Measurement-based, Practical Techniques to Improve 802.11ac Performance

Apurv Bhartia[†], Bo Chen[†], Feng Wang, Derrick Pallas, Raluca Musaloiu-E, Ted Tsung-Te Lai, Hao Ma
{apurv,bochen,wangf,pallas,ralucam,tedlai,hao}@meraki.com
Cisco Meraki

ABSTRACT

Devices implementing newer wireless standards continue to displace older wireless technology. As 802.11ac access points (APs) are rapidly adopted in enterprise environments, new challenges arise. This paper first presents an overview of trends in enterprise wireless networks based on a large-scale measurement study, in which we collect data from an anonymous subset of millions of radio access points in hundreds of thousands of real-world deployments. Based on the observed data and our experience deploying wireless networks at scale, we then propose two techniques that we have implemented in Meraki APs to improve both overall network capacity and performance perceived by end users: (i) a dynamic channel assignment algorithm, TurboCA, that adjusts to frequent RF condition changes, and (ii) a novel approach, FastACK, that improves the end-to-end performance of TCP traversing high-throughput wireless links. Finally, we evaluate TurboCA with metrics taken from a variety of real-world networks and evaluate TCP performance of FastACK with extensive testbed experiments.

CCS CONCEPTS

• **Networks** → **Wireless access points, base stations and infrastructure**; **Network measurement**; **Network protocol design**;

KEYWORDS

Network measurement, 802.11ac, Channel assignment, TCP

ACM Reference format:
Apurv Bhartia, Bo Chen, Feng Wang, Derrick Pallas, Raluca Musaloiu-E, Ted Tsung-Te Lai, Hao Ma. 2017. Measurement-based, Practical Techniques to Improve 802.11ac Performance. In *Proceedings of IMC '17, London, United Kingdom, November 1–3, 2017,* 15 pages.
https://doi.org/10.1145/3131365.3131398

1 INTRODUCTION

Wireless networks continue to grow at a prolific rate and have become the communication medium of choice in enterprise environments, including offices, campus, hospitality, and retail. A recent report [3] predicted that by 2020, 20 billion devices will be online and extensively using the wireless medium for data transfer. Further, [6] estimates that 1.53 ZB of data will be transferred over-the-air as a result. However, allocation of unlicensed spectrum has not increased as quickly as demand, making it a precious commodity.

Several researchers [20, 30, 36, 44] in the past have studied isolated wireless networks and their behavior, proposing protocols and methodologies intended to improve the performance of wireless networks under various settings. While their work has improved our understanding of wireless network behavior, there are very few studies across numerous varying environments in real-world deployments.

Further, the introduction of 802.11ac has led to rapid adoption of devices capable of achieving gigabit data rates over the air. This not only changes, but also defies some of our assumptions regarding performance of enterprise wireless networks in the wild. A previous study [18] presented measurement aspects with a holistic view of networks. In this paper, we look at wireless networks from a performance perspective. We argue that it is necessary to redesign a few fundamental knobs in order to ensure high performance.

The Meraki system provides a unique vantage point over a wide range of network deployments owing to its cloud-based management architecture. The Meraki backend system polls each Meraki AP periodically to extract network information such as traffic usage, channel utilization, client density, etc., which provides crucial insight into the deployed network. This information is then written into a database and can be accessed by network administrators through the Meraki dashboard. This database has grown to contain information drawn from millions of network devices, including APs, switches, and gateway routers, and records billions of client devices. In this paper, we focus on data from APs that are distributed geographically over the mainland United States.

We collect data from tens of thousands of real-world operational wireless networks and make the following observations:

- From a dataset consisting of 50 billion packets across a hundred thousand APs, we look at the device, channel, density and traffic trends in enterprise wireless networks at scale.
- Given the current spectrum trends, we identify the need for changes in channel planning mechanisms, and make the case for a new auto channel planning algorithm. We then study the benefits observed after deploying the new solution in a few real-world test networks.
- From the traffic trends, we also argue that TCP, as it is implemented today, negatively interacts with performance features in high-throughput wireless networks. By analyzing

[†]Co-primary authors

IMC '17, November 1–3, 2017, London, United Kingdom

ACM ISBN 978-1-4503-5118-8/17/11. . . $15.00
https://doi.org/10.1145/3131365.3131398

several key TCP parameters, we show that the end-to-end performance of TCP is underwhelming when compared to the high over-the-air performance that is achievable. We then discuss a potential solution to close this gap, and compare it against the baseline TCP in our performance lab setup.

The remainder of this paper is organized as follows: In section 2 we give an overview of Cisco Meraki architecture and our data collection methodology. We then study wireless network behavior by focusing on a few fundamental metrics in section 3. We focus on the channel utilization and behavior seen in various deployments in section 4, and then introduce a simple but effective automatic channel planning mechanism. We study traffic behavior, in particular the performance of TCP streams in the presence of 802.11 wireless links in section 5, and introduce TCP FastACK, which can improve the performance of TCP over 802.11 links. Finally, we conclude in section 6.

2 SYSTEM ARCHITECTURE

2.1 Overview

The Cisco Meraki system consists of wireless APs deployed at customer sites, a back-end subsystem hosted in data centers deployed at various geographic locations, and a front-end web user interface for network management. Individual APs are grouped into deployments called networks and multiple networks can be managed by a single organization. Each wireless AP either utilizes an Ethernet uplink (gateway mode) or automatically meshes with nearby APs (repeater mode). This paper focuses on gateway APs.

APs use an encrypted tunnel to communicate with the Meraki back-end, from which they download configuration parameters. These parameters are generated based both on user input in the front-end UI as well as centralized algorithms in the back-end, e.g. in the case of automatic channel assignment.

The back-end subsystem also collects real-time and periodic statistics from APs, stores the statistics in a time-series database, post-processes this data, and generates automated reports. The front-end UI displays collected statistics for clients, APs, and networks based on user requests.

Wireless client devices (laptops, mobile phones, &c.) discover and associate with APs, which relay wireless traffic between the client and the local wired network. Client traffic does not go through the back-end subsystem.

All Meraki hardware platforms incorporate at least one 802.11 radio. Most models contain additional radios for multi-band operation, scanning, or alternative protocols. Most models (including all 802.11ac APs) are equipped with a single-antenna scanning radio, which scans all available channels over 150 ms intervals, gathering neighbor and channel information. All APs run the Linux kernel, with proprietary vendor radio drivers and the Click Modular Router [29] for packet processing. Various aspects of 802.11 are implemented in the wireless driver, Click, and third-party software, e.g. hostapd for configuration, advertisement, and authorization.

Because 802.11ac radios are able to process packets at Very High Throughput (VHT), the radio itself is often programmable and timing-sensitive features are implemented in microcode, which is a black box from Meraki's perspective. Some platforms also support offload features that allow packets to bypass the host CPU processing altogether, increasing performance at the expense of flexibility. This means that features critical for high performance (frame aggregation, power save, bit rate selection, &.) are inaccessible without significant vendor interaction.

2.2 Data Collection Methodology

Each Meraki AP keeps track of various statistics about itself, client devices associated to it, and the RF environment. Some statistics are only stored in memory. Others are sent to backend, which aggregates raw data and stores the results in a time-series database called LittleTable [42]. For the network behavior study, we leverage the ability of Meraki's cloud to collect live statistics from APs in addition to querying historical data.

3 WIRELESS BEHAVIOR

3.1 Related Work

Past wireless measurement work is significant. Customized firmware and hardware [36, 44] has been developed to accurately monitor 802.11 characteristics. Passive monitoring [20, 30] uses additional hardware sniffers to understand the wireless environment and does not require modifications to existing network infrastructure. However, these custom deployments can only be evaluated at small-scale without the help from industry. WiFiSeer [45] presents a campus-level measurement study over 47,000 unique clients by using a practical software approach. They conclude that using RSSI to select AP is inadequate and develop ML algorithms based on radio factors (e.g. channel utilization) to choose a low-latency AP. It requires the client device to pre-install their software to obtain the AP recommendation. Our previous work [18] studied enterprise wireless networks at a large scale for the first time. This paper focuses more on metrics related to performance and provides insights on how to improve performance. BeHop [51] is presented as a testbed for characterizing dense networks and relies on TCP latency as a key metric, showing the effect of band-steering on this measurement.

3.2 Network Trends

In our measurement-based study, we look at trends in several key categories, including capabilities of both access points and client devices, channel utilization on both 2.4 GHz and 5 GHz bands, network density with respect to access points and client devices, as well as traffic characteristics.

3.2.1 Device Trends. For better understanding, we classify the device trends by looking at both access points and clients separately.

Access Point Perspective: Meraki manages several million APs in a variety of enterprise deployments. Because these devices regularly check in to the back-end, we are able to tell how many APs are active at any given time. World-wide, roughly 52% of active Meraki APs are 802.11ac, 47% are 802.11n, and 1% are 802.11g. We expect the percentage of 802.11ac APs to continue increasing rapidly before newer 802.11 standards will be adopted by the enterprise wireless market. Of all the APs, fewer than 1% have a single antenna chain, 73% have two, 24% have three, and 2% have four. Meraki APs are deployed indoors 93% of the time vs. 7% outdoors.

Client Device Perspective: Figure 1 shows the trend in capabilities advertised to the APs by 1.7 million client devices. Since 2015 [18],

802.11ac-capable clients have grown significantly, from 18% to 46% of devices, corresponding with an increase in the number of devices that can utilize 40 MHz and 80 MHz channel widths. Surprisingly, the relative number of devices supporting 2.4 GHz but not 5 GHz has remained steady at around 40%. Number of devices supporting 2-stream MIMO has increased from 19% to 37%.

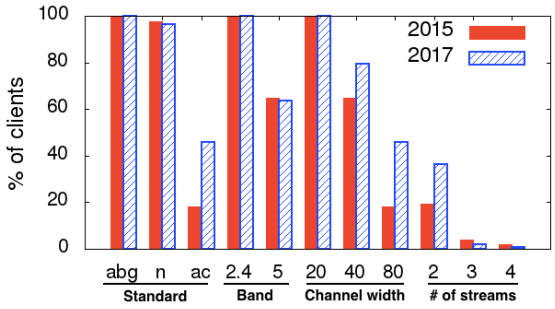

Figure 1: Advertised client capabilities

3.2.2 Channel Trends. Figure 2 tracks channel utilization trends for enterprise networks in both 2.4 GHz and 5 GHz. We first consider all APs in networks with at least 10 APs, which shows a median utilization of 20% for 2.4 GHz and 3% for 5 GHz; this is similar to the trend in [18], where we observed median utilization of 22% on 2.4 GHz band and 2% on 5 GHz band. It is important to note that utilization depends on actual deployment scenarios. As a comparison we collect the information from Meraki HQ office network as a single office network where we have around 31 – 35 APs and 300 – 400 clients in a floor during regular office hours. We see dramatically higher numbers with median utilization at 82% for 2.4 GHz and 23% for 5 GHz. This suggests an effective channel assignment algorithm is critical for network capacity in high density deployments.

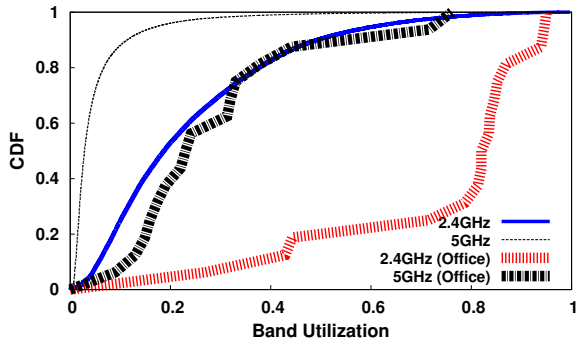

Figure 2: CDF of utilization seen by APs in networks with 10 or more APs vs. Meraki HQ office

3.2.3 Network Density Trends. Network density is a good way to gauge medium interference, and we look at the density numbers for both the APs and client devices.

Access Point Density: For any AP, there may be other APs within transmission range on the same channel, which we label interferers and show in Figure 3. On 2.4 GHz, APs see a median of 7 interferers and 90% see fewer than 29. On 5 GHz, it is less crowded, with a median of 5 and 90% seeing fewer than 14.

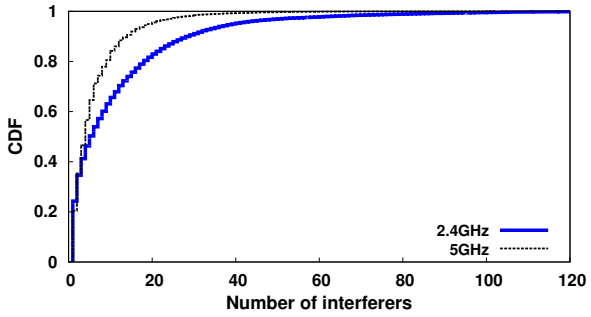

Figure 3: CDF of interfering APs

Client Density: To measure client density we selected a 1000 networks, each with more than 10 active 802.11ac APs, comprising 41,000 APs in total. We then record the maximum number of associated clients for each AP in March 2017.

Of these APs, 33% had a client density of 5 or fewer, 22% had a density of 6–10, 20% had a density of 11–20, and 25% had a density of 21 or greater. The most heavily loaded AP in the data set had 338 unique, associated clients.

3.2.4 Traffic Trends. To understand traffic trends, we inspect packet-level details which provide insight into access categories and bit rate usage seen in real-world deployments.

Quality of Service: The 802.11e standard defines four separate Access Categories (ACs), which affect how aggressively an AP should attempt to transmit a frame. From least to most aggressive, the ACs are Background (BK), Best Effort (BE), Video (VI), and Voice (VO). Frames in a more aggressive AC have faster, longer access to the medium but also exhaust retry attempts more quickly. The AC is often mapped from other QoS markings, typically Differentiated Services Code Point (DSCP) bits in an IP header.

Figure 4 describes the latency experienced by packets in different ACs. Here, the latency is the time interval between transmission of a frame and reception of its link layer acknowledgment. Overall, we see 14% BK traffic and 86% BE, with little use of VI and VO. This varies greatly among deployments and depends on gateway routers to set or preserve DSCP marks.[1] In a specific but typical enterprise office environment, we found 10% VO and 90% BE with little BK or VI in the middle of a work day.We also note the lack of significant VI or VO traffic in the field, and our intuition is that incorrect DSCP markings by the upstream devices are the root cause of this.

Each AC experiences loss differently, where loss means failure after exhausting retransmission attempts. In this data set, 5.0% of BK packets were lost, 2.7% of BE, 0.2% of VI, & 0.9% of VO, with overall loss of 3.0%.

Bit Rate Usage: Wireless standards specify sets of rates at which data may be transmitted over-the-air, dependent upon hardware

[1] If clients set DSCP marks on uplink traffic, Meraki APs can be configured to mirror that same mark on the downlink.

capabilities and signal quality. The rate of a transmission is important because it inversely affects the total air time of a transmission. Bit rate can be increased in various ways, often with trade-offs. Utilizing a wider channel comes at the expense of spectrum contention and spreads power across the channel. Denser modulations both require better linearity in the power amplifier, resulting in lower gain and therefore shorter range, and are subject to higher bit-error rates. Multiple streams require a better signal-to-noise ratio and good spatial diversity. As the bit rate increases, the chance of additional latency due to retransmission or loss increases. Bit rate selection algorithms are expected to adapt to changing conditions and have been extensively studied [47, 49]. With 80% of clients supporting

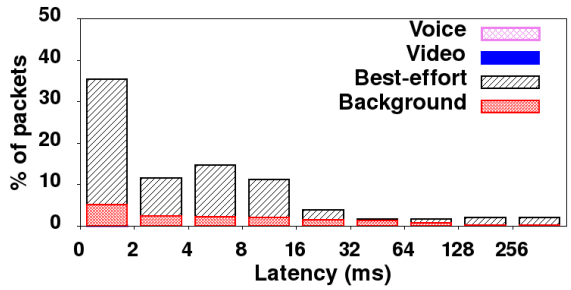

Figure 4: Latency experienced by Access Category

40MHz channels and 36% supporting 2 streams, typical 802.11n/ac clients will have maximum bit rates of 300 Mbps and 867 Mbps respectively.[2] Figure 5 shows bit rate usage for all the clients in the field over the course of one day over the 5 GHz band, with most rates between 256–512 Mbps.

In Section 4.6.2, we argue for bit rate usage as a metric for network performance.

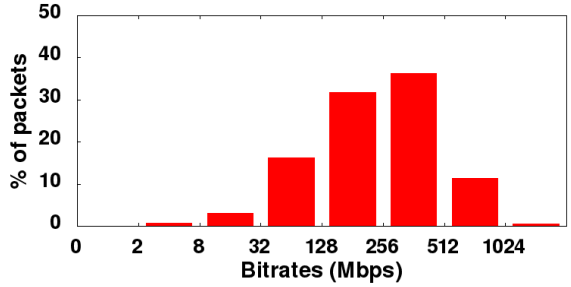

Figure 5: Bit rate distribution for all clients in the 5 GHz band

4 AUTO-CHANNEL ASSIGNMENT

As 802.11ac devices become more widely deployed, channel assignment becomes more complicated. In this section we examine the challenges of developing an efficient and practical automatic channel assignment solution. We then present our solution, TurboCA, deployed in late 2016, and evaluate its performance in large networks (university & museum).

[2] Assuming a short guard interval (SGI) of 400ns.

4.1 What's new in 802.11ac

4.1.1 Extended Channel Width. One major improvement in recent 802.11 standards is channel bonding. With no spectrum contention, wider channels generally imply higher potential throughput due to increased capacity. The 802.11n standard allows two adjacent 20MHz channels to be bonded into a single 40MHz channel. Continuing this trend, 802.11ac allows for 80MHz, 160MHz, and 80+80MHz (non-adjacent) configurations.

In the US, the Federal Communications Commission [2] currently allows unlicensed use of twenty-five 20MHz, twelve 40MHz, six 80MHz, and two 160MHz channels in 5 GHz bands. In contrast, only 3 non-overlapping are available in the 2.4 GHz. The 8× difference in available spectrum and the flexibility in choosing channel width greatly increases the complexity of channel assignment.

Higher potential throughput comes at the expense of contention. For an 80MHz transmission, interference on any of the four 20MHz sub-channels can causes contention or corruption. Examining 80MHz-capable APs in our deployments, Table 1 shows that 34% are manually configured to decrease the channel width. For networks with more than 10 APs, 37% have had administrators decrease channel width for the entire network.

Channel Width	All APs	Large Networks(> 10 APs)
20MHz	14.9%	17.3%
40MHz	19.1%	19.4%
80MHz	66.0%	63.3%

Table 1: Channel width for individual 802.11ac APs and for large networks, prior to TurboCA auto-channel assignment

4.1.2 Virtual Carrier Sense. It is well known that neighboring APs on overlapping channels not only increase medium access contention but can also introduce the hidden node problem [19]. To mitigate this, 802.11 provides request to send, clear to send (RTS/CTS) as a virtual carrier sense mechanism.

With RTS/CTS, nearby APs operating on overlapping channels would ideally share the wireless medium, with each consuming a fair share of the airtime. This behavior, verified in Section 5.6.3, motivates us to model the performance of such APs in Section 4.4.

4.2 Related Work

Channel assignment has been extensively studied in the literature. [21] provides an survey on some of proposed approaches. They can be broadly divided into following categories: (i) Centralized Approaches [8, 17, 24, 37, 39, 44, 48]. They rely on a central WLAN controller to perform channel assignment. For example, PIE [44] presents online interference estimation by collecting information from the various APs to a WLAN controller and infer the interfering patterns. (ii) Decentralized Approaches [7, 23, 28, 31, 38]. In these approaches, an AP selects the channel itself by using information from its neighbors. A key limitation of these approaches is that they do not allow for lightly loaded APs to give up good-quality channels for overall network optimization. (iii) Channel Hopping Approaches [10, 16, 22, 32, 46, 50]. In this scheme, APs hop between different channels based on a hopping sequence. The benefit is that it could utilize channel diversity and prevent an AP from getting stuck in a low throughput channel. However, the limitation is that

it needs accurate knowledge of interfering APs and their traffic dynamic to be effective. Deciding on a hopping sequence in advance without this information is undesirable. While IQ-Hopping [16] takes into account these considerations, it does not take into account the side effects associated with a channel switch. (iv) Flexible Channel Widths Approaches [11, 34, 41]. Flexible channelization gives each transmission the flexibility of choosing center frequency and channel width. If not designed carefully, it could incur significant overhead.

4.3 Motivation for TurboCA

Several channel assignment solutions have been proposed over the past decade. In realistic deployments, we found that they fell short of Meraki's need for an efficient, high-speed, channel-bonding aware automatic assignment algorithm.

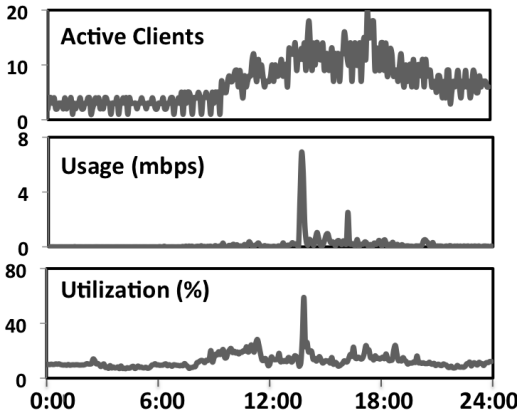

Figure 6: 802.11ac 3x3 AP snapshot in an office environment

4.3.1 Performance vs. Stability. Figure. 6 shows a snapshot of an 802.11ac AP with three antennas, placed at Meraki HQ, operating on channel 36 with 20MHz width, taken on a weekday. While the number of associated clients passing traffic change gradually during the day, both data-usage and channel utilization change more rapidly. For example, around 2:00pm, there is a sudden burst of traffic for 30 minutes. This coincides with a spike in channel utilization. A good channel assignment scheme should be able to react to such events.

Such trends are typical and not anomalies in today's networks. Network usage depends on user demands, which is hard to predict. Events over the course of the day can affect the location of devices, and therefore wireless conditions. During lunch, a cafeteria is likely to experience higher load. Network usage is likely higher in a conference room when a meeting is in progress. In a school, the network trends are likely to correlate with class schedules and enrollment.

Because conditions that affect the performance of a network can change rapidly, we believe that a channel assignment algorithm that adapts to RF changes frequently is necessary to ensure good real-world performance.

On the other hand, it is unrealistic to react to channel conditions instantly, since channel changes can disrupt client traffic. Without a notification that an AP is changing channels, clients have to first detect that the AP is no longer responding, then go through a scanning process to find an appropriate AP and re-associate[3]. Our tests show that this process usually takes around 5 seconds for laptops, and around 8 seconds for mobile devices. This can be especially disruptive for latency-sensitive applications, like Voice-over-IP.

To reduce the overhead of switching channels, 802.11h standardizes Channel Switch Announcements (CSAs). Prior to a channel switch, the AP broadcasts several beacons announcing it's intention to move to a target channel. Clients can then follow the AP to the target channel without scanning. Unfortunately, not all clients respond to CSAs and the beacons might be missed even by clients that do support CSAs[4]. Thus, a goal of TurboCA is to avoid too many channel switches.

4.3.2 Optimality vs. Complexity. Suppose there are k available channels and n APs. The number of all possible channel assignments is k^n. It is well known that channel assignment can be modeled as a graph coloring problem, which is NP-complete, so any practical solution has to be a heuristic solution. A greedy approach is straightforward. All APs choose a best channel sequentially, which can result in a locally optimal solution for any single AP. However, there may be a better network-wide (globally optimal) solution.

For example, assume two APs, A and B, are near each other and only channels 36 and 149 are available. Initially both channels are clean, i.e. low interference, A is on channel 36, and B is on channel 149. Then an interferer close to B (but not A) starts operating on channel 149. With sequential assignment, A may still choose channel 36, to avoid channel switch. To avoid interfering with A, B continues choosing channel 149. Each AP sees this assignment as locally optimal. However, considering the whole network, a globally optimal assignment is A on channel 149 and B on channel 36.

This limitation exists for all distributed channel assignment approaches. Even with a centralized approach, it is still impractical to search the entire solution space, so we must balance the optimality of channel plan and the computational overhead of the algorithm.

4.4 System Design

In this section we present the design and architecture of TurboCA. The Meraki back-end system collects the required data from all the APs periodically, including neighbor reports, channel utilization, traffic load, clients capability, etc. The TurboCA service then generates a new channel plan, both periodically and event-based, and delivers the updated configuration to the participating APs. Fundamentally, TurboCA treats each network as a unit.

4.4.1 Performance Metrics. First, we present two metrics used byTurboCA : (i) $NodeP$ and (ii) $NetP$. (i) represents the effectiveness of a single AP with a potential channel assignment. A higher value of $NodeP$ implies that the clients associating to this AP will have a better wireless experience. (ii) is the target function of the entire TurboCA algorithm over the entire network. Higher values imply that the network has a better channel assignment. We now explain these metrics in more detail.

[3] 802.11k Neighbor Reports may reduce the need to scan by providing a good heuristic for nearby APs in the same network.

[4] Other steering mechanisms exist but also require client support.

NodeP(c, cw): estimates the performance on channel c with channel width cw as:

$$NodeP(c, cw) = \prod_{b=20MHz}^{cw} channel\_metric(c, b)^{load(b)}$$

where

$$channel\_metric(c, b) = [airtime(c, b) \times capacity(c, b)] - penalty_c$$

$airtime(c, b)$ represents the estimated proportion of airtime an AP can expect on channel c with channel width b and is calculated based on the channel utilization and the neighboring APs' reports. $capacity(c, b)$ of channel c is estimated using the channel quality, non-wifi interference, and channel width b. $load(b)$ represents the weight of the channel_metric. It is proportional to the number of associated clients with maximum channel width b and their corresponding usage. $penalty_c$ is used to characterize the negative effect associated clients experience on switching to channel c. It is channel and hardware related. For example, for 2.4 GHz radios, since many client devices do not support CSA, $penalty$ is set to a very high value to avoid disassociation for connected clients.

$NodeP$ has two important properties; (i) If channel c is heavily utilized or there are many neighboring APs on the same channel, $NodeP$ will quickly approach 0. (ii) If associated clients do not support wider channel widths, $NodeP$ will not increase for wider channels. In this case, an AP can avoid adjusting its channel width according to the clients' capabilities.

NetP: estimates overall network performance and is the product of $NodeP$ over the entire set V of APs in the network, given a proposed channel plan.

$$NetP = \prod_{v \in V} NodeP$$

These performance functions provide several benefits. First, the metric prefers to assign wider channels to APs with higher client density and usage. This encourages enhanced wireless experience for each connected client. Further, by incorporating $penalty$, APs with few/zero associated clients are more likely to change channel to achieve neighbor channel isolation. Second, single node failure is avoided. If spectrum coverage or the total network throughput is the performance function, it is easy to have a high metric despite assigning poor channels to several APs. In contrast, $NetP$ will approach 0 as a single $NodeP$ approaches 0.

4.4.2 AP Channel Calculation (ACC). The AP Channel Calculation ($ACC(v, \psi)$) is the basic channel computation in TurboCA. It produces a channel assignment for a target AP v that maximizes $NetP$. Since the only potential channel switch is for v, $NodeP$ is only affected for n and it's neighbors.

The parameter ψ denotes a set of APs that *ACC* should *not* consider in the current calculation. By ignoring the current channel of these APs, in essence presuming a channel change, TurboCA avoids locally optimal solutions as mentioned in 4.3.2.

4.4.3 Network Basic Operation (NBO). Network Basic Operation (NBO) is an operation (Algorithm 1) in TurboCA that iterates through the entire AP set V, once to perform the channel assignment. It takes the scan results, current channel assignment and load information as the input and uses parameter i to determine how much the new assignment is affected by the current channel assignment. Initially, NBO starts with no assignments in the proposed channel plan (PCP). At each step, NBO picks a random AP without a assignment in the PCP (line 4) and forms a candidate set of nodes (CSN) up to i hops away, also without PCP assignment. The algorithm then randomly removes a node n from the CSN and adds $ACC(n, CSN)$ to the PCP (line 10) until the CSN is empty.

Algorithm 1: Network Basic Operation

```
1  Input: Scanning results, current load for all APs in the network, current
       channel plan, hop limit i.
2  S ← V, PCP ← φ // whole network AP set
3  while S ≠ φ do
4      Randomly pick AP n ∈ S
5      S_group ← n and APs within i hops of n
6      S ← S − S_group
7      while S_group ≠ φ do
8          Randomly pick AP m ∈ S_group
9          S_group ← S_group − {m}
10         PCP ← PCP ∪ {(m, ACC(m, S_group))}
11     endwhile
12 endwhile
13 Output: PCP: A proposed channel plan for all APs in the network
```

The hop limit i controls how much the new assignment is affected by the initial assignment. If $i = 0$, NBO iterates over all APs, and assigns a channel for each AP considering only it's interfering neighbors, in effect considering all initial neighboring channel assignments. However, if i is sufficiently large, the first S_{group} includes all the APs, and their initial channel assignment is completely ignored.

When $i > 0$, the first AP evaluated by ACC has the chance to pick a relatively clean channel. More generally, it allows NBO to assign better channels to APs evaluated earlier in the process. Therefore, the probability of picking any AP (line 8) is weighted proportionally to the $load$ on that AP to encourage better assignment to more heavily loaded APs.

4.4.4 Run-Time Schedule. As mentioned in Section 4.3.1, one of the biggest challenges of TurboCA is maintaining a mostly stable channel assignment while increasing the frequency of channel assignment computation. To achieve this, TurboCA runs multiple rounds of NBO and picks the best proposed channel plan. The actual number of runs is proportional to the network size. Whenever a single run of NBO increases $NetP$, the new proposed channel plan replaces the assigned channel plan for the following rounds.

In practice, we run NBO with different i values on different schedules. By default we run NBO with $i = 0$ every 15 minutes. Every 3 hours, we run NBO with $i = 1$ followed by $i = 0$. Once a day, we run $i = 2$, then $i = 1$, then $i = 0$. This allows TurboCA to adapt to volatile channel conditions within 15 minutes. The longer schedules avoid getting stuck in local optima.

All schedules end with $i = 0$, since that guarantees that $NetP$ will increase unless a local optimum is found in previous rounds. When a local optimum is found in $i > 0$, we are able to avoid unnecessary channel switches in the final round.

4.5 Practical Deployment Issues

4.5.1 5 GHz band vs. 2.4 GHz band. TurboCA manages both 5 GHz and 2.4 GHz channel planning. Though there are many more available channels for 5 GHz band, higher utilization in 2.4 GHz results in more channel switches on that band. When utilization is above 90%, even small variations can reduce $NetP$ by half. TurboCA adds a relatively large channel switch penalty to the $NodeP$ expression in such cases.

4.5.2 DFS Channels. Many of the available 5 GHz channels are subjected to Dynamic Frequency Selection (DFS) and users must vacate the channel if radar signals are detected during operation. Furthermore, access points on DFS channels must perform a 1-minute Channel Availability Check (CAC) before transmitting. In the US, devices not certified for DFS have only nine 20MHz channels, four 40MHz, two 80MHz, and zero 160MHz channels to choose from, increasing the difficulty of selecting a good channel plan.

We have added several features regarding the DFS channel assignment. First, no AP can switch to a DFS channel if there are connected clients. This prevents clients from having to wait for the AP to complete CAC. Second, since radar events on DFS channels mandate a channel switch, TurboCA maintains an appropriate fallback channel setting whenever an AP is using a DFS channel.

4.6 Evaluation

In this section, we evaluate the performance of TurboCA as observed from our experience in large real-word deployments.

4.6.1 Data Set. Prior to TurboCA deployment, Meraki used another dynamic channel assignment service referred to as ReservedCA here. The data collection process is similar to TurboCA, capturing both the neighbor information and channel utilization data. The key difference lies in the operation of the main algorithm. ReservedCA iterates through all the APs in sequence. Based on channel quality, for each AP ReservedCA then computes a channel assignment maximizing *that* AP's isolated performance. Further, ReservedCA only uses fixed channel widths, and re-evaluates the network every 5 hours.

We collect four different types of metrics: signal strength, usage, TCP latency, and bit rate efficiency. We test both channel assignment algorithms in two large deployments: UNet, a university campus with ≈600 APs and 40,000 daily active users; and MNet, a national museum with ≈300 APs and 10,000 daily active users. ReservedCA is enabled on both networks on 03/25/17 and data is collected from 04/01/17 to 04/15/17. TurboCA is then enabled on 04/16/17, and data is collected from 04/23/17 to 05/07/17. All the weekends data are filtered out and the daily network wireless usage is relatively stable within each network and period (refer to usage result in Section 4.6.2). Although both algorithms can stabilize themselves within 1 day, we still skip the first week of each algorithm for reliable comparison.

4.6.2 Results. To study the efficiency of TurboCA we look at several key parameters like signal strength, data-usage, TCP latency and bit rate selection.

Signal Strength: Received Signal Strength Indicator (RSSI) is a commonly used measure of signal strength based on the power of a signal as seen by the receiver[5]. However, a client might lower transmission power intentionally, e.g. to improve linearity in the amplifier or to increase battery life. This makes it hard to use RSSI as an indicator for network health. Figure 7 shows separate hour-long RSSI measurements from MNet on 5 GHz, during non-peak and peak hours. We observe that the RSSI distribution is similar for both peak and non-peak hours for different levels of RSSI. However, the usage (not shown in the figure) doubled from 12GB during non-peak hours to more than 25GB during peak-hours.This observation is similar for both TurboCA and ReservedCA.

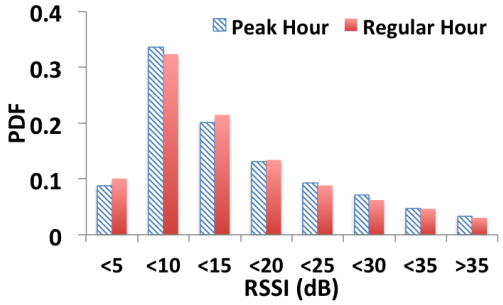

Figure 7: PDF of RSSI distribution at peak & non-peak hours

Usage: Aggregate throughput achieved under maximum load is routinely used to measure network performance. However, in real-world deployments, we cannot saturate the medium to measure throughput. Instead, we rely on the observed data-usage during a given time period. However, using usage as a metric towards evaluating the efficacy of different network settings has two key practical limitations: First, network usage is traffic-oriented and therefore depends on client behavior. Second, the network uplink often has significantly less capacity than the local network and ends up being the limiting factor of achievable usage.

Table 2 records usage from UNet and MNet under both solutions: ReservedCA and TurboCA. We calculate the daily usage variance σ_{daily}, and observe that it is relatively small in all the scenarios and both networks show similar daily network usage with different algorithms. Peak hour usage at UNet is also similar since its usage is limited by the network uplink setting most of the time. However, at MNet we observe that the usage is not limited by the uplink capacity, and consequently notice that TurboCA improves the usage by 27% over ReservedCA.

Network	ReservedCA			TurboCA		
	Daily	σ_{daily}	Peak	Daily	σ_{daily}	Peak
UNet	11.3	0.830	0.584	10.7	0.396	0.542
MNet	0.562	0.171	0.0588	0.564	0.142	0.0748

Table 2: Daily and peak hour average usage (TB)

TCP Latency: TCP latency has been evaluated as a metric for network performance [51]. It is measured at the AP as the interval between processing a TCP data packet and processing the corresponding TCP ACK. Figure 8 shows the change in latency distribution at MNet. With TurboCA, the median latency drops by 40% when

[5]Received Channel Power Indicator (RCPI) [1] is an attempt to standardize power-based signal strength

compared to ReservedCA. Similar results are seen at UNet. Lower TCP latency implies that APs and clients experience less contention for wireless medium, suggesting TurboCA improves efficiency of medium usage and therefore overall network capacity. Another observation from the result is that the distribution of latency over 400ms is similar for both TurboCA and ReservedCA. This occurs as certain client devices have arbitrarily become slow/non-responsive to the AP while being connected. We believe this is an orthogonal problem with no relation to medium availability and are investigating this as part of another ongoing issue.

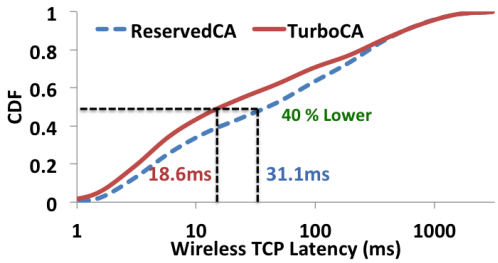

Figure 8: CDF of TCP latency distribution

Bit Rate Efficiency: Unlike RSSI or RCPI, bit rate usage can be measured at the transmitter. Successful transmissions at high bit rates are desirable to improve the medium efficiency. Enterprise networks often experience downlink-dominated traffic, so the bit rates selected by an AP directly affects network performance as perceived by the client in the current environment.

Since bit rate selection depends on client and AP capabilities amongst other factors like channel quality, distance, etc., we normalize recorded rates by the maximum possible rate supported by both for a particular association and call this metric: bit rate efficiency. Figure 9 plots this metric as a CDF and it shows that TurboCA achieves a 15% gain in bit rate efficiency at MNet. We observe a similar trend at UNet and omit the graph for brevity. Based on this evidence, we again infer that TurboCA is able to reduce medium contention, thus enabling both the APs and clients to use higher bit rates.

Besides demonstrating the improvement of TurboCA over ReservedCA, our results suggest that TCP latency and bit rate efficiency offer key insight into network performance, as compared to more commonly used indicators of usage and signal strength, especially in real-world deployments.

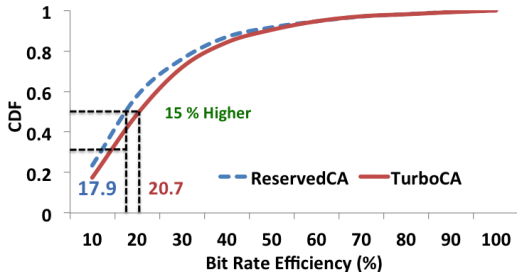

Figure 9: CDF of bit rate efficiency

4.7 Discussion

While there has been significant work in the area of channel assignment, evaluating those works against TurboCA in production networks is not practical. The key reason is that most existing work focuses directly on the optimality of the channel assignment itself, rather than overall stability [21]. Also, as discussed in section 4.5, practical considerations like frequent channel switches and regulatory challenges make it hard to deploy existing solutions in real-word networks. Further, Meraki being an enterprise networking company, we direct our focus primarily to large-scale networks with potentially hundreds of APs. Replicating such a setup in a lab environment is not a viable solution either. As a result, we do not have a quantifiable comparison of TurboCA against some of the works in literature. Having said that, we have detailed some of these works in Section 4.2 and our reasons for a new approach. Further, it will be an exaggeration to suggest that TurboCA is the most optimal channel assignment solution for all scenarios. It is important to consider that the nature of our deployments coupled with our system architecture played a critical role in the design of TurboCA. While we understand that throughput optimality is the objective of most proposed approaches, with the inherent dynamic nature of network demand and wireless environment such optimality is transient and will soon disappear. Continued iterations to follow the optimal assignment at any moment is likely to sacrifice the stability of any ongoing transmissions. We instead believe TurboCA should focus on overall client experience and the balance between the network performance and stability.

5 TCP FASTACK

With the continuous adoption of high-speed wireless standards like 802.11ac, the interaction between wireless links and TCP needs to be reconsidered. In this section, we explain the need for this and propose a mechanism towards improving the performance of TCP atop 802.11ac wireless networks.

5.1 The Problem of TCP over 802.11ac

TCP is a self-clocking protocol, i.e. data is generated by the TCP sender when the corresponding TCP acknowledgement (ACK) is received from the client, and vice-versa [25]. On the wireless link, 802.11 mandates the use of Carrier-Sense Medium Access with Collision Avoidance (CSMA/CA) as the channel access mechanism. In practice this means that TCP ACKs, like TCP data, have to contend for the medium prior to transmission.

Since gaining access to the wireless medium is so expensive, high-speed wireless protocols like 802.11ac rely heavily on packet aggregation to achieve high performance, i.e. the ability to amortize overhead by sending more than one packet[6] on a transmit opportunity. In 802.11, there are two types of aggregation: A-MPDU (Aggregate MAC Protocol Data Unit) and A-MSDU (Aggregate MAC Service Data Unit). In practice, A-MPDUs are the primary factor in reducing CSMA/CA overhead and are required by 802.11ac. We therefore focus on A-MPDUs here and refer to the number of individual packets in an A-MPDU frame as the aggregate size.

There are three major problems in achieving large aggregates for TCP data traffic. First, a transmitter must queue packets destined for

[6] 802.11ac wave-2 allows up to 5.3ms worth of data in a single transmission.

the same receiver[7], inducing latency. Due to the additional TCP ACK latency variation over wireless links, and the self clocking aspect of TCP senders, the sender releases its subsequent data segments with similar latency variations. In turn, this results in the formation of small aggregates at the 802.11 layer, negatively impacting airtime efficiency of the wireless medium. This results in poor utilization, widening the gap between achievable and actual end-to-end TCP throughput.

Second, in multi-client scenarios, airtime efficiency is impacted by medium access contention faced by the TCP ACKs. This manifests as additional variation in latency experienced at the TCP sender, further inhibiting the achieved aggregation size. Figure 10 shows that for a moderately busy network (25 clients), it takes 85 ms on average for TCP ACKs to arrive at the sender. We have found that many client devices take over 2 ms to even begin transmitting TCP ACKs. Section 3.2.3 shows evidence of increasing network density. This, combined with increasing adoption of 802.11ac, calls for a new mechanism to achieve high TCP throughput over the air.

Third, TCP congestion control treats latency spikes as loss; when loss is detected, the congestion window shrinks in order to avoid overwhelming the network. Aggregation by the receiver causes bursts of TCP ACKs at the sender, narrowing the acceptable variation in latency before loss is assumed. Wireless latency variation is affected by fluctuating channel conditions, making it more likely that TCP congestion control will back off erroneously, reducing performance and hurting aggregation. Section 5.6.2 quantifies the achieved aggregation in our testbed.

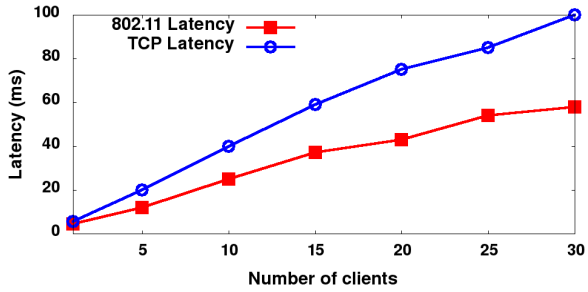

Figure 10: 802.11 latency vs. TCP latency

5.2 FastACK: Key Idea

The 802.11 standard mandates the use of layer-2 (MAC) acknowledgements (ACKs) and provides a mechanism for transmitting multiple packet ACKs in a single wireless frame. Both types are collectively referred to as 802.11 ACKs here and apply to individual packets, not the wireless frame itself. They indicate that an individual data packet was received correctly at the 802.11 layer, even in the presence of corruption affecting other packets in the aggregate.

Unlike TCP ACKs, CSMA/CA protects 802.11 ACKs, which do not contend for the wireless medium. Receivers respond with 802.11 ACKs after a Short Interframe Space (SIFS) delay.[8] Figure 10 compares the mean 802.11 latency vs. TCP latency in a testbed with a varied numbers of traffic-generating clients. 802.11 latency is measured as the interval between receiving a packet on the wire and receipt of an 802.11 ACK for that packet; this includes delay due to queuing, media contention, and retransmission. TCP latency measurement is described in Section 4.6.2.

We make two observations here. First, TCP latency is much higher than 802.11 latency, by up to 75% for 30 clients. Second, as the number of clients increase, the gap between 802.11 latency and TCP latency increases due to increased medium contention experienced by TCP ACKs from clients.

We propose that receiving of an 802.11 ACK is a reliable hint that the corresponding TCP segments will eventually be processed by the receiver's transport layer. On receipt of an 802.11 ACK from the wireless client, the AP will proactively generate a fake TCP ACK on behalf of the TCP receiver, forwarding it to the TCP sender. This results in the TCP ACKs arriving sooner at the sender, eliminating the delay variation induced by medium contention at the receiver. Since the receiver will still send a TCP ACK, these now-duplicate TCP ACKs should be suppressed by the AP.

The TCP sender is thus shielded in the reverse direction from the busy and unreliable wireless medium. The sender continues to transmit TCP data, filling up the queues at the AP. Variation in latency is also avoided, preventing the congestion window from shrinking unnecessarily. This in turn allows an 802.11ac AP to form larger aggregates, thereby increasing medium efficiency.

5.3 Related Work

Many techniques have been proposed to improve the performance of TCP over wireless medium. Most were proposed more than a decade ago, when there was significant mismatch between the wired and wireless speeds. The split connection (or indirect-TCP) approach [12, 13] involves splitting the TCP connection into two separate connections. FastACK instead is much simpler[9] and does not maintain any TCP retransmission timers. Unlike most proxy-based architectures, FastACK maintains end-to-end flow control.

Like FastACK, [14, 40], and [33] also cache packets on the AP and perform local retransmissions over wireless links. However, unlike FastACK, their motivation is to mainly reduce end-to-end TCP retransmissions. The work closest to ours is TCP-Snoop [14], but the fundamental difference lies in the motivation. FastACK aims to increase the aggregation achieved over the air, and leverages 802.11 ACKs as a hint for impending TCP ACKs. This is more aggressive than TCP-Snoop which primarily aimed to hide the wireless losses from affecting the TCP congestion window. While FastACK is complimentary to the above techniques, the main motivation here is to generate large aggregates, in order to take full advantage of high-throughput wireless protocols like 802.11ac. FastACK is also compatible with link-layer retransmission schemes [9, 27, 35]. Performance-enhancing proxies, such as ACK-spoofing [15], have also been proposed in the context of satellite links. FastACK uses a similar philosophy, but focuses on increasing aggregate size in the context of high throughput wireless links. Finally, [43] details the effects of greedy TCP receivers optimistically acknowledging TCP data. While FastACK preemptively sends a TCP ACK based on the 802.11 ACK, the techniques in [43] can be used against greedy receivers when the TCP ACKs arrive later.

[7] We use TCP *receiver* and *wireless client* interchangeably.

[8] SIFS is 10μs or 16μs based on the underlying 802.11 protocol.

[9] FastACK is $\approx 2k$ lines of code including additional handlers and debug switches.

5.4 System Design

This section details the overall design for FastACK. We discuss the case for TCP data flow and the case for 802.11 ACK flow.

TCP Data Flow: Figure 11 depicts the flow of TCP data through the AP. For each packet, p, the corresponding flow entry is determined. A decision is made to determine if the flow, f, is/should be fast-acked.[10] The state, S, for f is initialized, if needed. Table 3 shows the state information held. Depending on the sequence number of p, seq_{in}, one of these four cases can occur (note that $seq_{fack} < seq_{exp}$):

(i) if ($seq_{in} < seq_{fack}$), then p is a spurious retransmission, and it is simply dropped by the AP,

(ii) if ($seq_{fack} <= seq_{in} < seq_{exp}$), then p is an end-to-end retransmission from the TCP sender, in which case p is forwarded after priority elevation to allow it to be transmitted before the other packets at the head of the queue,

(iii) if ($seq_{in} == seq_{exp}$), then p is inserted into the local retransmission cache and forwarded downstream after updating S,

(iv) if ($seq_{in} > seq_{exp}$), it implies that a queue upstream of this AP has dropped some packets. In this case, an entry is added to $holes_{vec}$ indicating a TCP hole, followed by steps in (iii).

$holes_{vec}$	TCP holes vector
seq_{high}	highest TCP data seq_{no} seen
seq_{exp}	expected TCP data seq_{no} from the sender
seq_{fack}	last fast acked TCP data seq_{no} by the AP
seq_{TCP}	last TCP data seq_{no} ACKed at the TCP layer
q_{seq}	queue of TCP data seq_{no} waiting to be fast-ACKed

Table 3: FastACK flow state information, S

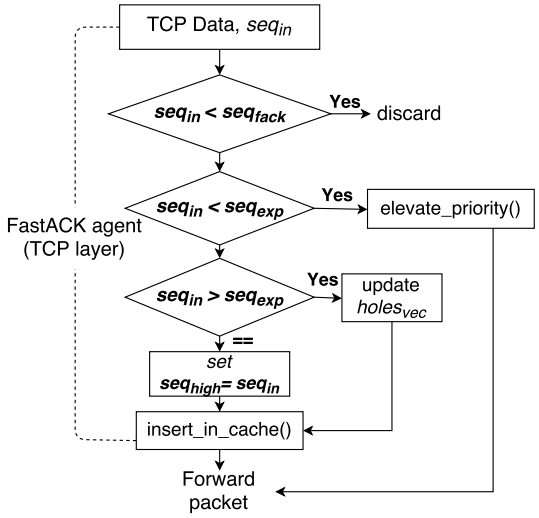

Figure 11: TCP Data flow with FastACK

802.11 ACK Flow: The overall 802.11 ACK flow is show in Figure 12. First, the 802.11 ACK, ack_i^{80211} is inspected to see if it corresponds to a FastACK TCP flow. This can be achieved by mapping the ack_i^{80211} to the data packet p_i for which it was intended. If not, then normal processing resumes. Otherwise, both the flow information S, and sequence number seq_i are extracted from p_i and sent to the FastACK agent. seq_i is the TCP sequence number corresponding to ack_i^{80211}.

FastACK agent then enqueues seq_i into q_{seq}, a queue maintained at the AP which contains all the TCP sequence numbers acknowledged at the 802.11 layer (but not fast-ACKed) in a contiguously sorted order. This is required as 802.11 ACKs often arrive in a non-contiguous order (i.e. client might acknowledge seq_i and seq_{i+2}, but not seq_{i+1} immediately due to error in reception), but TCP ACKs are cumulative[11] in nature. To ensure similar continuity before fast-ACKing TCP data, FastACK compares seq_0, the first entry in q_{seq}, with seq_{fack}, the last TCP data acknowledged at the 802.11 layer. A match ensures continuity, resulting in the AP sending a fast ACK for $seq_0 + seq_{len}$, where seq_{len} is the length (in bytes) of the TCP data segment. Next, seq_0 is removed from the q_{seq}. This process is then repeated over the other entries in q_{seq} until the continuity is broken, at which point we wait until the missing 802.11 ACKs arrive from the wireless client.

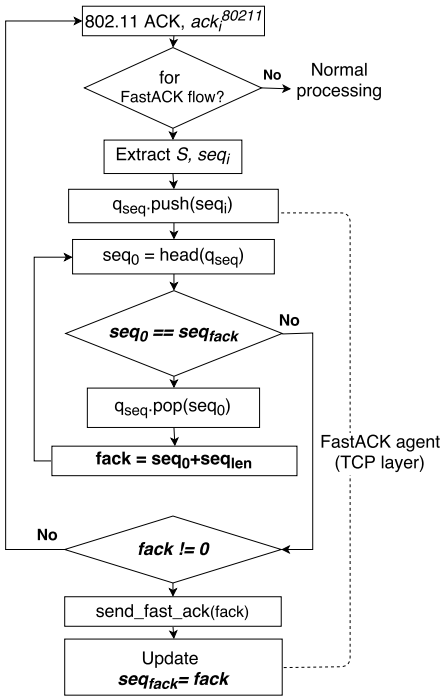

Figure 12: 802.11 ACK flow with FastACK

TCP ACK flow: When a TCP ACK with sequence number seq_{TCP} is received by the AP from the client, the processing is much simpler. Since the TCP ACK follows the 802.11 ACK and the corresponding fast ACK has already been sent to the sender, this TCP ACK is dropped by the AP. Further, the corresponding TCP DATA segments acknowledged by it, i.e. all segments with sequence numbers ($seq_i \leq seq_{TCP}$), are removed from FastACK agent's retransmission cache and the flow state S is updated.

[10] This decision can be made based on the length of the f or alternatively every flow can be marked as fast-acked.

[11] If a TCP ACK with seq_i is received, then all segments up to seq_i-1 are automatically acknowledged.

5.5 Implementation Details

Given the above design for FastACK, significant challenges emerge in terms of the actual implementation, the most notable being:

- What is the TCP retransmission strategy? What is the role of the retransmission cache?
- How to ensure that the TCP receiver does not run out of buffer space?
- There might be multiple devices (e.g. switches) between the actual TCP sender and the wireless AP, with varying buffer space. This might result in packets getting dropped on the intermittent devices, resulting in holes in the TCP sequence numbers. How can the AP handle these holes effectively?
- How is client roaming handled when in the middle of a FastACK TCP session?

5.5.1 Retransmission Strategy. There are 3 types of TCP retransmissions, (i) in response to duplicate acks, (ii) selective-ack based, and (iii) timeout-based. In FastACK, (i) and (ii) are handled by the FastACK agent on the AP, and (iii) by the TCP sender endpoint.

FastACK retransmissions. These are triggered by the FastACK agent on the AP when it receives a duplicate ACK from the client, and is the rationale behind having a retransmission cache at the AP. The obvious question here is: *"Why not let the TCP sender handle these retransmissions?"* The answer is two pronged. First, while the 802.11 ACK is a strong hint that the corresponding data segment was received at the transport layer, it is not always the case.[12]

The receipt of an 802.11 ACK at the AP triggers it to send the corresponding fast ACK to the TCP sender, thereby moving it past the current sequence number. The client on the other hand would send a duplicate ACK for this lost segment. This is a problem since the TCP sender might not hold that packet anymore in its outgoing buffer, thereby disrupting the TCP session. To avoid this, the FastACK agent inserts each data packet into its local cache before forwarding it downstream. A duplicate ACK from the client then triggers local retransmissions from this cache, on behalf of the TCP sender. Second, local retransmissions are cheaper (and low-overhead) as compared to the end-to-end retransmissions. Further, it also avoids lowering the congestion window of the TCP sender, and maintaining the high end-to-end data flow.

Selective-ack based retransmissions are handled similarly by the FastACK agent, by retransmitting the missing sequence numbers.

Timeout-based retransmissions. To reduce complexity, the FastACK agent does not maintain any retransmission timers and leaves the timeout-based retransmissions to the TCP sender endpoint. The only reason these retransmissions will occur is due to wireless loss. If no 802.11 ACKs are received, then the FastACK agent will not transmit any fast ACKs to the TCP sender. This might cause the sender's retransmission timer to timeout, resulting in end-to-end retransmissions. This reduces the congestion window at the TCP sender, resulting in fewer release of data packets. While this negatively affects aggregation in the short term, we believe it is desirable as the medium is not ideal for high throughput transmission.

5.5.2 TCP Receiver Window (rx_{win}). The rx_{win} field in the TCP ACKs from the wireless clients specifies the amount of buffer space that they have for incoming packets. This translates to the amount of data they can accept without sending an ACK back to the sender. If the TCP sender transmits more data than advertised by rx_{win}, the receiver will drop those packets due to buffer overflow, resulting in poor performance. With FastACK, data packets from the TCP sender are released at a rapid rate to fill up the driver queues to assist aggregation. While this is the desired objective, it can lead to rx_{win} overflow if not handled explicitly. The driver queues on the AP might have outstanding packets, and including the unacknowledged bytes (bytes in flight), could account for a large number of outstanding bytes, out_{bytes}, potentially exceeding rx_{win}. To avoid this, FastACK agent at the AP advertises the modified rx'_{win} in the fast ACKs to the TCP sender as follows,

$$rx'_{win} = rx_{win} - out_{bytes}, \text{where}$$
$$out_{bytes} = seq_{high} - seq_{TCP}$$

5.5.3 TCP Holes. While the TCP sender itself does not drop (or skip over) any segments, gaps in TCP seq_{num} are observed at the AP. This most likely implies some packets are getting dropped en-route between the TCP sender and the AP. FastACK combats this by emulating the wireless client and sending back a duplicate ACK back to the TCP sender corresponding to the lost TCP segments. This also avoids an end-to-end re-transmission later on, if this lost TCP segment was discovered by the wireless client instead. The FastACK agent can also use the selective acks option (if enabled) to ask for specific segments.

5.5.4 Roaming. In enterprise networks, the ability for clients to roam between APs without significant disruption to existing flows is crucial. Many enterprises now rely on controller-less networks, which complicates roaming in the presence of FastACK.

The local packet cache resolves the discrepancy between the state of the TCP sender and the proxied TCP receiver. If an 802.11 ACK from the client indicated that the packet would always be received by the transport layer, the TCP sender might receive a duplicate ACK originating from the roam-to AP in the worst case. As noted above, this is not always the case. Therefore FastACK must implement a mechanism to detect the roam and to transfer state from the roam-from AP to the roam-to AP. This mechanism is outside the scope of this paper and we omit it for brevity.

5.6 Evaluation

5.6.1 Methodology. Since FastACK has not been released yet to customer networks, we do not currently have numbers for large scale deployment. Having said that, we have exhaustively tested the performance of FastACK in our performance testbed. Figure 13 shows the testbed map comprising of 2 APs and 40 clients, and tries to mimic a realistic enterprise multi-client environment like an office. Each of the client machines is a MacBook Pro equipped with a 802.11ac wireless card supporting 3x3 MIMO, and running OSX El Capitan . The AP supports 802.11ac wave-2 and 3x3 MIMO. The TCP sender is connected to the AP via a MGig switch and is a HP Compaq Elite 8300 machine with a 2.3 GHz CPU and 16 GB of RAM running Windows 7. We run all the performance tests using ixChariot [4], which is a commonly used industry-wide live network performance testing tool. We compare FastACK with the TCP implementation running on the host and refer to it as TCP Baseline.

[12]This behavior can be observed in client devices, e.g. Macbooks running BCM43XX.

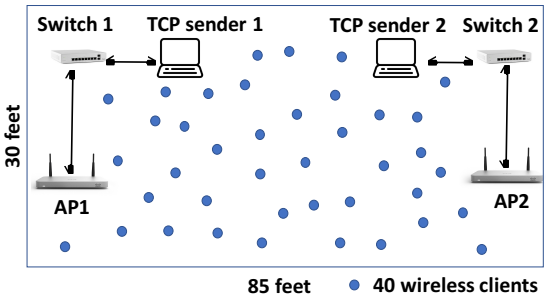

Figure 13: Testbed map

5.6.2 Micro-benchmarks. We focus on the effect of using FastACK on the resulting TCP congestion window and the achieved 802.11 aggregation size.

TCP congestion window: We use *tcp_probe* [5] to observe the sender's TCP congestion window *cwnd*. We associate 10 clients to the AP, and run one TCP flow from sender to each wireless client. Figure 14 shows the results. The key observation is that for baseline TCP, not all flows increase *cwnd* to the maximum allowed value of 770 segments[13]. On the other hand, in FastACK, the *cwnd* for each flow opens up quickly, which facilitates the release of data segments from the TCP sender when fast ACKs arrive.

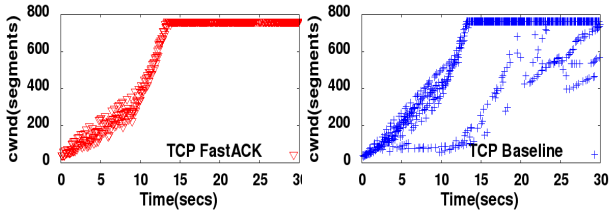

Figure 14: Comparison of TCP *cwnd* size

802.11 aggregation size: In this test, we associate 30 clients to the same AP, and run one downlink flow to each client. We then extract the average 802.11 aggregation achieved by each client and the result is shown in Figure 15. We note that FastACK enables significantly larger aggregate sizes ranging from 33 to 56, while baseline TCP's aggregation ranges from 17 to 41, an improvement of 36-94%. To get an approximate upper bound, we evaluate aggregate sizes of UDP traffic, owing to its connection-less characteristic. While we predicted UDP would approach the maximum aggregation size[14] for each flow, we suspect that limitations in the microcode running on the radio prevented us from achieving that.

5.6.3 Testbed Results. In our testbed, we run FastACK and compare against TCP baseline with varying number of clients. We measure the performance for both single and multi-AP deployments.

Aggregate throughput: Figure 16 shows the throughput comparison between baseline TCP and FastACK under varying number of connected clients. We make the following observations. (i) FastACK outperforms baseline TCP for each scenario, with throughput benefits of up to 38%, (ii) both the aggregation achieved and the throughput benefits improve in general as the number of associated clients increase. The reason is that contention on the medium increases with

[13] OS default value

[14] A-MPDU will aggregate up to 64 packets in one frame.

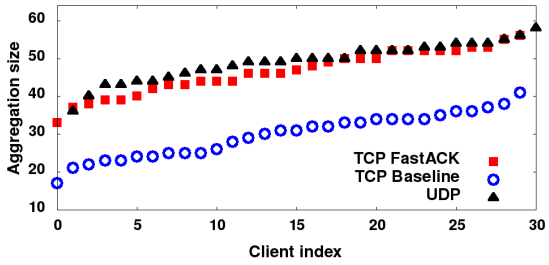

Figure 15: Comparison of 802.11 aggregation size

the number of clients (seen in Figure 10), thus giving more room for FastACK to improve upon.

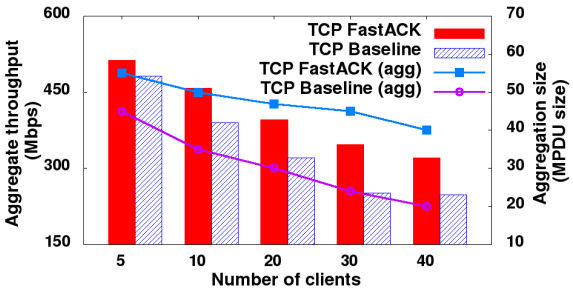

Figure 16: Comparison of aggregate client throughput

Throughput fairness: We also evaluate the effect of FastACK on throughput fairness when compared against baseline TCP. Figure 17 shows the throughput achieved by each of the clients in a 30 client test instance. The clients are sorted in increasing order of their throughput for ease of understanding. The figure shows that around 80% of the clients achieve within 70% of the maximum throughput achieved by the top-performing client. This compares favorably to baseline TCP, where only 25% of the clients are in this bracket. The Jain's fairness index [26] is 0.94 for FastACK as compared to 0.88 for TCP baseline. The reason for relatively low throughput for the lowest performing clients is the low data rates served for these clients owing to their distance from the AP. It is further interesting to note that the fairness index for the top 80% of the clients is 0.99 for FastACK vs. 0.88 for TCP baseline. This shows that FastACK does not achieve higher performance by greatly improving just a few clients, instead increasing the performance of most clients.

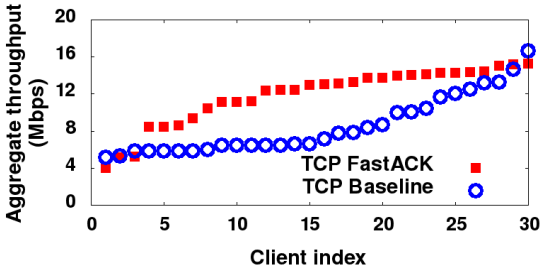

Figure 17: Comparison of throughput fairness

Multi-AP deployment: In these tests, we consider the combined throughput of multiple APs placed in the same collision domain. Given the increasing density of networks, as seen in Section 3.2.3,

this is a common scenario. We consider 2 APs here, with 10 clients associated to each AP, and run the TCP performance tests. Both APs are serving TCP traffic from a separate TCP sender, and are placed on the same wireless channel so that they need to contend for the medium using CSMA. We change the TCP algorithms on both the APs, so that we have three tests cases: (i) both AP1 and AP2 run baseline TCP, (ii) AP1 runs baseline TCP and AP2 runs FastACK, and (iii) both AP1 and AP2 run FastACK. The resulting performance is show in Figure 18.

We note that the performance improvement is highest when both AP1 and AP2 are running FastACK, resulting in an average combined throughput of 395 Mbps. Compared to (i) with an average throughput of 251 Mbps, we see a net benefit of 51%.

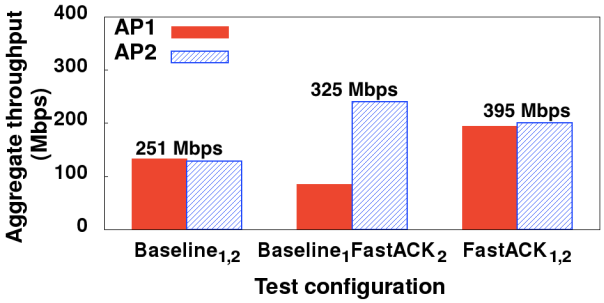

Figure 18: Multi-AP deployment results

Note that FastACK does not suffer if enabled in isolation, as shown in (iii). In fact, the throughput of AP2 (running FastACK) improves from 132 Mbps to 240 Mbps, while AP1 (running baseline) drops from 127 Mbps to 85 Mbps. Still, the combined network performance increases to 325 Mbps, an improvement over (i). The decrease in AP1's throughput is attributed to better airtime utilization by AP2 each time it gets a transmission opportunity. This shows the FastACK-enabled APs experience significant uptick in throughput even when the neighboring APs are running baseline TCP.

FastACK can be toggled at run-time and in our tests the performance characteristics described above manifest within seconds.

5.7 Discussion

While there have been a few proposals made for improving performance of TCP over the wireless medium, FastACK is, to the best of our knowledge, the first mechanism that aims to improve aggregate size. There are also further optimizations possible, especially related to fairness among flows.

FastACK relies on 802.11 ACKs, which are sometimes inaccurate due to idiosyncrasies in wireless drivers. This implies that at times, an 802.11 ACK does not always imply an impending TCP ACK. While we are yet to ascertain the root cause for this, we believe this is a client driver bug[15]. As we have seen in our tests, this inaccuracy causes unnecessary retransmissions. We expect these issues to be fixed by wireless chipset vendors soon. Also, as wireless capacity increases, the host CPU becomes a bottleneck between wired and wireless. Hardware offload engines are a popular solution to this problem but contain limited resources and are unlikely to implement the full TCP stack required for split connection.

[15]In our setup, bad hints occur ≈ 1.5%.

FastACK also relies on packet inspection, and will not work when payload is encrypted. However, in our networks, we do not currently see an extensive use of encryption techniques like IPSec. FastACK is also completely transparent to the TCP endpoints, requiring no changes to either. It is currently running in our test networks and will be deployed in production networks soon, where we will measure its impact on key real-world metrics like TCP latency and medium efficiency. We intend to release results under more large-scale scenarios, once available.

6 CONCLUSION

Enterprise wireless networks continue to grow at a rapid rate and are deployed under various scenarios. Large-scale studies focusing on wireless behavior have been infrequent but we believe such an exercise is imperative towards understanding network behavior as standards evolve. Through the unique vantage point offered by Meraki, we note the gradual shift towards high-speed wireless protocols like 802.11ac. In the last two years, client support for wider channels has increased significantly, enabling the capacity required to feed high-bandwidth applications like video, voice, and gaming. While overall utilization shows little variation from 2015, it can vary significantly between various deployment scenarios. Networks are getting crowded, which is reflected both by high per-AP associated client counts and also the number of interferers seen by APs.

We argue that throughput alone is an insufficient and difficult metric to measure, especially in large-scale deployments. We propose using a variety of metrics, such as achieved bit rates and latency experienced by different classes of traffic, as important indicators to measure the health of the network. Based on our experience deploying large-scale, enterprise networks, we also propose two techniques towards improving performance.

First, we argue for a new auto-channel assignment algorithm TurboCA, which results in lower TCP latency and improved bit rate usage in the networks we have observed, as seen by the results in Section 4. We propose that this experiment shows that bit rate usage is a promising metric for network performance.

Second, we revisit the performance of TCP in wireless networks, driven by the adoption of 802.11ac, and propose FastACK, discussed in Section 5, as a method to achieve better aggregation over-the-air. In initial testbed experiments, we have seen up to 35% increase in throughput for single APs and up to 51% in multi-AP deployments. These results are promising and we expect to see similar performance improvement in the field.

Both techniques we propose have evolved from the results of our large-scale study, which shows the importance of such an exercise towards shaping the networks of the future.

ACKNOWLEDGEMENTS

We thank our shepherd, Aaron Schulman, and the anonymous reviewers for their constructive feedback towards improving this paper. We also appreciate Nathan Brahms, whose insightful feedback contributed to better readability of this paper. Finally, we also thank the engineers at Cisco Meraki who have contributed significantly towards building the data collection infrastructure, and making this study possible.

REFERENCES

[1] [n. d.]. 802.11k-2008 - IEEE Standard for Information technology– Local and metropolitan area networks. https://standards.ieee.org/findstds/standard/802.11k-2008.html. ([n. d.]).

[2] [n. d.]. FCC. https://www.fcc.gov/. ([n. d.]).

[3] [n. d.]. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. http://www.gartner.com/newsroom/id/3598917. ([n. d.]).

[4] [n. d.]. IxChariot. https://www.ixiacom.com/products/ixchariot. ([n. d.]).

[5] [n. d.]. Linux Foundation wiki. https://wiki.linuxfoundation.org/networking/tcpprobe. ([n. d.]).

[6] [n. d.]. The Zettabyte Era - Trends and Analysis - Cisco White paper . Available at http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf. ([n. d.]).

[7] M. Achanta. 2006. Method and apparatus for least congested channel scan for wireless access points. (April 6 2006). https://www.google.com/patents/US20060072602 US Patent App. 10/959,446.

[8] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li. 2005. Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom '05)*. ACM, New York, NY, USA, 58–72. https://doi.org/10.1145/1080829.1080836

[9] Ender Ayanoglu, Sanjoy Paul, Thomas F. LaPorta, Krishan K. Sabnani, and Richard D. Gitlin. 1995. AIRMAIL: A Link-layer Protocol for Wireless Networks. *Wirel. Netw.* 1, 1 (Feb. 1995), 47–60. https://doi.org/10.1007/BF01196258

[10] Paramvir Bahl, Ranveer Chandra, and John Dunagan. 2004. SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-hoc Wireless Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)*. ACM, New York, NY, USA, 216–230. https://doi.org/10.1145/1023720.1023742

[11] Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, Rohan Murty, and Matt Welsh. 2009. White Space Networking with Wi-fi Like Connectivity. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. ACM, New York, NY, USA, 27–38. https://doi.org/10.1145/1592568.1592573

[12] A. Bakre and B. R. Badrinath. 1995. I-TCP: indirect TCP for mobile hosts. In *Proceedings of 15th International Conference on Distributed Computing Systems*. 136–143. https://doi.org/10.1109/ICDCS.1995.500012

[13] Ajay V. Bakre and B. R. Badrinath. 1995. Handoff and Systems Support for Indirect TCP/IP. In *Proceedings of the 2Nd Symposium on Mobile and Location-Independent Computing (MLICS '95)*. USENIX Association, Berkeley, CA, USA, 11–24. http://dl.acm.org/citation.cfm?id=646407.692379

[14] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H Katz. 1995. Improving TCP/IP performance over wireless networks. In *Proceedings of the 1st annual international conference on Mobile computing and networking*. ACM, 2–11.

[15] Vijay G Bharadwaj. 1999. *Improving TCP performance over high-bandwidth geostationary satellite links*. Ph.D. Dissertation.

[16] Apurv Bhartia, Deeparnab Chakrabarty, Krishna Chintalapudi, Lili Qiu, Bozidar Radunovic, and Ramachandran Ramjee. 2016. IQ-Hopping: Distributed Oblivious Channel Selection for Wireless Networks. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '16)*. ACM, New York, NY, USA, 81–90. https://doi.org/10.1145/2942358.2942376

[17] G. Bianchi. 2006. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE J.Sel. A. Commun.* 18, 3 (Sept. 2006), 535–547. https://doi.org/10.1109/49.840210

[18] Sanjit Biswas, John Bicket, Edmund Wong, Raluca Musaloiu-E, Apurv Bhartia, and Dan Aguayo. 2015. Large-scale Measurements of Wireless Network Behavior. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 153–165. https://doi.org/10.1145/2785956.2787489

[19] R Michael Buehrer. 2006. Code division multiple access (CDMA). *Synthesis Lectures on Communications* 1, 1 (2006), 1–192.

[20] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. 2006. Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*. ACM, New York, NY, USA, 39–50. https://doi.org/10.1145/1159913.1159920

[21] S. Chieochan, E. Hossain, and J. Diamond. 2010. Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey. *IEEE Communications Surveys Tutorials* 12, 1 (First 2010), 124–136. https://doi.org/10.1109/SURV.2010.020110.00047

[22] C. Cordeiro and K. Challapali. 2007. C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Networks. In *Proceedings of the 2007 2Nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*. IEEE Computer Society, Washington, DC, USA, 147–157. https://doi.org/10.1109/DYSPAN.2007.27

[23] Aditya Dhananjay, Hui Zhang, Jinyang Li, and Lakshminarayanan Subramanian. 2009. Practical, Distributed Channel Assignment and Routing in Dual-radio Mesh Networks. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. ACM, New York, NY, USA, 99–110. https://doi.org/10.1145/1592568.1592581

[24] A. Hills. 2001. Large-scale wireless LAN design. *IEEE Communications Magazine* 39, 11 (Nov 2001), 98–107. https://doi.org/10.1109/35.965365

[25] V. Jacobson. 1988. Congestion Avoidance and Control. In *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM '88)*. ACM, New York, NY, USA, 314–329. https://doi.org/10.1145/52324.52356

[26] Raj Jain, Dah-Ming Chiu, and W. Hawe. 1998. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *CoRR* cs.NI/9809099 (1998). http://arxiv.org/abs/cs.NI/9809099

[27] Phil Karn. 1993. The Qualcomm CDMA Digital Cellular System.. In *Symposium on Mobile and Location-Independent Computing*.

[28] B. J. Ko, V. Misra, J. Padhye, and D. Rubenstein. 2007. Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks. In *2007 IEEE Wireless Communications and Networking Conference*. 3978–3983. https://doi.org/10.1109/WCNC.2007.727

[29] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The Click Modular Router. *ACM Trans. Comput. Syst.* 18, 3 (Aug. 2000), 263–297. https://doi.org/10.1145/354871.354874

[30] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. 2006. Analyzing the MAC-level Behavior of Wireless Networks in the Wild. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*. ACM, New York, NY, USA, 75–86. https://doi.org/10.1145/1159913.1159923

[31] Arunesh Mishra, Suman Banerjee, and William Arbaugh. 2005. Weighted Coloring Based Channel Assignment for WLANs. *SIGMOBILE Mob. Comput. Commun. Rev.* 9, 3 (July 2005), 19–31. https://doi.org/10.1145/1094549.1094554

[32] Arunesh Mishra, Vivek Shrivastava, Dheeraj Agrawal, Suman Banerjee, and Samrat Ganguly. 2006. Distributed Channel Management in Uncoordinated Wireless Environments. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom '06)*. ACM, New York, NY, USA, 170–181. https://doi.org/10.1145/1161089.1161109

[33] Mehta Miten and Vaidya Nitin. 1998. *Delayed Duplicate-Acknowledgements: A Proposal to Improve Performance of TCP on Wireless Links*. Technical Report. College Station, TX, USA.

[34] Thomas Moscibroda, Ranveer Chandra, Yunnan Wu, Sudipta Sengupta, Paramvir Bahl, and Yuan Yuan. 2008. Load-aware spectrum distribution in wireless LANs. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*. IEEE, 137–146.

[35] Sanjiv Nanda, Richard Ejzak, and Bharat T Doshi. 1994. A retransmission scheme for circuit-mode data on wireless links. *IEEE Journal on Selected Areas in Communications* 12, 8 (1994), 1338–1352.

[36] Ashish Patro, Srinivas Govindan, and Suman Banerjee. 2013. Observing Home Wireless Experience Through WiFi APs. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13)*. ACM, New York, NY, USA, 339–350. https://doi.org/10.1145/2500423.2500448

[37] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot. 2006. Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. 1–12. https://doi.org/10.1109/INFOCOM.2006.177

[38] A. Raniwala and Tzi cker Chiueh. 2005. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Vol. 3. 2223–2234 vol. 3. https://doi.org/10.1109/INFCOM.2005.1498497

[39] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh. 2004. Centralized Channel Assignment and Routing Algorithms for Multi-channel Wireless Mesh Networks. *SIGMOBILE Mob. Comput. Commun. Rev.* 8, 2 (April 2004), 50–65. https://doi.org/10.1145/997122.997130

[40] Karu Ratnam and Ibrahim Matta. 1998. WTCP: An efficient transmission control protocol for networks with wireless links. In *Proceedings of Third IEEE Symposium on Computer and Communications (IEEE ISCC)*.

[41] Shravan Rayanchu, Vivek Shrivastava, Suman Banerjee, and Ranveer Chandra. 2011. FLUID: Improving Throughputs in Enterprise Wireless Lans Through Flexible Channelization. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom '11)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/2030613.2030615

[42] Sean Rhea, Eric Wang, Edmund Wong, Ethan Atkins, and Nat Storer. 2017. LittleTable: A Time-Series Database and Its Uses. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM, New York, NY, USA, 125–138. https://doi.org/10.1145/3035918.3056102

[43] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. 1999. TCP Congestion Control with a Misbehaving Receiver. *SIGCOMM Comput. Commun. Rev.* 29, 5 (Oct. 1999), 71–78. https://doi.org/10.1145/505696.505704

[44] Vivek Shrivastava, Shravan Rayanchu, Suman Banerjee, and Konstantina Papagiannaki. 2011. PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11)*. USENIX Association, Berkeley, CA, USA, 337–350. http://dl.acm.org/citation.cfm?id=1972457.1972492

[45] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda. 2016. Characterizing and Improving WiFi Latency in Large-Scale Operational Networks. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16)*. ACM, New York, NY, USA, 347–360. https://doi.org/10.1145/2906388.2906393

[46] A. Tzamaloukas and J. J. Garcia-Luna-Aceves. 2001. A receiver-initiated collision-avoidance protocol for multi-channel networks. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Vol. 1. 189–198 vol.1. https://doi.org/10.1109/INFCOM.2001.916701

[47] Mythili Vutukuru, Hari Balakrishnan, and Kyle Jamieson. 2009. Cross-layer Wireless Bit Rate Adaptation. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/1592568.1592571

[48] P. Wertz, M. Sauter, F. A. Landstorfer, G. Wolfle, and R. Hoppe. 2004. Automatic optimization algorithms for the planning of wireless local area networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, Vol. 4. 3010–3014 Vol. 4. https://doi.org/10.1109/VETECF.2004.1400613

[49] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. 2006. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom '06)*. ACM, New York, NY, USA, 146–157. https://doi.org/10.1145/1161089.1161107

[50] Zhenyu Yang and J. J. Garcia-Luna-Aceves. 1999. Hop-reservation multiple access (HRMA) for ad-hoc networks. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 1. 194–201 vol.1. https://doi.org/10.1109/INFCOM.1999.749268

[51] Yiannis Yiakoumis, Manu Bansal, Adam Covington, Johan van Reijendam, Sachin Katti, and Nick McKeown. 2014. BeHop: A Testbed for Dense WiFi Networks. In *Proceedings of the 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH '14)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2643230.2643233