How Much Are Your Neighbors Interfering with Your WiFi Delay?

Changhua Pei[†], Youjian Zhao[†], Guo Chen[§], Yuan Meng[†], Yang Liu[‡], Ya Su[†],
Yaodong Zhang[†], Ruming Tang[†], Dan Pei[†*]
[†]Tsinghua University [§]Microsoft Research Asia [‡]University of Illinois Urbana-Champaign
[†]Tsinghua National Laboratory for Information Science and Technology (TNList)

***Abstract*—Previous studies have shown the WiFi, as the dominant last hop access to Internet, has become the weakest link in the round-trip network delay. Therefore it is critical to understand and minimize the WiFi interference in order to reduce the WiFi hop delay. For the first time in the literature, this paper defines an intuitive and accurate metric to quantify the impact of interference on each actual packet. For each packet traveling through the access point, it measures the percentage of MAC layer delay wasted due to neighbor APs' interference. This metric is defined based on a packet's various (measured or inferred) timestamps and can be measured with a small kernel modification on a commodity AP with little overhead. Our 29-AP two-month measurement results in the wild show that this metric is a strong indicator of interference's impact on WiFi hop delay. Using this metric as input, distributed channel selection on individual APs reduces the median WiFi hop delay by up to 5×. Collaborative optimization on multiple APs reduces the overall WiFi hop delay by 5× compared to the default channel.**

I. Introduction

Recently, it has been a very active research topic to reduce the delays throughout the end-to-end Internet paths, from the last hop [1] to last mile [2] to backbone [3], [4] to data centers [5]–[9]. This is because the end-to-end application delay highly affects the user-perceived quality of experience (QoE), which in turn impacts Internet company's revenue. For example, 100ms to 500ms delay will cause 0.74% to 1.2% revenue loss in Amazon, Bing, and Google [10]. In addition, many popular interactive applications (Web browsing, voice chatting, video chatting, instant messaging, *etc.*) are very sensitive to delay. One key observation by recent studies is that network layer delay [2], [10] can be amplified by up to 100 times in application layer delay. For example, Web page load time can increase thousands of ms when the last hop delay increases tens of ms [2]. Therefore it is very important to reduce network layer delay [10].

Along the end-to-end path from last hop to the data centers, WiFi, as the primary last hop for Internet access [11], [12], has become the weakest link in terms of network layer delay according to the recent studies [1], [13]. For example, In [1], they observe that 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. The primary reason is due to the chaotic deployment of residential APs and rogue APs around enterprise WiFi networks. These WiFi APs are autonomously managed or not managed at all, and do not coordinate with each other, in sharp contrast to the well-engineered last-mile, backbone and data center networks. Yet, more and more WiFi devices come online and compete for the same limited wireless spectrum, and the interference is the usual suspect for the poor WiFi hop latency [1], [13]. Therefore, there is a great need for minimizing the WiFi interference in order to achieve a low WiFi hop delay that matches the QoE requirement of modern Internet.

There are many existing approaches for minimizing the WiFi interference, but unfortunately they all fall short when faced with above delay requirements and the autonomously managed APs. On the one hand, there are some approaches [14]–[16] that try to centrally schedule packet transmission at microsecond granularity among multiple APs, but these approaches are impractical when coordinating APs that might be milliseconds away from each other. On the other hand, the distributed channel selection algorithms such as LCCS [17] commonly use metrics such as airtime utilization, which cannot capture the fine-grained dynamics of packet level WiFi interference. Similar for active measurement based approaches [18].

We use a real trace in Fig. 1 to illustrate why. We deployed an AP in a graduate student lab, where all clients have good RSSI. The detailed measurement methodology will be explained in detail later in the paper. The MAC layer delay experienced by the downlink WiFi packets (upper panel), the airtime utilization and packet loss ratio (bottom panel) are all averaged every minute and shown as time series. We can observe that the MAC layer delay, and the packet loss ratio all vary a lot over time. The MAC layer delay even varies from a few milliseconds to more than 100ms. While the airtime utilization can capture the coarse-grained probability for interference, it apparently cannot capture the packet or minute level inference and its impact. For example, the middle vertical line ① in Fig. 1 shows a case where high loss ratio in the RR curve causes the delay to spike. These real examples show that airtime utilization or active measurement sampling cannot capture the fine-grained dynamics of interference.

Therefore, our problem statement is to *quantify the the **time-variant** and **empirical** impact of interference on the delay experienced by the user's actual individual packets, and then take a data-driven approach for minimizing the interference*

* Dan Pei is the corresponding author.

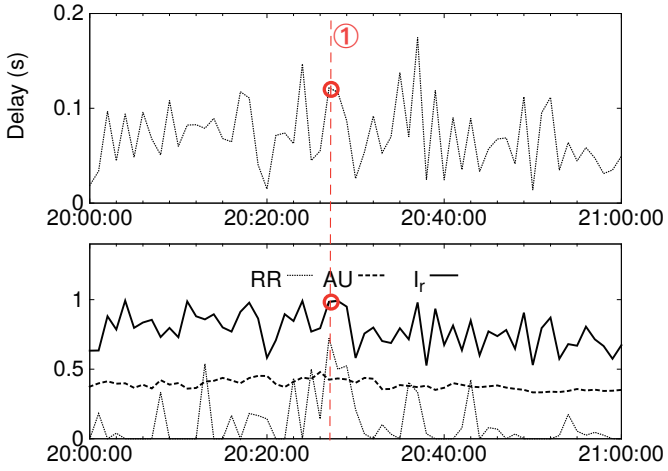


Fig. 1: The minutely average time series on an access point from 8:00pm to 9:00pm on Dec.1^{st}. I_r is the metric we proposed to represent the impact of interference on delay. AU is short for airtime utilization. RR is short for the retry ratio (or loss ratio). Delay is short for the MAC layer delay.

and improving the packet level WiFi delay. It faces two major challenges. First, we have to define an accurate metric, since there is no existing such metric. Second, such a metric should be able to be passively measured with low overhead for each packet on commodity AP devices to maximize deployment opportunities. A commodity AP device typically has only one NIC per frequency band and cannot have a dedicated sniffer NIC such as those used in [18]–[20].

This paper proposes a novel approach which solves the problem defined above and addresses its two challenges. We call our methodology *WING* (*W*iFi *IN*terference *G*raph), and an AP with *WING* deployed is a *WING* AP (or *wAP* for short hereinafter). Our core idea is to define a metric $I(p)$ which quantifies the percentage of MAC layer delay *wasted* by each packet p on a wAP due to neighbor APs' interference, including waiting time for channel contention or retransmission due to packet loss. To passively measure this metric, we make a small kernel modification such that the critical path timestamps of packet p can be tagged by the *wAP* to the packet p. Then $I(p)$ can be accurately calculated based on the timestamps that are directly measured or inferred. The metric $I(p)$ can be aggregated into I_γ for time period γ for different channels and serve as the input for distributed channel selection algorithm [17] so that an individual *wAP* can select its best channel.

Our major contributions are summarized as follows.

First, for the first time in the literature, in §II we define an intuitive and accurate metric $I(p)$ to quantify the impact of interference on each actual packet. This metric is defined based on a packet's various timestamps. This metric can be measured with a small kernel modification on a commodity AP with little overhead (§III), to maximize the deployment opportunities. Our measurement results in the wild (§IV) show that this metric is a strong indicator of WiFi hop delay.

Second, in §IV we conducted the first measurement study on interference-induced wasted time of packets in the wild. We distribute 29 APs which equipped with *WING* method to different students in campus to let them use the APs as their primary access to the Internet for 2 months. Our measurement show that in some cases the 90^{th} percentile of wasted time can reach 50ms and for more than 50% of packets, 80% of the time are wasted because of interference.

Third, we use I_γ to guide individual *wAP* to select the channel in §V. We use I_γ and airtime utilization as two different inputs of LCCS [17] algorithm. Our real-world deployment results show that I_γ consistently perform better than airtime utilization, and sometimes 5 times better. This improves shows the power of time-variant and empirical interference metric.

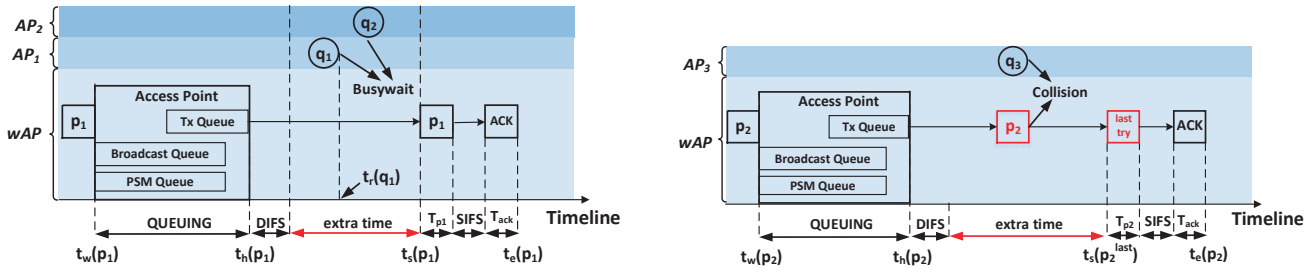
Fourth, in §VI we extend our definition I_γ to I_γ^k such that it can distinguish different neighboring interfering APs. The idea is very intuitive: the wasted time of packet p is attributed to a neighbor AP k if the *wAP* overhears a packet from k while p is waiting for the channel to be cleared. I_γ^k can then be used to build an interference graph, which is a key enabler for coordinated channel/power optimization in a SDN fashion [21]–[25]. In this paper, we use a 4-*wAP* testbed driven by real packet traces to show the potential for such coordinated optimization. The delay of *wAP* can be reduced by up to 5× compared to the default channel, and our approach can converges much faster ($O(1)$) than exhaustive channel search $O(3^n)$ for a topology with n APs.

II. Quantifying the Interference's Impact on Delay

In this section, we first give the model to break down the lifetime of packets in *wAP*. Then we summarize the critical path delays of one certain packet. Based on the critical patch delay, we give the definition of I_γ and equation to calculate it. At last, we detailedly discuss two key techniques which are used when calculating I_γ.

A. Lifetime of One Packet

We use an example in Fig. 2 to show how to measure the lifetime of one packet. We track one packet (denoted by a square) which is sent by *wAP* to its client to reveal the critical path delay of this packet. Fig. 2(a) shows the carrier sensing scenario. Fig. 2(b) shows the packet collision scenario. The corresponding notations are listed in Table I. $t_w(p_i)$ represents the time when packet p_i arrives at the wired interface of *wAP*. $t_s(p_i)$ represents the time when the first byte of packet p_i is sent into the air. $t_e(p_i)$ is the time when p_i is successfully sent by *wAP* and acknowledged in MAC layer, and removed from the *wAP* 's transmission queue. $t_r(q_i)$ is the time when *wAP* hears packet q_i, where q_i is the overheard packet of other APs. The rest of the notations listed in Table I will be explained in the following text when we trace the life time of packet p_1 and p_2. The packet p_1 may experience the following parts of delay when it enters *wAP* from the Internet at time $t_w(p_1)$.



(a) Carrier sensing scenario.

(b) Packet collision scenario.

Fig. 2: The lifetime of one packet from AP to client.

Variables	Notations	Directly Measured
$t_w(p)$	The time when packet p arrives at AP.	Yes
QUEUING	Time spent on buffering in the Queue.	No
$t_h(p)$	The time when packet p arrives at head of Tx Queue and ready to be sent.	No
$DIFS$	Constants. *E.g.*, for 802.11n protocol, DIFS=28us.	Yes
$t_r(q)$	The time when receiving the neighbor AP's packet q.	Yes
$t_s(p)$	The time when the first byte of non-retrying packet p is sent.	No
$t_s(p^{last})$	The time when the first byte of retrying packet p's last retry is sent.	No
T_p	The transmission time of packet p.	Yes
$SIFS$	Constants. *E.g.*, for 802.11n protocol, SIFS=10us.	Yes
T_{ack}	Constants. *E.g.*, for 802.11n protocol, $T_{ack} = 44us$.	Yes
$t_e(p)$	The time when p is ACKed. For a retried packet, it's the ACK time of the last retry.	Yes

TABLE I: Notations used in Figure 2.

Reaching the head of Tx Queue: For the broadcast or multicast packets, such as beacons, AP will send them into the Broadcast Queue and flush them into Tx Queue at the next beacon time. For Power Saving Mode (PSM) [20] packets, they will be buffered in PSM Queue until receiving the next delivery traffic indication message (DTIM). Note that in this paper, we only care about the unicast packet to a certain client. After queuing in the Tx Queue, p_1 arrives at the head of the *wAP* 's Tx Queue at $t_h(p_1)$ and is ready to be sent. $t_h(p_1)$ is also the time when *wAP* starts accessing the channel.

Competing for the channel and sending the first byte: When p_i arrives at the head of Tx Queue, *wAP* first waits for a small period of time (DIFS). Then *wAP* detects whether the channel is busy or not[1]. If the channel is free, *wAP* sends p_1 out directly. If the channel is busy, *wAP* waits for a timer $\mathcal{T}$ which is randomly set to be $\mathcal{T} = b_0 * slot\_time$. b_0 is a random number that ranges in [0, $aCWmin$]. $aCWmin$ is the minimum random window size. $slot\_time$ and $aCWmin$ vary with different IEEE802.11 protocols. For 802.11n protocol, $slot\_time$ is 9 or 20us in 2.4GHz and $aCWmin$ is 15. After *wAP* setting the timer $\mathcal{T}$, it continuously senses the channel and decreases $\mathcal{T}$ when the channel is sensed as free. However, when *wAP* finds channel is busy (*e.g.*, *wAP* senses that another packet is being transmitted, denoted by circles in Fig. 2(a)), the timer $\mathcal{T}$ will be frozen until the channel is sensed to be free again. *wAP* sends p_1 when $\mathcal{T}$ is decreased to 0. In Fig. 2(a), $t_s(p_1)$ is the moment when the first byte of p_1 is sent, also when the timer $\mathcal{T}$ is decreased to 0. Time interval $t_s(p_1)$ - $t_h(p_1)$ varies according to the carrier sensing interference. If there is no other AP occupying the channel, $t_s(p_1) - t_h(p_1) = DIFS$.

Transmitted and waiting for ACK: It will take time to send p_1, *i.e.*, , the transmission time T_{p_1}. After p_1 is sent out into the air, *wAP* waits for the return of MAC-layer acknowledgement from the client which indicates the successful receipt of p_1. Theoretically, as soon as *wAP* sends the last byte of p_1 into the air, *wAP* 's client can receive and decode the whole packet p_1, because the propagation time of the signal from *wAP* to its client is very short. After 10us (SIFS which is listed in Table I), the client should send the first byte of acknowledgement into the air.

After *wAP* receives the acknowledgement from the client, it means p_1 has been successfully delivered to its client. Then *wAP* removes p_1 away from the Tx Queue and p_1 ends its life in *wAP* . We label this time as $t_e(p_1)$.

Retrying: If *wAP* waits for an ACK-timeout time and still has not received the acknowledgement from the client, it regards the packet as lost and begins retrying. Packet loss is mainly caused by collision, *e.g.*, hidden terminal interference, which is shown in Fig. 2(b).

We use another packet p_2 in Fig. 2(b) to explain the retrying procedure. After getting the access to the channel, AP sends p_2 out and waits for the ACK from its client. Based on 802.11 protocol [26], once *wAP* occupies the channel, all the other APs cannot send packets until *wAP* receives ACK from its client. However, if *wAP* is out of the carrier sensing range

[1]Based on 802.11 protocol, the NIC of *wAP* can detect whether the received power strength is above *Carrier Sense Threshold* (CST) or not. If the Received Signal Strength Indication (RSSI) is above CST, *wAP* determines that the channel is busy.

of another AP (AP_k), AP_k regards the channel as free while *wAP* is sending p_2 and sends packet q_3 at the same time. As a result, the client of *wAP* will receive two overlapping signals and cannot decode the packet at the client-side, *i.e.*, collision happens.

When the collision happens, *wAP* will not receive ACK before the ACK-timeout. Then *wAP* will retransmit p_2 until it is acknowledged by its client or the retry counts exceed the maximum threshold. The retrying of p_2 is the same with the first transmission of p_2 which is explained before. The only difference is that the $aCWmin$ will be doubled after each retrying.

It is noteworthy that low RSSI can also cause packet loss, but in this paper, we focus on the delay problem caused by *interference*. Thus we continuously monitor the RSSI of client at AP side, and only study those clients with strong RSSI and filter those ones with low RSSI.

B. Definition and Calculation of I_γ

After tracing the lifetime of packet p_1 and detailedly breaking down its WiFi lifetime, we can calculate the "wasted" time due to interference for a certain packet. For easy reference, we listed the corresponding notations used for calculation in Table II. In Table II, we summarize $d_{mac}(p_i)$ as the overall WiFi MAC-layer delay. $d^i_{mac}(p_i)$ is the "wasted" time caused by interference.

Here we first give the formal definition of the interference metric $I(p_i)$ and I_γ:

$I(p_i)$ is the metric to measure the extent of Interference on packet p_i's WiFi MAC-layer delay. I_γ is the metric to measure the impact of Interference on wAP's WiFi MAC-layer delay in time interval γ.

The calculation of these two metrics can be conceptually expressed using Equation 1 and Equation 2. $I(p_i)$ is the ratio of "wasted time" to the overall WiFi MAC-layer delay of packet p_i. I_γ is the average of $I(p_i)$ of over all packets $p_i, (i = 1, ..., n)$ whose completion time is within the time interval γ.

$$I(p_i) = \frac{d^i_{mac}(p_i)}{d_{mac}(p_i)} = \frac{t_e(p_i) - t_h(p_i) - T_{p_i} - \mathcal{C}}{t_e(p_i) - t_h(p_i)} \quad (1)$$

$$I_\gamma = Average(I(p_i)),\ p_i\ completed\ in\ \gamma \quad (2)$$

All the notations are summarized in Table II. The reason why we use $d_{mac} = t_e(p_i) - t_h(p_i)$ in Table II to calculate I_{p_i} instead of $t_e(p_i) - t_w(p_i)$ is that $t_e(p_i) - t_w(p_i)$ includes the queuing time of p_i, which is shown in Fig. 2. Packet p_i will be blocked in the Tx Queue if the previous packet p_{i-1} takes too long to be sent out. Thus $t_e(p_i) - t_w(p_i)$ includes the cumulative influence of interference on all the previous packet waiting in the queue. In order to measure the impact of interference on each packet independently, we use $t_h(p_i)$ to exclude the queuing time of packet p_i.

Latency	Calculation	Notations
$d^i_{mac}(p_i)$	$t_e(p_i) - t_h(p_i)$ $-T_{p_i} - \mathcal{C}$	The part of "wasted" time caused by interference.
$d_{mac}(p_i)$	$t_e(p_i) - t_h(p_i)$	The overall WiFi MAC-layer delay.
T_{p_i}	$size(p_i)/rate(p_i)$	Sending time of p_i.
$\mathcal{C}$	$DIFS + SIFS$ $+T_{ack}$	Fixed time each packet spends which contributed by 802.11 protocol.

TABLE II: Notations of different kinds of latency, each value is for one packet p_i. The bottom part of the table is the essential variables needed to calculate d_{mac} and d^i_{mac}.

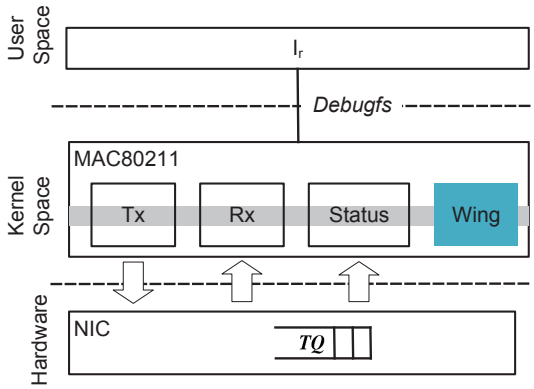


Fig. 3: The *WING* kernel measurement structure.

C. Approximating timestamp $t_h(p_i)$

$t_h(p_i)$ denotes the timestamp when packet p_i is ready to be sent after arriving at the head of Tx Queue (labeled as ***TQ*** for brevity). The main challenge to get $t_h(p_i)$ is that ***TQ*** is in physical layer and there is no information about the change of ***TQ*** from the kernel of *wAP*. Thus we use the following method to infer the behavior of ***TQ***. Once a packet is delivered to the physical layer by *wAP* , it will be appended to the tail of ***TQ***. For a unicast packet, this time is labeled as $t_w(p_i)$. Once a packet p_i is successfully sent by *wAP* to its client, it will be removed from the ***TQ***, which is labeled as $t_e(p_i)$ in Table I.

The time of p_i's arrival at the head of ***TQ*** (*i.e.*, $t_h(p_i)$) is decided by two aspects: when p_i enters the ***TQ*** and when the previous packet (p_{i-1}) leaves the ***TQ***. If p_i is blocked by p_{i-1} when it arrives, *i.e.*, $t_w(p_i) < t_e(p_{i-1})$, then p_i will reach the head of ***TQ*** as soon as p_{i-1} is removed from ***TQ***. If p_i arrives at ***TQ*** and finds ***TQ*** being empty, *i.e.*, $t_w(p_i) > t_e(p_{i-1})$, p_i will directly reach the head of ***TQ***, *i.e.*, $t_h(p_i) = t_w(p_i)$. In summary, $t_h(p_i)$ can be represented using Equation 3.

$$t_h(p_i) = max\{t_w(p_i), t_e(p_{i-1})\}. \quad (3)$$

Here $t_w(p_i)$ is the time when packet p_i arrives the ***TQ***. $t_e(p_{i-1})$ is the time when the previous packet is removed from the ***TQ***. For ***AGGREGATION*** packets, multiple MPDUs form one large A-MPDU and only need to compete for the channel once. In *WING* system, we regard different MPDUs in a same A-MPDU as one and calculate the WiFi hop delay as a whole.

III. System Implementation

A. Implementation

WING is implemented as an AP-side kernel module as shown in Fig. 3. Our *WING* kernel measurement is located in the linux wireless driver layer. The **MAC80211** module [27], a middle layer between the hardware layer and the user space layer, has three components. T_x module is responsible for handing a packet to the physical layer when this packet is transmitted to the WiFi client; R_x module is responsible for receiving the incoming packets from each wireless network device. $Status$ module is a callback function, called by each successful transmission of a packet in the physical layer.

The *WING* kernel measurement is performed by monitoring the timestamps of a packet going through the above three components. First, each packet p_i entering T_x module is tagged with a timestamp, approximating the time $t_w(p_i)$ when p_i enters the AP from the wire interface [2]. $t_w(p_i)$ will be kept along with p_i until the corresponding acknowledged (p_i's ack) is received from the client. Then p_i will be delivered to the $Status$ module, where the packet p_i will be timestamped as $t_e(p_i)$.

In order to sniff the neighboring packets, we virtualize a wireless network device "mon0" working in the monitor mode in R_x module. The command to create "mon0" is shown as follows.

```
root@Openwrt# iw dev wlan0 add interface mon0
     type monitor flags none
root@Openwrt# ifconfig mon0 up
```

"mon0" can sniff a packet q_i from *wAP* 's neighboring AP[3], and the sniffed timestamp is labeled as $t_r(q_i)$. Besides, we can also get the packet size, the transmission rate and RSSI (received signal strength indicator) for both *wAP* 's packet p_i and its neighboring AP's packet q_i. Note that *wAP* timestamps its own packets p_i and the neighboring AP's packet q_i with the same system clock. Therefore, there is no need for time synchronization. Such technique is different from the other methods requiring extra sniffers such as WISE [18] and PIE [16]. Overall, our *WING* method enables us to conduct passive measurement on a single COTS APs and makes it possible for large scale deployment for the first time.

B. Measurement overhead

We implemented *WING* as a software package on one OpenWrt-based AP (Netgear 4300), which has a 0.6 GHz CPU and 128 MB memory. In our stress test in which we have constant download throughput of at least 10MBps going through the *wAP*, the CPU utilization remains below 60%. For the *wAP* which is deployed in a CS graduate student lab, with a relative high density of wireless devices and APs, and high user traffic, the CPU cost remained under 20% while the average CPU usage was 12%. The usage of memory was stable at around 22% since they were mostly used for buffer of packet capture. Overall, the extra CPU, memory and bandwidth overheads introduced by *WING* are all insignificant on this hardware model.

[2]The time spent on traveling from the wired interface to T_x function consist of processing delay of operating system, and can be neglected.

[3]As we only need to extract the MAC addresses from neighboring packets, *WING* can also work in non open-network.

IV. Measurement in the Wild

We distribute 29 APs equipped with *WING* system in different places at campus, covering multiple dormitory and department buildings. We let students use these APs as their primary access to the Internet. We continuously measure I_γ, airtime utilization and d^i_{mac} to see the severity of the interference in the wild.

d^i_{mac} is the "wasted" time in WiFi MAC-layer delay caused by interference. Our distributed *wAPs* can be classified into two different categories: *wAPs* in dormitory and *wAPs* in department building. Department buildings have more access points as well as larger trunk of traffic, *i.e.*, the airtime utilization there is higher than that in dormitories. We choose one representative dormitory *wAP* and one representative department *wAP* to draw the CDF in Fig. 4, based on measurements lasting for one month. From Fig. 4(c) we can see that almost every packet in department (over 99%) wastes at least 100 microseconds when sending. The 90^{th} percentile d^i_{mac} exceeds 50ms in the department. The 50^{th} percentile d^i_{mac} of department is $10\times$ larger than dormitory, *i.e.*, 2000us vs 200us. This is mainly caused by the larger potential interference in department buildings, which is confirmed by the CDF of airtime utilization in Fig. 4(a).

Besides d^i_{mac}, we also draw the CDF of I_γ in Fig. 4(b). I_γ is the ratio of wasted time d^i_{mac} to the overall WIFi mac-layer time d_{mac}. Fig. 4(b) shows that in the dormitory, half of the time are wasted because of interference for over 50% packets, *i.e.*, $I_\gamma \leq 0.5$. In the department, this number increases to about 80%.

V. Optimization in Reality

Channel selection is the most effective and simplest way to improve an AP's performance under interference condition [25]. In practice, *Least Congestion Channel Search* (LCCS) [17] is the most popular channel selection algorithm used by single AP. The main idea of this algorithm is that each AP independently scans for different channels and selects *the least congested channel* to use. There may be different metrics to describe *the least congested channel*, the most commonly used in practice is the smallest airtime utilization (labeled as AU). However, none of these metrics directly reflects the impact of interference on WiFi hop delay. We show in this section that using *WING* 's measurement of *delay-oriented, time-variant, empirical* WiFi interference, *i.e.*, I_r, LCCS can achieve a much better optimization on delay performance than using the AU.

A. Optimization Environment

We picked 7 most active *wAPs* among our 29 deployed ones, and conducted the optimization on these 7 *wAPs*. They are located at different students' dormitory buildings. To ensure

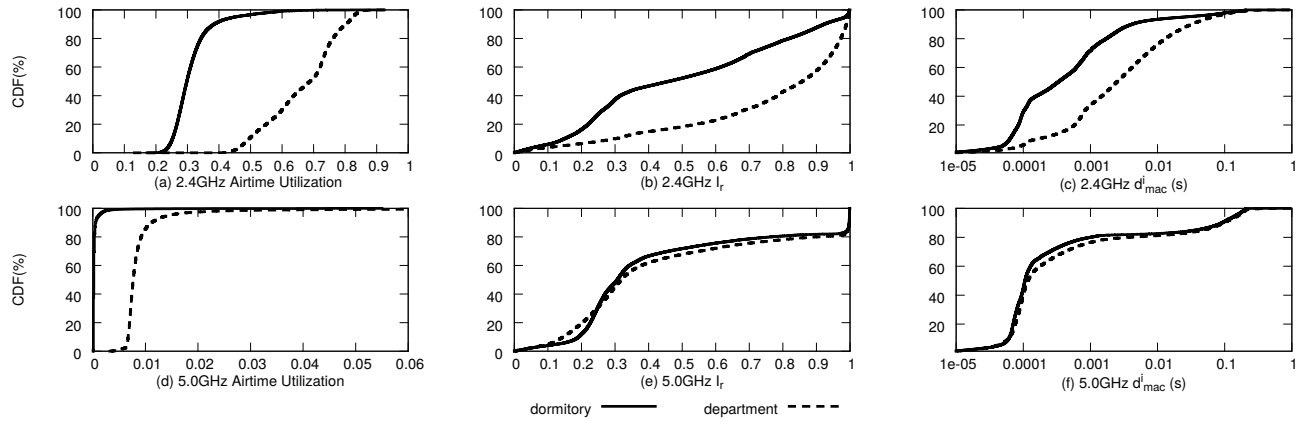


Fig. 4: The CDF of different metrics of a department *wAP* and a dormitory *wAP* both on 2.4GHz and 5GHz.

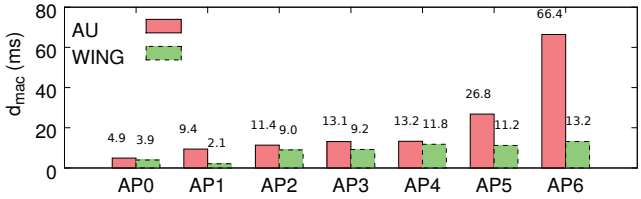


Fig. 5: The average WiFi hop delay under different dynamic channel selection algorithms on 7 deployed APs for two weeks. *AU* represents the LCCS algorithm using airtime utilization as input. *WING* represents the LCCS algorithm using our *WING*'s measurement results as input.

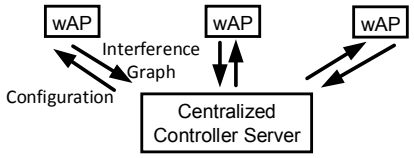


Fig. 6: The architecture of upgraded *WING*.

the fairness of comparison between two different algorithms, we run one certain algorithm for the same day (*i.e.*, AU or *WING*) across two different weeks. This is because there are weekly pattern for our deployed *wAP* in school. In each day, the corresponding algorithm either measuring the current channel or switch to another channel. Each channel measuring lasts for a half hour. After three orthogonal channels are all measured, the LCCS algorithm runs on the *wAP* will decide which channel is the best one based on the different metrics, such as smallest airtime utilization, or smallest I_γ measured by *WING*. After the decision is made, the *wAP* stays on the channel for a half hour. After a half hour, the LCCS algorithm will restart another round of optimization: scanning each orthogonal channel for a half hour and making new decision based on the latest measurement results. We use packet level latency, *i.e.*, d_{mac} to evaluate the performance of WiFi hop delay for each algorithm.

B. Results

Fig. 5 shows the average WiFi hop delay for different algorithms (*i.e.*, using different metrics as input for LCCS to select channel). We can see that our *WING* metric, *i.e.*, *delay-oriented, time-variant, empirical* WiFi interference I_r, outperforms AU consistently for all the *wAPs*. The largest reduction ratio of WiFi hop delay is 5× when compared with AU algorithm in AP6. This better performance compared with AU comes from that I_γ is a direct metric reflecting the instant impact of interference on packet level WiFi hop delay. However, AU, which records the WiFi airtime utilization, is an indirect metric for interference and is often inaccurate in quantifying the impact of interference on WiFi hop delay.

VI. Upgrading *WING*: Power of Interference Graph

Our *WING* method can also be extended to help collaborative channel selection of multiple APs, by quantifying the impact of *different interferers*. In this section, we first introduce interference graph (labeled as ***IG*** hereinafter), which is the more fine-grained metric of the current I_γ. Then we briefly introduce the methodology of measuring ***IG***. At last, we use a testbed consisting of 4 *wAPs* to show the power of upgraded *WING* , which utilizes ***IG*** to conduct collaborative optimization. Our experiment results show that our method performs much better than the default channel selection algorithm, and reaches near-optimal only needing a very short *constant convergence time*.

A. Definition of *IG*

Although the overall interference metric I_γ shows great potential in helping single AP to select the channel (§V), when there are multiple APs which change and select channel independently, it may become unstable and can not converge quickly. It will be greatly helpful if we can measure the interference of each certain neighbor AP to a *wAP* and collect the measured results together to make the decision centrally, which is shown in Fig. 6. Luckily, by extending

the measurement methodology we present before, we can distinguish different neighbor APs and measure the extent of the interference. Formally, we define a new metric called **interference graph** (***IG***) as shown in Table III. ***IG*** is a table in which each column belongs to one single interferer (denoted as AP_i). Each row belongs to one time interval (denoted as γ_i). Each element in ***IG*** (denoted as $I^k_{\gamma_i}$), represents AP_k's quantified interference to *wAP*'s delay, in the time interval γ_i.

TABLE III: ***IG***

	AP_1	AP_2	...	AP_n
γ_1	$I^1_{\gamma_1}$	$I^2_{\gamma_1}$	...	$I^n_{\gamma_1}$
γ_2	$I^1_{\gamma_2}$	$I^2_{\gamma_2}$	...	$I^n_{\gamma_2}$
...				
γ_m	$I^1_{\gamma_m}$	$I^2_{\gamma_m}$	...	$I^n_{\gamma_m}$

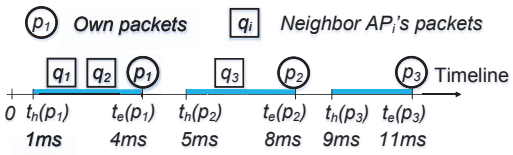
B. How to calculate *IG*

In this paper, we define $I^k_{\gamma_i}$ as the ratio of the wasted time caused by interferer *k to* the overall WiFi MAC-layer delay. For example, *wAP* spends 2ms to send a packet out from sensing the channel to being acknowledged by its client, in which 1ms is spent on waiting for the AP_k to finish its sending. We regard this 1ms time as wasted time caused by the interference of AP_k. Then $I^k_{\gamma_i}$ should be calculated using $1ms/2ms = 0.5$. More generally, $I^k_{\gamma_i}$ can be calculated using Equation. 4. p_i is a packet sent by *wAP* in the time interval γ_i. $d_{mac}(p_i)$ is the overall time spent on the WiFi MAC-layer to send the packet p_i out to the client. $d^i_{mac}(k)(p_i)$ means the "wasted" part of $d_{mac}(p_i)$ because of the interference of AP_k.

$$I^k_{\gamma_i} = \frac{\sum_{p_i} d^i_{mac}(k)(p_i)}{\sum_{p_i} d_{mac}(p_i)}, p_i \in \gamma_i. \quad (4)$$

Here we use an example in Fig. 7 to explain the core idea of calculating ***IG***. In Fig. 7, *wAP* sends three packets, p_1, p_2 and p_3 in sequence in the time interval γ. q_1, q_2 and q_3 are three neighboring packets sent by AP_1, AP_2 and AP_3 respectively. The three bold lines in the timeline respectively refer to the time period from the moment that p_1, p_2, p_3 are ready to be sent out by *wAP*, to the moment that *wAP* has successfully received the acknowledgement of p_1, p_2, p_3. In the ideal case, *i.e.*, there is no interference, *wAP* will spend a fixed time successfully delivering p_i to its client, *i.e.*, p_3, which consumes $2ms$. However, for packet p_1, it encounters interfering packets q_1 and q_2. As a result, *wAP* "wastes" 1ms to send p_1 out as compared to the ideal case (p_3). Similarly, the existence of AP_3 causes an extra 1ms when *wAP* is sending p_2. The overall results of ***IG*** is shown in the table of Fig. 7(b).

It is noteworthy that I^k_γ ranges from 0 to 1. The closer to 1, the heavier the interference. If $I^k_\gamma = 0.5$, it means that because of the existence of AP_k, *wAP* spends 50% of the current overall transmitting time to wait for the AP_k sending packets. If $I^k_\gamma = 0$, it means the existence of AP_k has no impacts to *wAP* in time interval γ.



(a)

	AP_1	AP_2	AP_3
$d^i_{mac}(p_i)$	$\frac{1}{2}ms$	$\frac{1}{2}ms$	$1ms$
$d_{mac}(p_i)$	$3ms$	$3ms$	$2ms$
I^k_γ	$\frac{0.5ms}{3ms+3ms+2ms}$	$\frac{0.5ms}{8ms}$	$\frac{1ms}{8ms}$

(b)

Fig. 7: Example of transmitting procedure of *wAP*'s packets ($p_i, i = 1, 2, 3$) in time interval γ.

C. Application of *IG*

In this section, we propose an *upgraded WING*, which utilizes ***IG*** and conducts collaborative optimization with the help of centralized control server. The architecture of upgraded *WING* is shown in Fig. 6. All *wAP*'s continuously measure the ***IG*** from its own perspective, and upload the measurement results to the control server, meanwhile receive collaborative optimization command.

1) Problem Statement: For a dense deployment scenario such as office, multiple APs are commonly located nearly in the same place and carrier sense each other. These APs often have fixed locations and their owners often are not willing to move them. Heavy interference impairs their performance when they are in the same channel. The most direct and effective way to reduce the interference is channel selection. As there are only three orthogonal channels, it is impossible to assign each AP with a distinct channel as long as there are more than 3 APs. Thus, in order to minimize the interference, we need to re-configure these APs using a smart channel combination. Assuming all these APs are *wAPs* deployed with upgraded *WING* , using ***IG*** measured, we can help these APs to collaboratively select the optimal channel. More generally, we give the formal statement of this kind of problem we want to solve:

Given n wAPs which are carrier sensing with each other, how can we collaboratively assign the channel for each of them to achieve the minimum WiFi hop delay?

*2) Three Steps for Optimization using **IG** :* To address the above problem, we propose a general three-step solution in our upgraded *WING*, as follows: 1. Set all the *wAPs*, which we want to collaboratively optimize, into the same channel and measure the interference graph, *i.e.*, ***IG*** at each *wAP*. 2. Merge the ***IG*** from different *wAPs* together in a central server to generate the bidirectional complete graph. 3. Use the graph color algorithm in [28] on the bidirectional complete graph, to assign channel for each *wAP*.

Next we use a small testbed as an example to illustrate how the above three steps works.

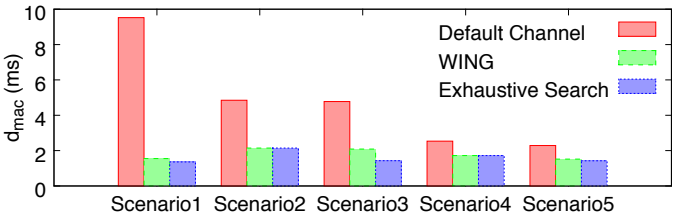


Fig. 8: The average WiFi hop delay under five random selected scenario using different channel assignment algorithms.

TABLE IV: ***IG*** measured at $AP_i, i \in \{1,2,3,4\}$ in scenario1 of Fig.8. The % value $(I_\gamma)_i^j$ in column i and row j represents the interference value of wAP_i to wAP_j.

	wAP_1	wAP_2	wAP_3	wAP_4
wAP_1		6.4	12.1	5.7
wAP_2	4.8		9.6	7.0
wAP_3	5.8	7.2		5.6
wAP_4	9.4	9.9	11.8	

3) Testbed using Collaborative Optimization: **Testbed setup:** we build a testbed with 4 *wAPs* as an example to show how upgraded *WING* uses the above methodology (§VI-C2) to collaboratively optimize the channel selection. Note that the above methodology is also applicable to more *wAPs*, and each step works the same regardless the number of *wAPs*. The traffic in each *wAP* of the testbed is derived from the real measured traffic pattern from the deployed *wAPs* in the wild. In this section, we randomly select 20 types of real traffic patterns to show the generality of our solution. For each run, we assign 4 types of traffic patterns to 4 *wAPs* respectively and we conduct 5 runs labeled as scenario 1∼5. Note that 4 *wAPs* can carrier sense each other in each scenario. The ***IG*** measured under certain scenario is shown in Table IV.

Schemes compared: the first algorithm we used to compare with our method is called **"Default Channel"** algorithm. "Default Channel" represents that there is not any optimization algorithms on 4 *wAPs*, *i.e.*, they are using the same channel by default since they are set up. For our *wAPs*, the default channel is channel 1.

The second algorithm we used to compare with our method is called **"Exhaustive Search"** algorithm. This is the optimal algorithm, where 4 *wAP* try every possible channel combinations and select the combination which has the minimum average WiFi hop delay. As there are four *wAPs* and three orthogonal channels, it needs potentially $3^4 = 81$ iterations to exhaustively search all the combinations to get the optimal result. We call each search of combination as a *measurement iteration*. In this controlled experiment, we replay the same traffic for each iteration to ensure that it finds the best channel combination.

Results: Fig. 8 shows the average WiFi hop delay under different channel combinations calculated by the three algorithms, under five random selected traffic scenarios. We can see that *WING* achieves almost the same performance as the optimal exhaustive search, but with no need of brutally search all the channel combinations. Our WING method only needs a const convergence time to find the final channel combination according to the ***IG***, which consists of one iteration of measurement and the execution time of graph coloring [28]. A typical measurement iteration period is 5 minutes [25], and the algorithm execution time in central server is much less compared with the measurement iteration period. However, for the exhaustive search algorithm, although it can find the best channel combination, it has to measure every possible combinations to get the optimal one. As there are 3 orthogonal channels, it has to iterate for 3^n for n collaborative APs. The long convergence time makes it hardly usable in practice.

VII. Related Work

Active interference measurement. [29]–[31] measure the conflict graph in an active way. In [29], [31], they use either a small probe or active method to measure the interference, which could interrupt the normal usage of residents and do not reflect the actual delay experience by the user traffic.

Passive interference measurement. [32] decode the packet header information at the user side to predict the packet delivery probability and the throughput. The conflict graph is static which cannot reflect the interference between the APs caused by the actual traffic. PIE [16] also uses the packet level information to passively get empirical conflict graph between enterprise WLAN APs. However, APs' packet timestamps have to be highly synchronized at packet level, which is hard to achieve with many autonomous APs. [33] acquires the conflict graph based on the measurement-calibrated propagation models, but not the actual traffic. [34] estimates the interference between APs and links in a live wireless network by additional deployment of multiple sniffers across the network. In contrast to these works, *WING* can be deployed in each independent AP to passively measure the interference without the help of users and the measurement is based on the actual traffic.

WiFi monitoring tool. [18]–[20], [35] provides diagnosing methods and tools in the WiFi network, which inspired the design of *WING*. These monitoring methods need extra NICs or boxes for sniffing packets in the WiFi network, while *WING* method act as a software package which can work on any OpenWrt-compatible APs and does not need any extra NICs or boxes used for sniffering.

VIII. Conclusion

Previous studies have shown that WiFi, as the dominant last hop access to the Internet, has become the weakest link in the round trip network delay. Therefore it is critical to understand and minimize the WiFi interference in order to reduce the WiFi hop delay. For the first time in the literature, this paper defines an intuitive and accurate metric to quantify the impact of interference on each actual packet on a commodity AP, and it can be measured with a small kernel modification and with little overhead. Our 29-AP two-month measurements in the wild confirms this metric's accuracy, and the individual AP optimization results based on this metric show significant performance improvement over other approaches.

We believe that this paper is an important towards the vision [21]–[25] in which hundreds of millions of APs in the world all measure their empirical and time-variant interference and coordinate with their neighbors to jointly determine their best channels, power, or even locations.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedbacks. We thank Juexing Liao for her proofreading. This work has been supported by National Natural Science Foundations of China (NSFC) under Grant 61472210 & 61472214, the Sate Key Program of National Science of China under Grant No.61233007, the Tsinghua National Laboratory for Information Science and Technology key projects, the National Key Basic Research Program of China (973 program) under Grant No.2013CB329105, the Global Talent Recruitment (Youth) Program and the Cross-disciplinary Collaborative Teams Program for Science, Technology and Innovation, of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

REFERENCES

[1] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "Wifi can be the weakest link of round trip network latency," in *INFOCOM, 2016 Proceedings IEEE*, 2016.

[2] 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.

[3] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan, "Modeling web quality-of-experience on cellular networks," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 213–224.

[4] W. Mühlbauer, S. Uhlig, A. Feldmann, O. Maennel, B. Quoitin, and B. Fu, "Impact of routing parameters on route diversity and path inflation," *Computer Networks*, vol. 54, no. 14, pp. 2506–2518, 2010.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2010.

[6] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "Detail: reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 139–150, 2012.

[7] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 523–536.

[8] G. Chen, Y. Zhao, D. Pei, and D. Li, "Rewiring 2 links is enough: Accelerating failure recovery in production data center networks," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015.

[9] G. Chen, Y. Zhao, and D. Pei, "Alleviating flow interference in data center networks through fine-grained switch queue management," *Computer Networks*, 2015.

[10] A. Singla, B. Chandrasekaran, P. Godfrey, and B. Maggs, "The internet at the speed of light," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 1.

[11] I. Cisco, "Cisco visual networking index: Forecast and methodology, 2013–2018," *CISCO White paper*, 2013.

[12] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 153–165.

[13] S. Sundaresan, N. Feamster, and R. Teixeira, "Measuring the performance of user traffic in home wireless networks," in *Passive and Active Network Measurement Conference*, 2015.

[14] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 297–308.

[15] J. Manweiler, P. Franklin, and R. R. Choudhury, "Rxip: Monitoring the health of home wireless networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 558–566.

[16] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "Pie in the sky: online passive interference estimation for enterprise wlans," in *Proc. of NSDI*, 2011.

[17] J. Geier, "Assigning 802.11b access point channels," in *WiFi Planet*, 2004.

[18] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through WiFi APs," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 339–350.

[19] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage, *Jigsaw: solving the puzzle of enterprise 802.11 analysis*. ACM, 2006, vol. 36, no. 4.

[20] 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*. ACM, 2007, vol. 37, no. 4.

[21] A. Patro, S. Govindan, and S. Banerjee, "Outsourcing home ap management to the cloud through an open api," *Traffic (in Mbps)*, vol. 2, p. 4, 2013.

[22] Raychaudhuri, "Savant – high performance dynamic spectrum access via inter network collaboration," http://www.winlab.rutgers.edu/pub/docs/focus/documents/SAVANT_focus_sheet_7.14.pdf. [Online]. Available: http://www.winlab.rutgers.edu/pub/docs/focus/documents/SAVANT_focus_sheet_7.14.pdf

[23] N. Feamster, "Managing the home network," http://www.gcatt.gatech.edu/news/Feamster_broadband2020.pdf. [Online]. Available: http://www.gcatt.gatech.edu/news/Feamster_broadband2020.pdf

[24] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, "Flowqos: Qos for the rest of us," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 207–208.

[25] A. Patro and S. Banerjee, "Coap: A software-defined approach for managing residential wireless gateways."

[26] I. C. S. L. M. S. Committee *et al.*, "Wireless lan medium access control (mac) and physical layer (phy) specifications," 1997.

[27] "MAC80211," http://linuxwireless.org/en/developers/Documentation/mac80211/, accessed: 2015-12-01.

[28] E. Rozner, Y. Mehta, A. Akella, and L. Qiu, "Traffic-aware channel assignment in enterprise wireless lans," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*. IEEE, 2007, pp. 133–143.

[29] N. Ahmed, U. Ismail, S. Keshav, and K. Papagiannaki, "Online estimation of rf interference," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 4.

[30] N. Ahmed and S. Keshav, "Smarta: a self-managing architecture for thin access points," in *Proceedings of the 2006 ACM CoNEXT conference*. ACM, 2006, p. 9.

[31] P. Kanuparthy, C. Dovrolis, K. Papagiannaki, S. Seshan, and P. Steenkiste, "Can user-level probing detect and diagnose common home-wlan pathologies," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 1, pp. 7–15, 2012.

[32] S. Liu, G. Xing, H. Zhang, J. Wang, J. Huang, M. Sha, and L. Huang, "Passive interference measurement in wireless sensor networks," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. IEEE, 2010, pp. 52–61.

[33] X. Zhou, Z. Zhang, G. Wang, X. Yu, B. Y. Zhao, and H. Zheng, "Practical conflict graphs for dynamic spectrum distribution," in *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. ACM, 2013, pp. 5–16.

[34] U. Paul, A. Kashyap, R. Maheshwari, and S. R. Das, "Passive measurement of interference in wifi networks with application in misbehavior detection," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 3, pp. 434–446, 2013.

[35] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, and N. Feamster, "Peeking behind the nat: an empirical study of home networks," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 377–390.