WiFi Can Be the Weakest Link of Round Trip Network Latency in the Wild

Changhua Pei[†], Youjian Zhao[†], Guo Chen[†], Ruming Tang[†], Yuan Meng[†], Minghua Ma[†], Ken Ling[‡], Dan Pei[†][*]
[†]Tsinghua University [‡]Carnegie Mellon University
[†]Tsinghua National Laboratory for Information Science and Technology (TNList)

Abstract— **As mobile Internet is now indispensable in our daily lives, WiFi's latency performance has become critical to mobile applications' quality of experience. Unfortunately, WiFi hop latency in the wild remains largely unknown. In this paper, we first propose an effective approach to break down the round trip network latency. Then we provide the first systematic study on WiFi hop latency in the wild based on the latency and WiFi factors collected from 47 APs on *T* university campus for two months. We observe that WiFi hop can be the weakest link in the round trip network latency: more than 50% (10%) of TCP packets suffer from WiFi hop latency larger than 20ms (100ms), and WiFi hop latency occupies more than 60% in more than half of the round trip network latency. To help understand, troubleshoot, and optimize WiFi hop latency for WiFi APs in general, we train a decision tree model. Based on the model's output, we are able to reduce the median latency by 80% from 50ms to 10ms in one real case, and reduce the maximum latency from 250ms to 50ms in another real case.**

I. Introduction

As mobile Internet is now indispensable in our daily lives, WiFi's latency performance has become critical to mobile applications' quality of experience for the following reasons. First, WiFi has become the primary Internet access method. In 2013, 55% of the Internet traffic traverses WiFi as the last hop [1]. Second, users expect fast response time from mobile applications. [2] shows that if the page load time is larger than 3 seconds, 40% of users will abandon the pages. It also indicates that nearly 47% of web users expected a web page to be loaded within 2 seconds. Third, majority of the mobile applications rely on HTTP protocol, which is sensitive to network latency. [3] shows that every 10 milliseconds latency increase on broadband access will cause 1000 milliseconds increase in Web page load time. Fourth, the WiFi interference, when presents, can cause long packet latency at the WiFi hop. A back-of-envelope calculation using the above numbers from [2] and [3] shows that, WiFi hop latency should be under a stringent threshold of 20$\sim$30ms in order to satisfy the user expectations.

Despite its importance, WiFi hop latency in the wild remains largely unknown, probably due to the lack of effective measurement methodologies. In this paper, we conduct the first systematic study to answer the following three important questions regarding WiFi hop latency and WiFi factors in the wild:

- *What does the WiFi hop latency look like in the wild?*
- *Which factors influence the WiFi hop latency the most?*
- *How to optimize WiFi hop latency in the wild?*

[*] Dan Pei is the corresponding author.

This paper makes the following contributions.

First, we propose an approach called **WiLy** (*Wi*Fi hop *L*atenc*y*) that effectively breaks down the *Round Trip Network Latency* (**RTNL**, which is RTT minus the *server processing time*. More details are in §II-A) of TCP packets into uplink wireless latency (*UL*), wired network latency (*WL*), and downlink wireless latency (*DL*). WiLy can accurately measure *DL* for all TCP packets, and use the accurately measured *UL* and *WL* for TCP packets in 3-way handshake to approximate *UL* and *WL* of other TCP packets. WiLy approach does not require any client-side instruments, and can be deployed on OpenWrt-compatible APs without any support from WiFi chip vendors. To the best of our knowledge, WiLy is the first such approach reported in the literature.

Second, we present the first systematic study on WiFi hop latency in the wild. 47 OpenWrt-based Access Points (APs) (with WiLy enabled) were used by student volunteers in *T* university campus as their primary access to the Internet. We measured WiFi hop latency and WiFi factors (such as RSSI, retry ratio, airtime utilization) for a continuous period from $20^{th} May$ to $20^{th} July$, and accumulated more than 2 terabytes of data. We observed that for certain APs, more than 50% of TCP packets suffer from WiFi hop latency larger than 20ms, and 10% of TCP packets have WiFi hop latency larger than 100ms. It is surprising that the WiFi hop latency takes more than 60% of RTNL for over half of cases. WiFi, the primary last-hop Internet access method, has become the weakest link in the round trip network latency in many cases.

Third, utilizing the WiFi factor dynamics in our large data set, we train a generic decision tree model which can provide optimization guidance to both our APs or other APs. **1)** The interpretable decision tree shows which WiFi factors have the most important influence on the latency. For example, we observe there is an over 67% possibility to have undesirable latency experience when the airtime utilization is larger than 55%. **2)** In one real case, we use our general Decision Tree to classify packets with their WiFi factors. The classification results show that the main cause of this AP's long latency is high airtime utilization. After switching to another channel whose airtime utilization is 8% lower, the AP's largest latency was reduced from 250ms to 50ms. **3)** In another real case, the classification results show that client RSSI is the problem.

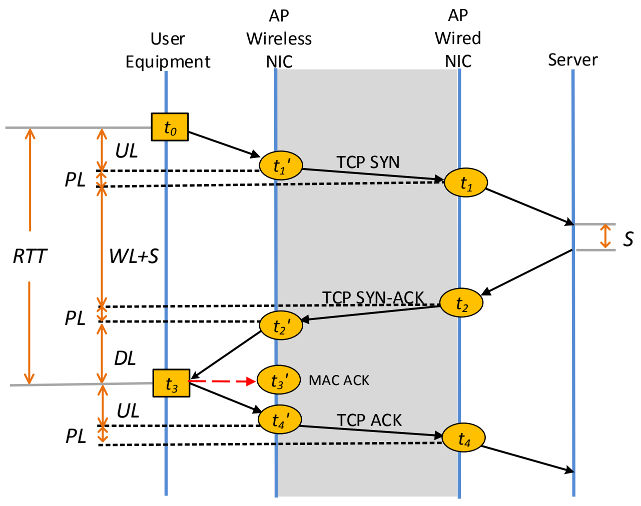


Fig. 1. Example of round trip time **RTT=RTNL+S**.

Relocating the AP reduced its median latency by 80% from 50ms to 10ms.

II. Breaking Down Network Latency

In this section, we first describe round trip network latency (RTNL, RTT minus the *server processing time*) using the life time of involved packets. Next, we introduce our data collection platform and methodology. At last, we describe our approach WiLy to breaking down the network latency into different parts using TCP 3-way handshake packets and data packets respectively.

A. Timestamps in Round Trip Network Latency

With the purpose of studying the RTNL, we first study the life of a packet p generated from the user and its response packet q generated by the server in a round-trip communication, to extract latency component. We focus on the TCP packets because TCP is the dominant transport protocol [4], [5]; and HTTP, today's most popular application protocol, which is delay sensitive [3], is based on TCP protocol. Now we take a look at the lifetime of TCP 3-way handshake to study each part of network latency during a RTNL.

Fig. 1 shows an example of the packets' life in a round-trip communication in the TCP 3-way handshake, and what latency they will encounter. In this example, p is the TCP SYN packet, and q is the TCP SYN-ACK packet.

At the very beginning, packet p is generated by user equipment (UE) at time t_0. After traversing queuing time, channel backing off time, transmission time and potential packet retrying time, p reaches the AP's wireless NIC at $t_1^{'}$. The time from t_0 to $t_1^{'}$ is defined as WiFi uplink latency, labeled as *UL*.

As soon as p arrives at the AP's wireless NIC, it will be processed and forwarded to the AP's wired NIC. We use t_1 to denote the time when p reaches the wired NIC. $t_1 - t_1^{'}$ is the processing latency of AP, labeled as *PL*. After that, AP waits until time t_2 to receive the response packet q arriving at its wired NIC. The elapsed time $t_2 - t_1 - S$ is called wired latency, which is denoted as *WL*. S is the processing time on server side. *WL* mainly consists of network latency on the wired Internet between the AP and the server.

When q arrives at the AP's wired NIC, it will be processed and forwarded to the wireless NIC. We use $t_2^{'}$ to denote the time when q reaches the wireless NIC and waits in the sending queue to be sent out. Similarly, $t_2^{'} - t_2$ is the processing latency of AP, *i.e. PL*. When q arrives at wireless NIC of AP, it also experiences the queuing time, channel backing off time, transmission time, and packet retrying time before the MAC ACK from UE is successfully received by the AP. We use t_3 to denote the time when q reaches the UE, and use $t_3^{'}$ to denote the time when q's MAC ACK from UE is received by the AP. Similar to *UL*, the time from $t_2^{'}$ to t_3 is called WiFi downlink latency, denoted as *DL*. By now, *p,q* finish their travels in a round-trip communication. Time from t_0 to t_3 is defined as the round trip time (**RTT**). Here, **RTNL** $= RTT - S$.

B. Data Collection Platform and Methodology

1) Platform: Our data is collected through a real world deployment within a China's University T, which has about 42,000 students and 11,000 faculty and staff members. We distributed 47 access points (APs) to students in T campus for free and let them use these APs as their primary method to access the Internet. All the distributed APs are *NETGEAR N750 WiFi Dual Band Gigabit Router Premium Edition (WNDR4300)*. Each AP has a 560MHz AMD CPU, 128MB DRAM, 128MB Flash Memory, four 1Gbps LAN ports, 1 1Gbps WAN port and two Atheros wireless interfaces (one b/g/n and one a/n).

Among these 47 APs on T campus, 44 APs are deployed in the student dorms while the rest in graduate student offices. On each floor of different dormitory buildings, the number of rooms varies from 20 to 40, with the same room size of $5 \times 4m^2$.

2) Data Collection: We implement a lightweight user-level measurement program WiLy on top of each AP's operating system (OpenWrt Barrier Breaker [6] version). WiLy can be deployed on any OpenWrt-compatible APs without any support from WiFi chip vendors. WiLy continuously collects data which include i) abstract information of packets from the real traffic and ii) WiFi factors that can reflect current WiFi environment characteristics.

Packet capturing: We use the standard *libpcap* [7] library to capture packets in real traffic through the AP. In this paper, we take TCP packets as our study sample, which most of the Internet services are based on [4], [5]. Unlike UDP, the reliable communication manner (*e.g.* immediate ACK for SYN packet) of TCP helps us to extract the network latency from the whole RTT more accurately, and avoids conflating with the up-layer application latency. For the privacy and performance concerns, we only extract the following fields of a TCP packet: IP address, Sequence Number, Acknowledgment number. All the IP addresses are anonymized and cannot be traced to a certain user. The Sequence Number and Acknowledgment number are used to reconstruct the TCP interactions. We also record the AP's system timestamps when each TCP packet arrives at and/or leaves AP's certain NICs, to further extract network latency. During the two months from *May* 20^{th} *2015* to *July* 20^{th} *2015*, we have collected 2 terabytes data in total from the 47 active APs. All the generated data are compressed and synced to the server using $rsync$ tool [8] every 2 minutes.

C. Breaking Down RTNL of TCP 3-way Handshake in the Wild

After showing the anatomy of the TCP 3-way handshake, we break down its round-trip latency into several individual parts (*UL*, *WL*, *DL*)[1]. However, the practical challenge is how to measure these latencies while we can only directly measure the seven AP timestamps shown as ovals in Fig. 1. Specifically:

- *WL*: *WL*= $t_2 - t_1 - S$. The wired latency can be directly calculated using t_2, t_1, which are both available by recording the time when packets arrive at the wired NIC in the AP. Note that the processing time (denoted as *S*) on server side is negligible for TCP 3-way handshake because there is little processing time for TCP SYN packet on the server. We thus have *WL*$\approx t_2 - t_1$ for TCP 3-way handshake.
- *DL*: *DL*= $t_3 - t_2^{'}$. **Approximating** $t_3 \& t_2^{'}$: The first challenge WiLy needs to address is how to measure t_3 and $t_2^{'}$. $t_3^{'}$ can be directly measured at the AP. Once the MAC-layer acknowledgement of q arrives at wireless NIC, the driver of wireless NIC will call a function to report that q is delivered and forward q to the capturing program listening on the wireless NIC where q is timestamped. This time is recorded as $t_3^{'}$. Then we can use $t_3^{'}$ to approximate the t_3 because: 1) MAC ACK will not be queued at UE side and the MAC ACK's channel backing off time can be neglected because MAC ACK has higher priority than other packets [9], 2) the transmission time of MAC ACK can be neglected because of the small packet size ($\frac{14*8}{6Mbps} \approx 19us$), 3) the frame error of MAC ACK rarely happens and can be ignored according to [10]. For $t_2^{'}$, we use t_2 to approximate it. The actual measurement results confirm that the timestamp gap between t_2 and $t_2^{'}$, *i.e.*, *PL* is small enough and can be neglected compared with *DL*. Similarly, we have $t_i \approx t_i^{'}, i = 1, 2, 4$.

  Matching a wired packet with its corresponding encrypted wireless packet: Another challenge of WiLy is that, given $t_3^{'}$ is obtained from the wireless NIC while t_2 is got from the wired NIC, how can we get the corresponding $t_3^{'}$ and t_2 pair which belongs to the same packet q? [11] merges the wired trace and wireless trace by matching the content of packets one by one. However, it is impractical for measurement in the wild because the majority of real world APs (especially residential ones) encrypt the outgoing packets from AP using the SSID password. This encryption makes the wireless packets obtained from the wireless NIC unrecognizable for matching with the wired packets. To address this challenge, we make small modifications in the driver of OpenWrt APs. We embed time t_2 in an in-kernel structure of data packets which will remain intact in the AP driver's buffer until its 802.11 MAC ACK is received at $t_3^{'}$. At this time this packet with t_2 timestamp embedded will be forwarded to *libpcap*, and we can retrieve the t_2 from the 802.11 MAC ACK packet.
- *UL*: The exact uplink latency starting from the UE is $t_1^{'} - t_0$. t_0 is the time when client prepares to send p (the SYN packet). However, we cannot get t_0 without user-side instruments, thus we are not able to directly measure the exact uplink latency $t_1^{'} - t_0$. Instead, we use the next uplink packet TCP ACK's *UL* ($t_4^{'} - t_3$) to approximate *UL*= $t_1^{'} - t_0$, by having two assumptions. First, we assume the WiFi environment factors are similar between t_0 and t_3. This is because t_0 to t_3 is a relatively short period of time during which there are little changes on WiFi factors. Second, we assume the UE immediately sends out the TCP ACK packet after receiving the TCP SYN-ACK packet without being delayed by the delayed ACK mechanism [12], which holds true in reality. Similar to the case in *DL*, we use $t_3^{'}$ to approximate t_3, and then we get *UL*$\approx t_4^{'} - t_3^{'}$.

[1]*PL* is less than 1ms in our measurement result and can be ignored.

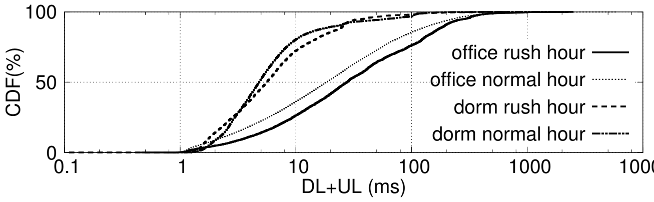


Fig. 2. Distribution of WiFi hop latency (*DL+UL*) in four different classes. We use *UL* measured using 3-way handshake to approximate the lower bound of other TCP packets' *UL*. We aggregate the whole data of two months in each class together.

D. Breaking Down RTNL for TCP Data Packets

We can actually measure *DL* for TCP data packets[2] fairly accurately as well. For data packets, *DL*$\approx t_3^{'} - t_2$, and both $t_3^{'}$ and t_2 can be measured accurately on the AP, which is the same as the TCP 3-way handshake packets.

Our definition of *WL* of regular TCP packets is the round trip *network* latency on the wired Internet between the AP and the server, and it does not include the server processing delay *S*. As the delay and congestion status of wired link is relatively stable, we argue that *WL* is primarily determined by the propagation delay between the AP and the server, thus using *WL* measured with 3-way handshake is a reasonable approximation for *WL* of regular TCP packets.

To accurately measure *UL* of regular TCP packets is challenging without the help of user-side instruments. However, we can use the *UL* measured using the nearest previous TCP 3-way handshake packet as the rough lower bound of regular

[2]In the paper, the TCP data packets used in measuring *DL* include pure DATA packets, pure DATA-ACK packets and DATA-ACK packets piggy-backed with data.

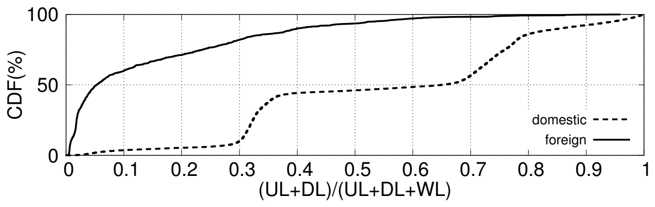


Fig. 3. Wireless part ratio in one RTNL. Wireless part ratio is calculated using $\frac{UL+DL}{UL+DL+WL}$ of all office APs' packets, including 3-way handshake packets and data packets. For *UL* and *WL* of a specific DATA packet, we use the nearest previous 3-way handshake packet's *UL* and *WL* to approximate them.

TCP packet's *UL*. This method underestimates the *UL* of regular TCP packets for the following reason. DATA packets may have larger size than 3-way handshake ACK packets[3], thus they are more likely to be interfered by other APs because their receiving time is longer than ACK packets', and have a longer *UL*. Note that the *UL* of a regular TCP packet starts when TCP hands the packets to the OS then to the 802.11 MAC, thus its *UL* does not include the delay at the UE due to delayed ACK.

III. Measurement results of Network Latency

In this section, we use our WiLy approach described in §II-C and §II-D to measure the RTNL (round trip network latency) in our deployed APs. First, we calculate the CDF to show how the WiFi hop latency looks like in the wild. Then we break down the RTNL into *WL, DL*, and *UL*. We find that there exist APs whose network latencies are dominated by WiFi hop latency. In this paper, all our measurements are done on 2.4GHz band, which has more users than 5GHz band. However, our measurement and analysis methodologies are also applicable to 5GHz band.

A. First View of WiFi Hop Latency in the Wild

Based on locations, the APs are divided into two categories: *office* AP and *dorm* AP. According to the measurement work [13] based on the EWLAN SNMP data provided by *T* campus, we define rush hour as the hour with the largest EWLAN traffic volume over the day, which indicates more potential interference around, while the rest marked as normal hours. Rush hour is from 16:00 to 17:00 and 23:00 to 24:00 for office and dorm APs, respectively. Note that, this office is the department of computer science which consists of graduate students and professors. There is no exact closing time for the graduate student's office, and people in graduate student's office will leave for dinner after 17:00 and few will come back to their work. Before they leaving for dinner, people in office tend to generate more traffic for entertainment, *e.g.*, watching video. According to [13]'s results, the overall traffic volume of office AP in rush hour is 3.5× larger than 18:00 to 19:00 and 1.7× larger than 10:00 to 11:00.

[3]We investigate the packet size distribution of uplink which shows that for about 55% uplink packets, their packet size is the same with handshake ACK (pure ACK) packet (*i.e.*, 54 bytes). The rest 45% has a larger packet size.

Using these classification schemes, we define four non-overlapping classes: **office rush hour**, **office normal hour**, **dorm rush hour** and **dorm normal hour**. Fig. 2 shows the CDF of WiFi hop latency for four classes based on the whole dataset. As expected, the WiFi hop latency is larger in rush hour than in normal hour for both *office* and *dorm* APs. Moreover, we find that *office* APs have larger WiFi hop latency than *dorm* APs because of the poor WiFi environment surrounding the *office* APs, which will be more thoroughly explained in §IV. We observe that for office APs, more than 50% of TCP packets suffer from WiFi hop latency larger than 20ms, and 10% of TCP packets have WiFi hop latency larger than 100ms. Dorm APs also have quite a long tail latency and 10% of packets' latencies are larger than 30ms.

B. CDF of Different Parts in RTNL

Using the aforementioned WiLy approach in §II-C and §II-D, we break down the RTNL into *UL, WL, DL*. We show the WiFi hop latency ratio of the overall RTNL ($\frac{UL+DL}{UL+DL+WL}$) based on the same dataset in Fig. 3. We calculate the *WL* for domestic and foreign servers. Foreign servers have much longer *WL* due to longer physical distance (thus longer propagation delay). Furthermore, we count the number of wired packets and find that packets to/from domestic servers occupy more than 98% of overall wired packets. In Fig 3, for nearly 50% of cases, WiFi hop latency occupies over 60% of RTNL of domestic services, which means that WiFi, the primary last-hop Internet access method, has become the weakest link in the round trip network latency in many cases.

IV. ANALYSIS OF WIFI FACTORS

After breaking down the RTNLs in §III, we have an overview of WiFi hop latency in the wild. In this section we mainly focus on the WiFi factors and their relationship with WiFi hop latency, and we want to answer the following questions.

- *What does the distribution of WiFi factors look like?*
- *How are WiFi factors and WiFi hop latency correlated?*
- *Which factors influence the WiFi hop latency the most?*

We first describe how to extract WiFi factors from the collected dataset. Then we calculate the CDF of WiFi factors to see the difference between dorm APs and office APs, rush hour and normal hour. Thirdly, we visualize the results to see if there exist linear relationships. At last, we use Kendall correlation and relative information gain to quantify the importance of certain WiFi factors. It is noteworthy that all the measurement results in this paper are based on our particular dataset which is collected under the environment described in §II-B. But the methodology of measurement and optimization in the following sections can be applied to other datasets.

A. WiFi Factors Statistics

We summarize all the WiFi factors collected from the APs in Table I. OpenWrt offers built-in command line tools (4th column in Table I) which enable us to collect various WiFi

TABLE I
WIFI FACTORS WE USED IN THIS PAPER. UE REPRESENTS USER EQUIPMENT ASSOCIATED ON AP (ACCESS POINT).

Abbreviation	WiFi factors	Description	Generated By
AU	airtime utilization	% of channel time used by all the traffic	iw info
Q	queue length snapshot	Number of packets queued in hardware queue.	debugfs
RR	retry ratio	%packets retried in IEEE 802.11 MAC-layer.	iw info
RSSI	RSSI	Received signal strength of UE associated on AP.	iw info
T_{tx}	transmitting throughput	Bytes sent to UE every 10s.	ifconfig info
T_{rx}	receiving throughput	Bytes received from UE every 10s.	ifconfig info
RPR	receiving physical rate	Snapshot of physical rate for receiving packets from UE.	iw info
TPR	transmitting physical rate	Snapshot of physical rate for sending packets to UE.	iw info

factors that can comprehensively reflect current network environment. All collected WiFi factors can be roughly classified into two categories: AP based information (upper part of the Table I) and AP-CLIENT pair based information (lower part of the Table I). [14] uses some of listed factors to estimate the throughput. Later we will select some of these characteristics as features to train the machine learning model.

We collect these factors every 10 seconds to lower the overhead. Note that we collect T_{rx} and T_{tx} mainly to reflect the traffic demands of AP users in 10 seconds. It is not necessary to collect T_{rx} and T_{tx} more frequently, considering the overhead to APs. For the rest of the factors, we have swept the sampling intervals from 20ms to 10s, and the error range stayed within 3% (using measurement results at finest interval 20ms as the ground truth). Thus 10s sampling interval is a reasonable tradeoff between accuracy and overhead.

B. CDF of WiFi factors

In §III-A, we show the CDF of WiFi hop latency under different temporal and spatial conditions, *i.e.*, dorm rush hour, dorm normal hour, office rush hour and office normal hour. We also calculated the CDF of these four non-overlapping classes for WiFi factors of Table I to explain why the WiFi hop latency in office is larger than dormitory. We omit the CDF figure because of the space limitation. The median *AU* of office APs is 0.45, which is larger than the median value 0.42 measured by dorm APs. We also count the number of APs in dorm and office environment. The number of APs are 120 and 24 in office and dorm environment, respectively. These numbers show that there are larger possibilities for office APs suffering from carrier sensing and hidden terminal interference. Besides, the median *RSSI* of office APs is -55dbm, which is smaller than the median *RSSI*, -46dbm, of dorm APs. *RSSI* indicates the measured signal strength of client associated on the AP. In summary, office APs have larger possibility to suffer from heavier interference and lower *RSSI* problem, which are the main causes of the higher WiFi hop latency of office APs.

C. Preliminary visualizations of correlation between WiFi factors and latency

We plot Fig. 4 to understand how WiFi factors influence the WiFi hop latency. Y axis represents the corresponding *DL* or *UL*, and X axis in each subfigure represents a certain kind of WiFi factor. We bin the factors using different intervals based on their properties: 10Mbps in *TPR*, *RPR*, 10dbm in *RSSI*, 0.1 in *RR*, *AU*, 10 in *Q* and 20Kbps in T_{rx}, T_{tx}. We use the following statistical summary indicators for each bin's latency: average, median (50th percentile) and 90th percentile. We plot the graph of *DL* with factors *AU, RR, Q, TPR*, and T_{tx} because they mainly affect *DL*. Similarly, we plot the graph of *UL* with factors *RSSI, RPR* and T_{rx} because they mainly affect *UL*. From Fig 4, we can divide the factors into two types: (1) factors which show relatively clear relationship with latency (*AU, RR, TPR, RSSI, Q, RPR*), either negative or positive. (2) factors which have no clear relationship with latency (T_{rx}, T_{tx}).

TABLE II
MEDIAN VALUE OF KENDALL SCORE AND RELATIVE INFORMATION GAIN (RIG) OF DIFFERENT APS WHICH IS MEASURED BETWEEN WIFI FACTORS AND CORRESPONDING *UL* OR *DL*.

Quality metric	Kendall Score	RIG
AU	**0.86**	0.05
$RSSI$	-0.5	0.06
RR	0.4	0.08
TPR	-0.3	**0.11**
RPR	-0.2	0.09
T_{rx}	-0.17	0.01
Q	0.15	0.007
T_{tx}	-0.006	0.02

D. Quantifying the relationship

In this part, we mainly study the Kendall correlation and Relative Information Gain which are used in [15]. By quantifying the relationship between WiFi factors and WiFi hop latency, we can get more information of the middle steps for the machine learning model in §V. The information gain also helps guide the feature selection procedure of the Decision Tree in §V.

1) Correlation: To compare among different factors, we choose Kendall correlation as a metric. This is because Kendall does not need the data obeys certain distribution, *e.g.*, Guassian distribution. After binning the value of WiFi factors using the same interval as §IV-C, we calculate the average value of corresponding *DL* or *UL* that falls into this bin. Then we compute the Kendall correlation score between *DL* or *UL* and different WiFi factors for each AP-CLIENT pair. The median Kendall scores of different pairs are shown in the second

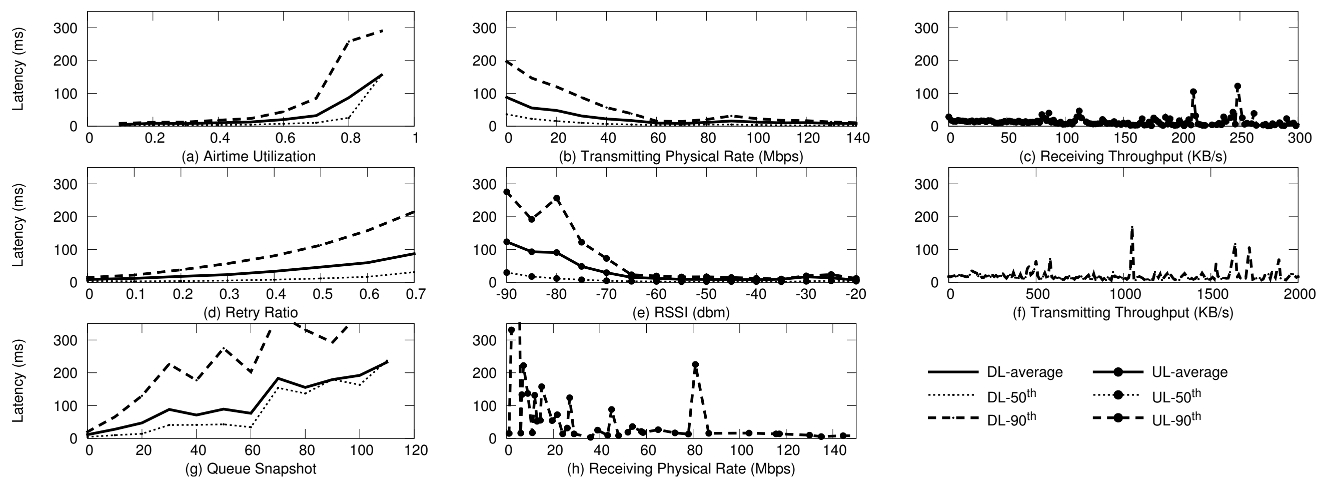


Fig. 4. Qualitative relationship between WiFi factors and latency on *DL* or *UL* aggregated over all user equipments and APs.

column Table II. We can see that *AU, RSSI* and *RR* are top 3 factors in correlation coefficient.

2) Relative Information Gain: [15] points out that correlation-based analysis cannot handle well those non-monotone relationships such as T_{rx}, *RSSI*, T_{tx}, *Q* and *RPR* in Fig. 4 (c), (e), (f), (g) and (h). We choose relative information gain, abbreviated as RIG, to solve this problem. RIG is used to represent how much help there is in predicting Y with the knowledge of X, *a.k.a.*, how much of Y's uncertainty will be reduced when knowing X's value. We first bin the value of *DL* or *UL* into discretized values using 10ms interval to form Y. For each kind of corresponding WiFi factors X_i, we calculate the relative information gain. We repeat this for different APs. The median value among different APs is shown in the third column of Table II. Table II shows *TPR* has the largest RIG 0.11. Based on the definition of RIG, RIG represents the reduction percentage of WiFi hop latency's uncertainty after knowing the value of certain WiFi factor, which ranges from 0 to 1. RIG equals to 0 if and only if WiFi hop latency is independent with certain WiFi factor. Any value of RIG which is larger than 0 shows that this WiFi factor is useful for predicting WiFi hop latency. [16] lists three reasons for *TPR*'s change: (i) a shift of RSSI, (ii) frame losses caused by interference, which related to *RR* and *AU* in our work, (iii) rate sampling algorithm which aimed to evaluate wireless channel quality. This means that *TPR* is dependent on *RSSI, RR* and *AU* and is a comprehensive metric taking *AU, RR* and *RRSI* into consideration. This also explains why *TPR* has the highest relative information gain in Table II. Note that the value will be further amplified when combining other factors in Decision Tree model in §V. Another thing we can learn from Table II is that though *AU* has the highest Kendall score, it is not the best candidate to predict *DL*. This can be explained using Fig. 4(a). *DL* increases only when *AU* is larger than 0.5. There is no obvious difference in *DL* when *AU* is smaller than 0.5. Besides, both T_{rx} and T_{tx} show little correlation with *UL* or *DL*.

E. Summary of WiFi Factors Analysis

Here we summarize the key observations obtained from this section:

- *TPR* has the highest relative information gain to predict the WiFi hop latency.
- WiFi hop latency can only be affected when *AU* exceeds a certain threshold. In our measurement, this threshold is 0.5.
- There is little correlation between WiFi hop latency and average traffic demands in 10 seconds both in terms of Kendall score and relative information gain.

V. MODELING WIFI HOP LATENCY

Because of the rate adaption method [16], *TPR* is dependent on *AU, RSSI* and *RR*, and it is a good overall indicator for the channel conditions. For many WiFi factors, such as *Q*, T_{rx} and T_{tx}, the relationship with WiFi hop latency can be non-linear. How can we predict the WiFi hop latency given all these factors? How many factors are enough to accurately predict the WiFi hop latency? To address these challenges, in this section we choose machine learning to model the relationship between WiFi factors and WiFi hop latency. We first give a formal definition of the problem that we want to solve. Then we introduce the machine learning approach to solve this problem. At last, we evaluate the effectiveness of our approach using 10-fold cross validation.

A. Problem Statement

Predicting WiFi hop latency using WiFi factors is important for home AP owners to optimize or reconfigure their APs. However, it is challenging to do so because different kinds of network factors are interdependent and affect each other. We want to jointly use the WiFi factors accessible from the home APs to predict the *DL*, *UL* and two-way latency (*i.e.*, *DL*+*UL*). Therefore, we give the problem statement as follows: *given a set of WiFi factors, predict the WiFi hop latency, for both UL and DL.*

B. Modeling Approach

Considering the complex relationship between WiFi factors in Table I and WiFi hop latency, we choose machine learning as our tool to address the above problem. It is crucial to choose the appropriate machine learning methods. An appropriate method for our problem should have the following properties:

- *The model should be accurate enough to predict the WiFi latency using the WiFi factors.*
- *The model should be easily interpretable and provide guidance for further optimization.*

We model the problem as a binary classification problem, and we label the latency into **FAST** and **SLOW** using a threshold. [3] points out that 20∼30 ms last-mile latency will cause the average page load time to exceed 2 seconds, which will cause revenue decrease [2] [17]. A back-of-envelope calculation using the above numbers shows that, WiFi hop latency should keep below a stringent threshold of 20∼30ms in order to satisfy the user expectations. Thus we set the threshold to 25ms and 12.5ms for two-way latency (*DL+UL*) and one-way latency (*DL or UL*), respectively.

After a pilot study, we choose **Decision Tree** from the Python scikit-learn package [18] as our classifier method. Decision Tree method outperforms Bayes and linear regression methods in the pilot study. Bayes and linear regression are not performing well on our dataset because the features in our dataset (*i.e.*, WiFi factors) are dependent on each other. The other reason is that the relationship between WiFi factors and latency is non-linear. However, the Decision Tree model has no assumptions about independence among features and linear relationship between features and latency.

C. Evaluation

We use 10-fold cross validation to evaluate the accuracy of the Decision Tree models. We randomly divide the whole dataset into two parts: 90% of data are used for training the models and the rest 10% are used for evaluation. For the evaluation, we first use the threshold to classify the measured latency into two classes, **FAST** and **SLOW**. We use these as the ground truth to evaluate the classification results obtained by the Decision Tree model trained by us. We use the classical ROC (Receiver Operating Characteristic) metrics to evaluate these two models. This includes the accuracy (abbreviated as **ACC**), true positive rate (abbreviated as **TP**) and false positive rate (abbreviated as **FP**). The results are summarized in Table III. We do the classification on *DL, UL, DL+UL* respectively.

Decision Tree: For Decision Tree, we exclude *Q*, T_{rx}, T_{tx} from the feature set because of their low RIG. Besides, we set the maximum tree depth as 15 and the minimum leaf size as 100. In this case, our Decision Tree achieves 78% accuracy for classifying *DL*.

The accuracy can be greatly improved by using the Random Forest model which votes between multiple trees. In our experiment, by using a Random Forest model whose tree number is 200, the accuracy can reach 81%. The accuracy and **TP** can further increase when increasing the tree number and the depth of the tree in Random Forest model. However, we do not recommend to do so because there will be huge number of trees and each tree is very large, which cannot give us insights behind the model. To give actionable insights to AP owners, we plot the pruned decision tree in Fig. 5(a). We use this model to classify *DL* into **FAST** and **SLOW**. The number in brackets represents the population size falling into each branch using the entire two month dataset. By tracking splitting nodes from root to certain leaf, we can know exactly what happened to the classified packets. It is noteworthy that the learning algorithm of Decision Tree in scikit-learn package is a greedy algorithm [18], *i.e.*, locally optimal decisions are made at each node. *TPR* is chosen as the root of decision tree in Fig. 5(a) as *TPR* has the largest overall information gain in Table II. However, the nodes in the remaining layers are decided using the information gain when combining the upper layer nodes which lying in the path up to the root. This combined information gain used to select the nodes differs from the information gain for single factor in Table II. This explains why *AU* only have the 5^{th} relative information gain in Table II but appear as the second layer of decision tree. In other words, the combination of *AU* and *TPR* act as a better split criterion than other factors.

TABLE III
ACCURACY OF CLASSIFICATION METHOD.

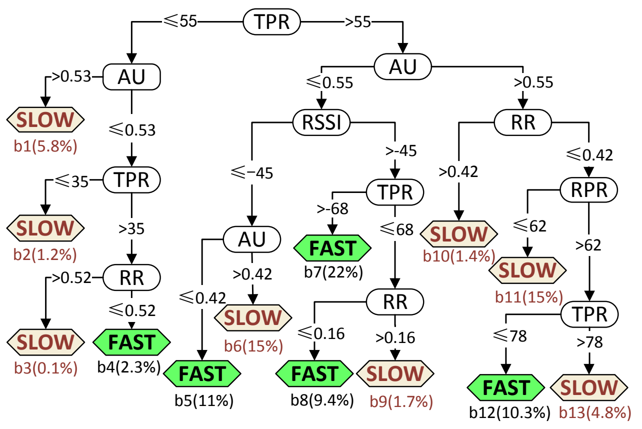
Classification				
Method	Latency Type	Accuracy	TP	FP
Decision Tree	*DL*	**0.78**	0.76	0.24
	UL	0.68	0.67	0.27
	DL+UL	0.77	0.79	0.31

VI. Optimization Enabled by the Prediction Model

In §V, we have trained a generic decision tree model using our large dataset that provides enough Wi-Fi factors dynamics, such that tree model in Fig. 5(a) can be applied to APs beyond just ours. It helps understand, troubleshoot, and optimize WiFi hop latency for APs in the wild. For a specific AP, we need to map its specific packet's WiFi factors onto the tree in Fig. 5(a), and find its specific path from the root to the leaf. We highlight two real examples to illustrate the power of this model based optimization.

A. Three Steps for Optimization

We take three steps to optimize WiFi hop latency using the generic Decision Tree model that has already been trained. Firstly, we collect packets which contain WiFi factors from the specific time and AP we want to diagnose. Then, we classify these packets using the Decision Tree in Fig. 5(a) and track the specific path to the **SLOW** leaf that has the largest fraction of packets. All the factors on this path can form a most promising candidate set to be optimized. Note that we do not need to compare the prediction results in this step with the measured delay of each packet. At last, an AP owner can attempt to



(a)

branch	b1	b2	b5	b6	b7	b10	b11	b12	b13
fraction of packets before optimization	0.1	0.8	3	**58**	0	6.7	13.4	4.2	13.8
fraction of packets after optimization	0	0	0	0.5	**61.2**	0.4	3.5	0.4	34

(b)

branch	b1	b6	b7	b10	b11	b12
fraction of packets before optimization	3	0	0	0	**82**	15
fraction of packets after optimization	0	12.6	**82**	0.9	4.5	0

(c)

Fig. 5. (a). Pruned Decision tree for classifying *DL* into **FAST** and **SLOW** categories. (b). %branches of *OAP* WiFi factors fall into before and after relocation. (c). %branches of *DAP* WiFi factors fall into before and after channel selection.

adjust the value of the factors in this set, *e.g.*, increasing the transmitting power, switching the channel, and relocating the APs. For the majority of cases, just one iteration of these three steps works well, and for the rest few, several iterations may be needed to reach a good optimization.

The three steps of optimization give the AP owners a candidate and priority list when diagnosing his AP. Without the Decision Tree model, AP owners often have no idea what will happen if they change the value of one certain WiFi factor. Our model can predict the possible WiFi hop latency by assuming the value of certain WiFi factor has changed before taking real actions.

B. Relocating the AP

The owner of an office AP (labeled as *OAP*) complained about long latency and tried several ways to reduce latency, including increasing the *OAP*'s transmitting power and switching between different channels (mainly affects *AU*), but neither of these worked. Our Decision Tree-based optimization helped solve the problem. First, we collected the packets with WiFi factors from *OAP* in 24 hours and classified the packets using the model in Fig. 5(a). We show the fraction of packets fall into each branch in Fig. 5(b)'s second row. This table informs us that most of the time (58%) *OAP* is in the **SLOW** class branch **b6**, which means that *OAP*'s *TPR* > 55 Mbps, $0.42 <$ *AU* ≤ 0.55 and *RSSI* ≤ -45 dbm.

We cannot change *TPR* manually because it can be adjusted only in the vendor chips. As *RSSI* indicates the signal strength

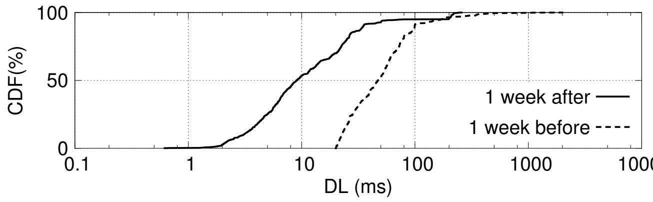


Fig. 6. CDF of *OAP DL* one week before and one week after optimization under the guidance of decision tree.

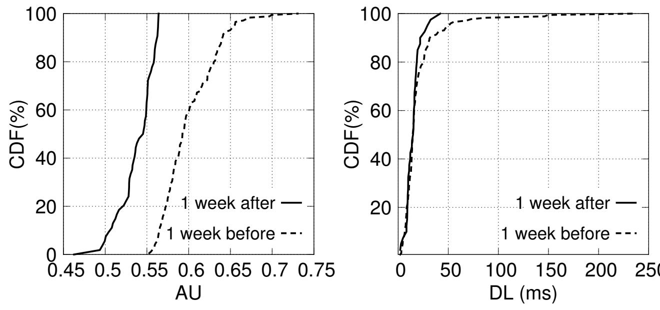


Fig. 7. CDF of *AU* and *DL* one week before and one week after the channel selection.

of user equipment (UE) received by *OAP*, it is decided by UE's transmitting power and the distance between UE and *OAP*. Increasing the transmitting power of *OAP* can not help with *RSSI* increase. We find that all the UEs are close to each other but far away from the *OAP*, which is in another room. As the UEs have to stay in a fixed desk space, we relocated *OAP* closer to the UEs, and the result is shown in Fig. 5(b)'s third row. Now most of time the packets fall into branch **b7**, the **FAST** class. We draw the CDF of *DL* before and after optimization in Fig. 6, where the median latency is reduced from 50ms to 10ms, which achieves a 5$\times$ improvement. It is noteworthy that, in both this and the next subsections, we measure *DL* before and after optimization both for one week time to eliminate the differences caused by diurnal traffic variation.

C. Channel Selection

A dorm AP (labeled as *DAP*) owner reported frequently long latency experience from 23:00 to 24:00, the rush hour of T campus dorm area. We classified the WiFi factors of packets collected from 23:00 to 24:00 to Decision Tree in Fig. 5(a). We counted the number of packets classified into each branch and calculated the fraction in Fig. 5(c). From the second row of Fig. 5(c) we notice about 82% of cases are classified into **SLOW** class branch **b11**, which means that *TPR* $>$ 55 Mbps, *AU* $>$ 0.55, *RR* $\leq$ 0.42 and *RPR* $\leq$ 62Mbps.

After detecting different channels around *DAP*, we found *DAP* currently was in channel 6. The average *AU* of channel 6 is 0.6. However, the average *AU* of channel 1 was lower (0.52). There was high possibility the *AU* of channel 1 will stay below 0.55 (one splitting condition in Decision Tree) after switching to channel 1. Also recall that *AU* starts to greatly affect latency at 0.5 according to Fig. 4(a). So we switched *DAP* to channel

1 the next day and the classification results of data from 23:00 to 24:00 are shown in Fig. 5(c)'s third row. Now about 82% of packets are falling into the **FAST** class branch **b7**. We also draw the CDF of *AU* and *DL* before and after the channel selection. The results are shown in Fig.7. From Fig.7 we can see that after channel selection, the largest *DL* (also largest in rush hour) is 50 ms compared to 250 ms before optimization. This improvement confirms the effectiveness of our optimization. Without our model, switching to a channel with only slightly lower *AU* (from 0.6 to 0.52) would not have been considered as an effective optimization.

VII. Related Work

The methodology we used to measure the WiFi hop latency is inspired by [17] and is similar to [19]. The measurements on WiFi hop latency show the consistent results with [19]. We make further improvement to separate the downlink and uplink WiFi hop latency, *i.e.*, *DL* and *UL*. Furthermore, we also use the decision tree model to predict the WiFi hop latency which shows great help in the optimization work.

WiFi Measurement. [11] measured the *DL* of WiFi network by capturing and merging packets on both wired and wireless interfaces. This will fail when the home APs in the wild encrypt the wireless packets using the SSID password. We make it practical using a minor modification in the driver of OpenWrt. Xian Chen's work [20] measured the WiFi latency by capturing the outgoing and incoming packets at the campus gateway router. It cannot measure the latency of single AP-CLIENT pair behind the home AP. [14], [16] and [21] deploy routers in different places and do the exhaustive measurements but they are all lack of investigation on WiFi hop latency.

Optimization Guidance. [15], [22] use the machine learning methods to study the relationships between mobile Web/Video QoE with cellular network factors or video quality respectively, which differ from the WiFi factors in residential WiFi environment.

VIII. Conclusion

Based on measurement results in a 47-AP real world deployment, this paper provides a first look into WiFi hop latency in the wild. This paper presents the first practical approach to break down round trip network latency on APs, without any user-side assistance or vendor chip support. Furthermore, we propose a Decision Tree based model to help understand, troubleshoot, and optimize WiFi hop latency for APs in the wild beyond just ours.

Our measurement results show that, WiFi, the primary last-hop Internet access method, has become the weakest link in the round trip network latency in many cases. This shows the urgency for the research community to spend more efforts on WiFi hop latency. In this paper, we conduct the systematic work to understand the WiFi hop latency in the wild. We believe it is a significant step to go from fundamental measurements to fully automated optimization of WiFi hop latency.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. We appreciate the help of Mengyu Zhou and Lab μ in distributing the APs. We thank Renata Teixeira for her valuable comments. We sincerely thank Kaixin Sui and Dapeng Liu for providing drawing template of decision tree. We thank Yaodong Zhang for helping check technical details in this paper. At last, we thank Yousef Azzabi, Xiaohui Nie, Yongqian Sun, Juexing Liao for their suggestions and proofreading.

This work has been supported by two National Natural Science Foundations of China (NSFC) under Grant No.61472210 and No.61472214, the Sate Key Program of National Science of China under Grant No.61233007, the National Key Basic Research Program of China (973 program) under Grant No.2013CB329105, the National High Technology Development program of China (863 program) under Grant No.2013AA013302.

References

[1] Cisco. Cisco visual networking index: Forecast and methodology, 2013–2018. *CISCO White paper*, pages 2013–2018, 2013.
[2] Akamai study. http://goo.gl/2pwozG.
[3] S. Sundaresan, N. Feamster, R. Teixeira, N. Magharei, et al. Measuring and mitigating web performance bottlenecks in broadband access networks. In *ACM Internet Measurement Conference*, 2013.
[4] EXFO. http://goo.gl/OMGKnE. Accessed: 2016-01-20.
[5] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *Network, IEEE*, 1997.
[6] Barrier Breaker. https://openwrt.org/. Accessed: 2016-01-20.
[7] Libpcap. http://en.wikipedia.org/wiki/Pcap. Accessed: 2016-01-20.
[8] Rsync. https://rsync.samba.org/. Accessed: 2016-01-20.
[9] Mustafa Ergen. Ieee 802.11 tutorial. *University of California Berkeley*, 70, 2002.
[10] Z. Tang, Z. Yang, J. He, and Y. Liu. Impact of bit errors on the performance of dcf for wireless lan. In *ICCCAS*. IEEE, 2002.
[11] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benkö, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker. *Automating cross-layer diagnosis of enterprise wireless networks*, volume 37. ACM, 2007.
[12] Robert Braden. Rfc-1122: Requirements for internet hosts. *Request for Comments*, pages 356–363, 1989.
[13] K. Sui, Y. Zhao, D. Pei, and Z. Li. How bad are the rogues impact on enterprise 802.11 network performance? In *INFOCOM*, 2015.
[14] A. Patro, S. Govindan, and S. Banerjee. Observing home wireless experience through wifi aps. In *MobiCom*, pages 339–350. ACM, 2013.
[15] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. *SIGCOMM*, 41(4):362–373, 2011.
[16] P. Ioannis, L. Henrik, S. Augustin, C. Jaideep, L. G. Pascal, D. Christophe, M. Martin, V. D. Karel, and V. O. Koen. Characterizing home wireless performance: The gateway view. In *INFOCOM*, 2015.
[17] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang. A provider-side view of web search response time. *SIGCOMM*, 43(4):243–254, 2013.
[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.
[19] S. Sundaresan, N. Feamster, and R. Teixeira. Measuring the performance of user traffic in home wireless networks. In *Passive and Active Measurement*, pages 305–317. Springer, 2015.
[20] X. Chen, R. Jin, K. Suh, B. Wang, and W. Wei. Network performance of smart mobile handhelds in a university campus wifi network. In *IMC*, pages 315–328. ACM, 2012.
[21] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, B. Ravi, and N. Feamster. Peeking behind the nat: an empirical study of home networks. In *IMC*, pages 377–390. ACM, 2013.
[22] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan. Modeling web quality-of-experience on cellular networks. In *MobiCom*, pages 213–224. ACM, 2014.