, our method usually prefers to staying at a low rate. This behavior also brings good video smoothness. As we can see in Fig 7.5c, the proposed method achieves the best smoothness expect for the minRate. Overall, as expected, the approaches (i.e., FESTIVE, PANDA) that adopts smoothed measurement yield better smoothness than the other approaches (i.e., BBA, GPAC).
Figure 7.6. Performance comparison under on-off light/medium-flow competing traffic.

**On-Off Light/Medium Flow Test**

In normal traffic environment, light/medium-flow traffic can be more prevalent than heavy-flow traffic. Therefore, we conducted a further evaluation under light/medium flows in the WiFi environment. Similarly, in this experiment, we started with streaming a 120-second video on a clear WiFi channel. Meanwhile, the competing client(s) generated traffic lasting for 20 seconds and then slept for 20 seconds recursively. For each competing client, the competing traffic was generated by launching a TCP
flow with the flow size randomly selected from three values (i.e., 2MB, 8M, 160MB). Again, we varied the competing client number from 1 and 5. For each client number and adaptation algorithm, we repeated the experiment over five times.

**Results Analysis:** As the result shown in Fig 7.6, the overall stall rate is much milder than the heavy-flow traffic scenario (Fig 7.5a). The general stall rate ranking is unsurprisingly similar to the previous scenario. However, in term of video bitrate (Fig 7.6b), we see that our proposed method can achieve comparable video quality as other methods while maintaining extremely low stall rate. For example, under 3 competing clients, our proposed method can yield 73% (35Mbps v.s. 48Mbps) out of the highest bitrate solution *PANDA*. But we can achieve four times of stall rate reduction from 0.47 to 0.11. Namely, the significant reduction in stall rate only costs a small amount of sacrifice on video bitrate. In terms of smoothness, the observation in Fig 7.6c reveals similar pattern as the previous case. It is noteworthy that when the number of clients increases, the smoothness becomes worse in this case. It is because the on-off pattern can force more bitrate switching. In summary, the experimental evaluation proves that the proposed method can effectively reduce the stall rate while keeps acceptable bitrate in various WiFi traffic conditions.
CHAPTER 8

FUTURE WORK

To wrap up the dissertation, in this chapter, we will discuss the future works of continually improving this work as well as exploring the field of mobile network characterization. We divide the discussion primarily into two categories: 1) technical problems in the research spectrum and 2) potential applications in industry.

8.1 Research Problem

Reflecting on the limitations and deficiencies of the works proposed in this dissertation, we list a series of research problems that are worthy of exploring in the future:

• **Reconciling AB and QoE:** In this dissertation, we mainly focused on measuring bandwidth condition by using the metric of available bandwidth. However, since the available bandwidth only indicates a minimum obtainable bandwidth, the estimated result does not necessarily reveal what a user may actually get. We dub the difference between available bandwidth and achievable throughput as elasticity. The problem of how we can interpret available bandwidth to accurately reflect QoE, e.g., achievable throughput, is largely remained unanswered. Namely, given the estimated available bandwidth, we need to translate this value to indicate the user experience, e.g., how long it takes to load a web page, how much bitrate can achieve on video streaming and etc.

• **Interpolating/Extrapolating the estimation:** Due to the nature of lightweight, available bandwidth estimation tends to sample the bandwidth in a small time window. The estimation is usually sensitive to any transient effect occurred on the wireless channel. In order to exploit the estimated result(s) to indicate the bandwidth condition in a different time slot(s), we have to study the applicability of the estimated result. Technically, the problems involve interpolating (given several estimated results, infer the estimation of a point between
the results) and extrapolating (predicting future or past result given a series of tests). How to model the variations and provide accurate inference are the keys problems in the future research.

- **Extending to more mobile networks:** In this dissertation, we compound the different physical layer transmission patterns on WiFi and LTE networks to facilitate bandwidth characterization on the IP layer. As the advances of the future mobile networks (e.g., Massive MIMO in 5G), we need revisit the bandwidth models under different physical medium (e.g., visible light communication, mmWave, etc) as well as different network architecture (e.g., heterogeneous network). The bandwidth characterization in those cases may require intrinsically different perspectives.

- **Leveraging the power of machine learning:** The bandwidth characterization usually depends on a mathematical model formulation. For example, in this dissertation, the very fundamental assumption is that we perceive the available bandwidth as a function of packet queuing. An alternative approach to tackle this problem is to leverage the intelligence of machine learning that we may learn such a model from the empirical data. Particularly, by taking advantage of unsupervised learning, we could gain a more insightful perspective regarding the bandwidth model.

8.2 Industrial Application

In addition to the research problems, there are also a large body of potential applications of mobile network characterization from the perspective of industrial products.

- **Improving bandwidth traffic engineering:** With the intensification of mobile network development, the future network bandwidth requires better management. The bandwidth characterization methods proposed in this dissertation aimed at minimizing the intrusiveness of the bandwidth tests. The lightweight property may enable a new method to conduct traffic engineering based on network characterization. The traditional network engineering approaches (e.g., rate limit) normally take no user-side observation into consideration. In the future, network engineering could involve user-side bandwidth estimation to achieve more intelligent resource allocation which aligns with the effort of the cognitive network.

- **Generalizing bandwidth adaptive services:** So far, only some certain services are implemented to adapt to bandwidth condition. With the maturity of bandwidth characterization, we may expect to enable more network services to be adaptive, e.g., web browsing, online phone call and etc. So that all kinds of network
services can take full advantage of the future high-speed network to maximize the QoE. For example, with the rising of VoLTE (Voice of LTE), the voice quality over the cellular network can be largely improved that utilizes high bandwidth to provide high fidelity (HiFi) experience.

- **Integrating with existing network stack:** In this dissertation, we explore the packet train based network characterization approach. This method requires sending a certain pattern of packet sequence which usually demands the modification on the transport layer. In today’s network, the boundary between the transport layer and the upper layers starts to blur, as the companies try to improve security and performance by unifying the transport and above layers. QUIC \[38\] employed by Google Chrome is a successful effort to integrate the transport layer design with the transport security layer (TSL). This unified design enables the packet train based bandwidth characterization method to be embedded into data transmission. In the future, we expect data transmission can innately work in tandem with network characterization, where the characterization can guide better scheduling for data delivery in a reciprocal way.
BIBLIOGRAPHY

2. iperf3. URL software.es.net/iperf


wlans. In 2009 Fifth International Conference on Mobile Ad-hoc and Sensor


24. 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. UbiComp

25. Cisco. 802.11ac: The fifth generation of wi-fi technical white paper, 2015. URL
wireless/aironet-3600-series/white_paper_c11-713103.html.

collateral/service-provider/visual-networking-index-vni/
complete-white-paper-c11-481360.html"

wi-fi link capacity estimation on commodity access points. In Traffic Monitoring
and Analysis Workshop (TMA) 2016, 2016.

for accurate admission control decisions in mobile ad hoc networks. IET
Communications, 1(4):577, 2007. ISSN 17518628. doi: 10.1049/iet-com:
20060243. URL http://digital-library.theiet.org/content/journals/
10.1049/iet-com{_]20060243.

29. S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or
http://dl.acm.org/citation.cfm?id=2663716.2663727

30. C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion tech-
puter Communications. Twentieth Annual Joint Conference of the IEEE Com-
puter and Communications Society (Cat. No.01CH37213), volume 2, pages 905–

31. A. B. Downey. Using pathchar to estimate Internet link characteristics. In
Proceedings of the 1999 ACM SIGMETRICS international conference on Mea-
surement and modeling of computer systems - SIGMETRICS ’99, pages 222–
32. ENST. GPAC. URL "https://github.com/gpac/gpac"


113. Wikipedia. ping, . URL "https://en.wikipedia.org/wiki/Ping_(networking_utility)"


