BUILDING BLOCKS TO UNDERSTAND WIRELESS EXPERIENCE

by

Ashish Patro

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2015

Date of final oral examination: 07/14/2015

The dissertation is approved by the following members of the Final Oral Committee:

Suman Banerjee, Professor, Computer Sciences
Aditya Akella, Associate Professor, Computer Sciences
Paul Barford, Professor, Computer Sciences
Mike Swift, Associate Professor, Computer Sciences
Xinyu Zhang, Assistant Professor, Electrical and Computer Engineering

To my parents, Sangitarani Patra and Una Ashoka Patro,
and my brother, Abhishek Patro.

ACKNOWLEDGMENTS

I joined UW Madison right after my undergraduate studies thinking about getting a Master's degree but ended up doing a PhD. Looking back at this experience, I feel that it was one of the best decisions I made. I've been fortunate enough to have met a number of awesome people during this journey and this is my shoutout to all of them! First of all, I'd like to thank my PhD advisor, Suman Banerjee. I sincerely thank him for giving me immense freedom to explore interesting and crazy research projects and always being there to give me valuable advise. His insightful comments and critical feedback have played a major role in shaping this dissertation. Besides research, I also appreciate his generous help with numerous professional and personal matters.

I'm grateful for having the opportunity to work with many great mentors during my graduate studies. I thank Srikanth Kandula for his mentoring during my internship at Microsoft Research. It was a fun experience to work with him. I also thank Victor Bahl, Amar Phanishayee and Mohammad Shoaib from Microsoft Research for their guidance. I also thank Arunesh Mishra for being a great mentor during my first graduate internship at Google. In addition to great mentoring, his advise was instrumental in helping me decide to get a PhD. I would also like to thank Aditya Akella, Paul Barford, Mike Swift and Xinyu Zhang for their insightful suggestions and comments that greatly improved this thesis. I also thank Rodrigo Rodrigues, who was my mentor during my undergraduate internship at Max Plank Institute for Software Systems, Saarbrucken. I highly value his advise and help during the early years of my graduate studies.

Graduate school wouldn't have been as much fun without my awesome colleagues at Madison. I would like to thank Shravan Rayanchu, Vivek Shrivastava, Sayandeep Sen and Ashok Anand for their support during my initial years at graduate school. I also thank my fellow graduate students: Yadi Ma, Michael Griepentrog, Srinivas Govindan, Tan Zhang, Dale Willis, Prakhar Panwaria, Vishal Siva Subramanian, Jordan Walker and Fatemeh Panahi. I had the honor of working with and mentoring some really smart undergraduate

students: Nathan Moeller, Nick Butch, Wes Miller and Mickey Barboi. Thank you all!

Finally, I would like to convey my heartfelt gratitude to my parents Sangitarani Patra and Una Ashoka Patro, and my brother Abhishek for their unconditional support and boundless love. This dissertation is dedicated to them.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

BUILDING BLOCKS TO UNDERSTAND WIRELESS EXPERIENCE

Ashish Patro

Under the supervision of Professor Suman Banerjee
At the University of Wisconsin-Madison

In recent years, there has been a rapid growth in the penetration of wireless networks such as 3G and 4G cellular networks (e.g. HSPA, EVDO, LTE) and WiFi networks. Simultaneously, there has been a rapid growth in the use of wireless-enabled devices such as laptops, netbooks, smartphones and tablets. As a consequence of these developments, the complex interactions resulting from the diversity of applications, client devices and wireless networks in this ecosystem cause many problems that may degrade end users' experience.

To tackle this problem, this dissertation makes contributions towards building systems and techniques that leverage various "vantage points", i.e., applications, client devices and wireless network infrastructure (e.g., WiFi Access Points) to estimate and manage various RF, network and application level parameters.

We start by building and deploying a measurement infrastructure named WiSe, to diagnose problems in residential WiFi networks using home Access Points as a vantage point. Due to the ad-hoc deployment of Access Points as well as lack of user expertise, residential WiFi networks experience many problems – inefficient spectrum utilization, channel congestion, WiFi and non-WiFi interference. We build an analytics toolkit for the WiSe infrastructure to characterize these wireless problems and estimate overall WiFi link performance (e.g., available TCP throughput). To mitigate these problems, we then develop the COAP framework to enable cooperation and coordination between neighboring home Access Points through a cloud-based controller. We propose vendor-neutral open APIs for home WiFi Access Points for this purpose.

In the last part of the thesis, we explore the use of client devices and mobile applications as vantage points to understand wireless experience. We develop the Insight toolkit library, which developers can bundle with their mobile applications. Insight uses the client device and application vantage point to

analyze and quantify the impact of device, location, resource consumption and network performance on end user experience.

We believe that these tools and techniques are useful building blocks in diagnosing the causes of poor user experience over wireless networks. By leveraging insights from different vantage points, these tools can provide a more unified view of users' overall wireless experience.

Suman Banerjee

1 INTRODUCTION

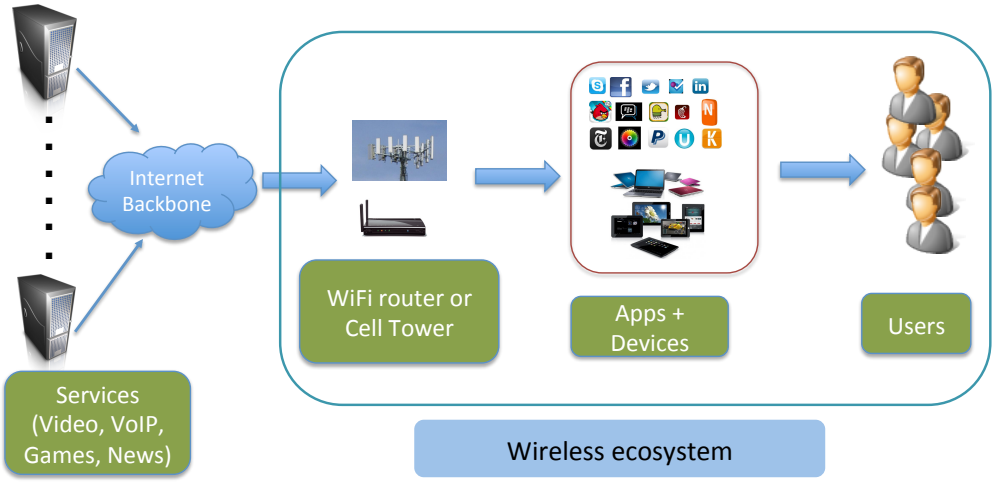

Figure 1.1: At a high level, this is how a typical wireless deployment looks like today. This thesis focuses on the *wireless ecosystem* of the wireless network infrastructure (e.g., WiFi Access Points), client devices (e.g., smartphones and tablets) and applications (e.g., games, video).

The last decade has seen a rapid growth in Internet usage over wireless networks [35]. This growth has been fueled by the increasing penetration of WiFi and 3G/4G cellular networks as well as the widespread adoption of mobile platforms (e.g., Android [46] and iOS [23]) and handheld devices such as smartphones and tablets. Users access multiple services such as video, music, games, social media and VoIP using applications installed on client devices, which in turn are connected to the Internet over the wireless infrastructure consisting of WiFi Access Points or cell towers. User expectations are rapidly rising in terms of capabilities, speed, reliability and responsiveness from this *ecosystem* of wireless networks, mobile devices and applications (Figure 1.1).

But, the increasing diversity of wireless networks, devices and applications is increasing the complexity of maintaining good user experience. Following is a common problem scenario – a user is playing a multiplayer game on his smartphone connected to a residential WiFi Access Point, but a high lag in

response time is degrading the user's gaming experience. Diagnosing this problem can prove to be a hard challenge because it may be caused due to issues at either the home WiFi Access Point, his smartphone or the gaming application itself. Following is a more detailed description of the possible issues at these three components:

- **At the wireless infrastructure (e.g., WiFi Access Points):** In residential areas such as multi-dwelling units, a large number of independently owned WiFi Access Points operate in the close vicinity of each other. During periods of high network activity, there can be high channel congestion which decreases the available TCP/UDP throughput for each individual Access Point. These WiFi Access Points can also experience hidden terminal interference resulting in high MAC layer packet losses at the WiFi client [89]. The resulting packet retransmissions can increase the WiFi link latency as well as reduce available throughput. There can also be cross-technology interference from nearby non-WiFi devices [84] such as microwave ovens, cordless phones, Bluetooth headsets, game controllers etc. These devices operate on the same unlicensed spectrum as WiFi networks and can also cause channel congestion and/or high packet losses.

- **At the client device (e.g., smartphone, tablet):** Problems at client devices such as low CPU and memory capacity or high resource usage (CPU, memory) by background applications can degrade user experience. Also, there may be WiFi network problems at the user's current location (e.g., weak WiFi signal) due to the device being far away from the WiFi Access Point. These network coverage issues can increase the link latency and reduce throughput for the affected client. But this problem may resolve itself when the client moves to another location. Such intermittent issues can be more frustrating for the user to identify and diagnose.

- **At the mobile application (e.g., games, video):** Mobile applications today range from simple ones (e.g., an alarm clock) to more sophisticated ones such as multiplayer games, social networking and banking. User

experience can be poor due to UI design problems (e.g., improper rendering across different screen sizes), poor crash handling and feature specific problems, such as a poor caching strategy.

One or a combination of such issues can lead to the example problem scenario specified earlier. The need for systems and tools to diagnose such problems is the main motivation for this thesis.

1.1 FOCUS OF THIS THESIS

As discussed in the earlier example, poor wireless experience can occur due to problems at different components, i.e., the wireless network infrastructure (e.g., WiFi Access Points), client devices (e.g., smartphones and tablets) and mobile applications (e.g., games). It indicates the need for systems and tools to provide users, ISPs and application developers with greater visibility into the factors that lead to poor experience at the end-user in this ecosystem.

In this dissertation, we posit that it is necessary to design and deploy tools and mechanisms that leverage the appropriate "vantage points" to capture and manage experience at various levels: application layer, network layer and the physical layer. But, getting access to one or some of these vantage points may not be feasible in different scenarios. For example, third-party mobile application developers can only use client devices installed with their specific applications as a vantage point. From these developers' perspective, tools that can leverage this vantage point to understand the impact of various network, device, location and time specific factors on user experience are the most desirable solution. In the case of residential ISPs, the WiFi Access Points (APs) or gateways are the best vantage points available to them. Residential WiFi APs can passively capture the wireless network performance experienced by various WiFi-enabled devices (e.g., smartphones, laptops, wireless TVs etc.) within homes. *Thus, given a specific vantage point and the associated limitations, this dissertation explores techniques to better understand and improve users' wireless experience.*

Problem statement and solution approach

Problem statement: Given an ecosystem of wireless network infrastructure (e.g., WiFi Access Points), client devices (e.g., smartphones, tablets) and mobile applications, this thesis tries to address the following question —

> *For a specific "vantage point" (e.g., a WiFi Access Point, client device or mobile application), how can we design systems and tools that estimate and manage various RF, network and application level parameters to better understand and improve users' wireless experience?*

More specifically, we investigated the design of systems and tools that run on wireless network infrastructure (e.g., home WiFi Access Points), smartphones and tablets to capture different parameters and diagnose causes of poor perceived user experience.

Figure 1.2 demonstrates our approach towards solving this problem by building tools that leverage WiFi Access Points, client devices and mobile applications as "vantage points". We describe these solutions in more detail next.

Our solution: We break down the solution space into two components. First, we focus on building solutions that leverage wireless network infrastructure, i.e. WiFi Access Points (APs) in dense residential multi-dwelling units, such as apartment buildings. We start by building the WiSe infrastructure (Chapter 2). For this purpose, we developed an WiFi Access Point platform (built using off-the-shelf components) and deployed them in 30 homes for more than 9 months. WiSe uses home APs to collect a diverse set of statistics at the PHY layer (airtime utilization, non-WiFi activity), MAC layer (link quality, local and external WiFi activity) as well as coarse-grained flow level statistics. We also incorporate passive techniques into WiSe to measure WiFi and non-WiFi interference and estimate overall link performance by predicting TCP throughput.

This study motivates the need to mitigate wireless performance issues, reduce interference and improve spectrum utilization in dense residential WiFi deployments. Thus, we develop the COAP framework to enable cloud based centralized management of neighboring home WiFi APs. COAP uses the

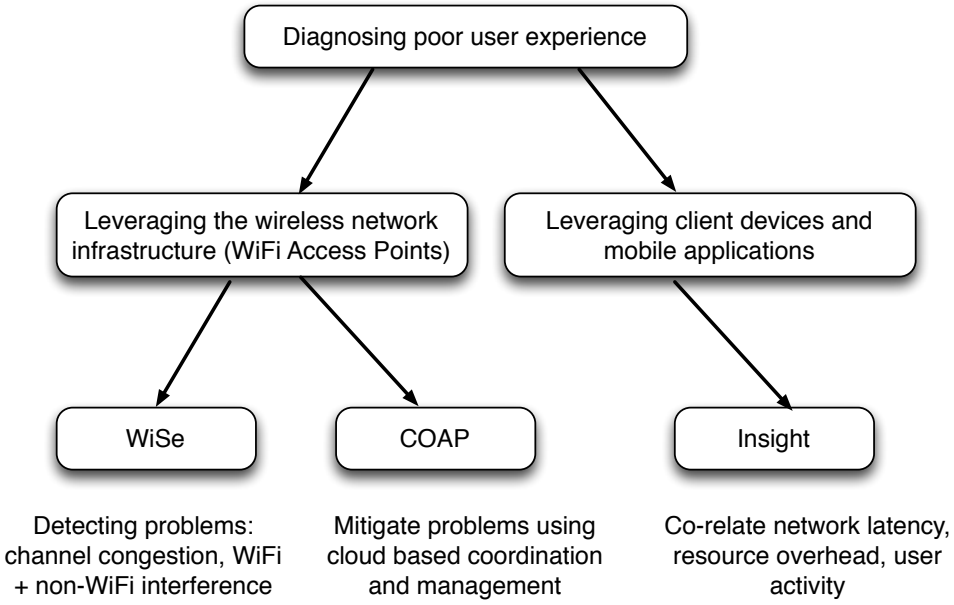

Figure 1.2: Our approach to the solution in this thesis. WiSe and COAP are systems implemented over WiFi Access Point infrastructure in residential environments to diagnose and mitigate infrastructure related problems such as channel congestion, WiFi and non-WiFi device interference. The Insight toolkit is inserted by developers into their iOS/Android mobile applications and thus leverages client devices and mobile application as vantage points. Insight uses this vantage point to analyze the impact of client device, network, performance and application related factors on user experience.

Software Defined Networking (SDN) paradigm to propose open APIs that can be implemented by commodity Access Point vendors to enable cooperation and coordination between nearby home APs. The framework allows COAP APs to share various types of information with the centralized controller – interference (e.g., non-WiFi activity), traffic phenomenon and various flow contexts, and in turn receive instructions – configuration parameters (e.g., channel) and transmission parameters (through coarse-grained schedules and throttling parameters).

In the last part of the thesis, we develop the Insight toolkit that leverages the WiFi client device and application vantage points to collect data about cellular/WiFi network performance, device characteristics, application UI, resource consumption (e.g., CPU, memory and battery) and user activity. Insight is implemented as library for third-party mobile application application developers (e.g., a *.jar file for Android) and gets installed on the client device along with the mobile application. We co-relate user activity within the application with the aforementioned parameters to understand their impact on user experience. We now describe each of these systems in more detail in the following sections.

1.2 OBSERVING RESIDENTIAL WIRELESS EXPERIENCE THROUGH WIFI ACCESS POINTS

We start by exploring the wireless problems experienced by residential WiFi networks using home WiFi Access Points as a vantage point. A typical home network today has different kinds of WiFi enabled devices such as laptops, handhelds (smartphones and tablets), printers, entertainment devices including game controllers (XBox, Wii), interactive television (e.g., Apple TV, Google TV), wireless HDTV transmitters [48], and wireless-based security cameras and other security systems. In addition, there is a plethora of other diverse wireless appliances in our homes — various cordless handsets, Bluetooth headsets, various types of sensors and actuators, and even microwave ovens that operate in the same unlicensed bands. Over the last decade or so, there has been a few individual studies that have tried to quantify the experience and performance of wireless networks. Most famous among them include Jigsaw for WiFi activities in a university campus [34], and a few in home environments [38, 74]. The approaches have attempted to understand and evaluate specific wireless characteristics through passive observations by carefully deployed sniffers.

We take a significantly different approach by building OpenWrt [12] based WiFi APs, enhanced with our network measurement and analysis software. We collect wireless-specific measurements *inline* using these WiFi APs as vantage points, by deploying these Access Points in 30 residential units for 9 months. Together, they form a unique type of wireless measurement infrastructure

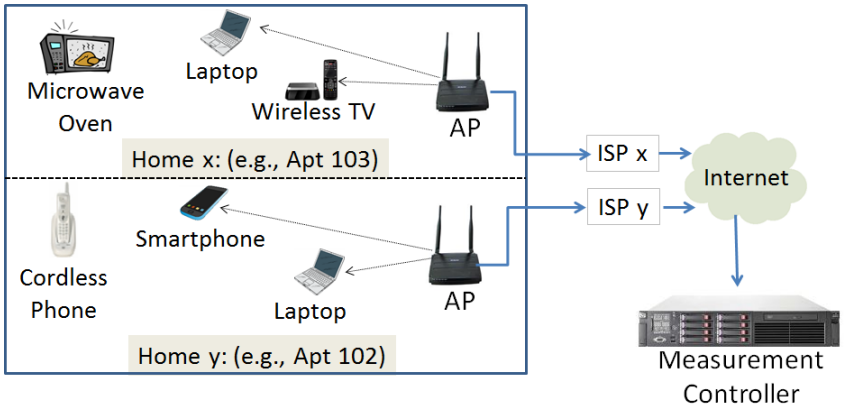

Figure 1.3: WiSe measurement framework.

that we call WiSe (Wireless infrastructure for inline Sensing). Unlike out-of-band wireless sniffers that observe a subset of the traffic, *our WiFi APs can exactly observe all traffic to and from its clients,* correlate them with wireless performance, and can hence conduct types of measurements not possible otherwise (Figure 1.3).

The WiSe WiFi APs are equipped with dual WiFi NICs — one provides the AP functionality to clients, and the other serves some unique and additional measurement functions when necessary.

In a way, this measurement effort is related to the recent BISmark project [94] and the SamKnows effort [15] which focused on measurements of broadband ISP behavior across different communities. The key different between BISmark, SamKnows, and WiSe is that the first two projects primarily focus on characterization of the wired Internet path from the ISP's network into the home and not on the wireless network properties, while WiSe focuses solely on the wireless network properties and not on the rest of the ISP path. Our measurement infrastructure is quite stylized both in software and hardware to our specific wireless measurement goals.

In particular, we wanted to answer a number of questions such as: (i) how often do home WiFi networks provide us with good, mediocre, or bad performances? (ii) when the wireless performance is bad, what are likely

causes of such experience and how persistent are these experiences? (iii) how much interference do we see in these environments from a) other WiFi sources and b) various non-WiFi sources?, and (iv) how do users configure their WiFi networks and what might be the impact of some different configuration choices on observed performance?

To evaluate and answer all of these questions, one of the first requirements is to define a wireless performance metric that can capture the overall "goodness" of the network. In general, many different performance metrics are possible. In defining a metric, we posited four desirable properties for our purposes: (i) it should be easy to observe this metric from any passive vantage point and especially so for a WiFi AP; (ii) the computation of the metric should not require us to impose additional load on the wireless network; (iii) the metric should be relevant to only the wireless part of any user's end-to-end path and ignores potential latencies and bottlenecks that exist uplink from a WiFi AP at home; and (iv) it is application-agnostic.

As a result of this, we cannot use certain simplistic measures as our metric, e.g., end-to-end throughputs of all flows of different users — which is a function of upstream wired bottlenecks and also the individual applications that generate these flows. Instead, we propose a specific metric that can provide the following estimate — given the current wireless environmental conditions, what is the likely TCP throughput of a new flow between the AP and the different WiFi clients connected to the AP? Given that TCP elastically adjusts its throughputs to current conditions and bottlenecks, this metric would require to consider the busy time of the channel, signal quality to and from the client, consequent loss properties, and re-transmissions, and also overall airtime expended by the AP to and from its different clients. We call this metric, *WiFi-based TCP throughput* or Witt. Note that in computing our metric we do not conduct active TCP measurements, but instead estimate it *through a model based on various passive observations.*

For this study, we deployed 30 WiSe APs across homes in Madison, Wisconsin over a period of more than 9 months. Through our measurements, we observed that while most of the WiFi clients experience moderate to good performance, poor performance plagues these environments about 2.1%

of the time. The major cause of poor network performance (e.g., airtime congestion, poor signal strengths) was dependent on the environment. Some APs experienced short periods of high impact interference (>50% throughput degradation) from external sources (e.g., microwave ovens). Also, majority of APs at homes tend to have static channel configurations over time, indicating that these APs do not adapt to interference or contention experienced by APs due to external sources.

1.3 SOFTWARE DEFINED FRAMEWORK FOR MANAGING RESIDENTIAL WIFI ACCESS POINTS

In the next piece of work, we explore mechanisms to mitigate problems experience by residential WiFi networks . Standalone Access Points today are supplied by a number of diverse vendors — Linksys, Netgear, DLink, Belkin, to name a few. Given their central role in home networks, the performance and experience of users at homes depend centrally on efficient and dynamic configuration of these APs. In this work, we propose a *simple vendor-neutral API* that should be implemented by commodity home wireless APs to enable a cloud-based management service that enables coordination between neighboring home APs, provides better performance, and reduces burden on users.

In our proposed solution, called COAP (Coordination framework for Open APs), participating commodity wireless APs (across different vendors) are configured to securely connect to a cloud-based controller. The controller provides all necessary management services that can be operated by a third-party (potentially distinct from the individual ISPs). In general, it is desirable that all nearby APs use the same controller service for the management function. This architecture is particularly applicable to large apartment buildings where one could envision that the apartment management contract with a single controller service (e.g., through a fixed annual fee) and all residents are asked to utilize the designated controller service within the building. This service would be no different than many other utilities, e.g., water, electricity, etc., which is arranged by the apartment management. Individual residents can also pick different controller services to realize many of our proposed benefits.

However, some advanced features, e.g., interference management and mitigation, are better served if neighboring APs participate through the same service.

Proprietary cloud-based solutions to manage *individual* home APs [37] exist today. However, such solutions do not attempt to leverage cooperation or coordination between neighboring APs. The COAP framework focuses on techniques to share various types of spectrum interference phenomenon and to enable cooperation between neighboring, participating APs and leads to significantly improved management, not otherwise possible. These techniques range from aggregating airtime utilization information from neighboring COAP APs to effect efficient channel assignments (§3.3), using co-operative transmission schedules for interference mitigation (§3.3), to using "learning-based techniques" (§3.4) to enable more sophisticated AP configuration algorithms. For example, if a non-WiFi device (e.g., microwave oven) is frequently used at certain times of day, the controller can *pre-emptively* configure neighboring COAP APs to other channels and mitigate interference (§3.4). In another possible scenario, if a client device consistently consumes high volume traffic (e.g., HD movies) at certain times of day the controller can *pre-emptively* configure the neighboring COAP APs to use other channels and mitigate high channel contention.

Most enterprise WLAN solutions today (including some recent SDN-style efforts [95]) adopt a centralized approach for managing a set of homogeneous APs in a uniform and coordinated manner. Further, a few commercial solutions, e.g., Meraki [64], provide vendor-specific proprietary cloud-managed service for APs in enterprise environments. The specific mechanisms used in all such solutions are based on proprietary signaling that primarily manage hardware of a single vendor alone. We believe that a SDN based approach (using an open API) is also important in home environments where each wireless neighborhood has a diverse set of APs. But, unlike enterprise environments, homogeneity in such environments is likely hard to achieve. Furthermore, enterprise WLANs consist of a tightly managed control and data plane with very low latencies between the controller and the APs (order of micro-seconds). For home APs, it is only possible to create a loosely coupled control and data plane due to the Internet scale latencies between the APs and controller (tens of milliseconds).

Finally, with the growing demands on in-home wireless networks, especially with the dominance of HD media streaming, the need for coordination between neighbors will continue to grow.

Through a combination of real-world AP deployments and controlled experiments, we motivate the benefits of the COAP framework by describing two applications that benefit from co-operation between nearby residential APs: (i) efficient channel assignments resulting in *upto 47% reduction* in congestion (§3.3), (ii) mitigating wireless interference issues due to channel congestion or hidden terminal style interference resulting in more efficient airtime utilization due to *50% reduction in MAC losses* and improved quality for HTTP video flows (§3.3).

We also study how learning about prior non-WiFi and WiFi activity (e.g., client and traffic properties) using data collected by COAP cloud controller can help in predicting future activity. For example, *using 5 days of microwave oven activity* data from APs in an apartment building lead to high predictability of future activity (§3.4). Also, using client device and traffic type information (§3.4) resulted in *at least* $2\times$ *better accuracy* in predicting wireless traffic properties (e.g., session lengths). Such information can enable the COAP controller to pre-emptively configure home APs to prevent interference.

1.4 CAPTURING MOBILE APPLICATION EXPERIENCE

While WiSe and COAP leverage WiFi Access Points, we now describe a complementary system, Insight, that uses client devices and mobile applications as vantage points to understand users' *mobile application experience*. The last decade has seen massive growth in usage of mobile applications with hundreds of thousands of applications populating different app stores and billions of downloads of these applications. Once released, these applications get deployed across a large range of devices, users, operating environments, and may cater to different social as well as cultural usage patterns. Insight attempts to understand the experience of these mobile applications once they are deployed in the wild amongst thousands of users.

In the recent years, researchers have adopted various approaches to study the usage of mobile applications. Some studies have instrumented smartphone devices of a controlled set of users to understand application usage patterns (e.g., sessions, network, context) [41, 81, 87]. Other studies have deployed standalone measurement apps for the mobile platforms. For example, 3GTest [49] uses a standalone app that allows users to run network performance tests from their current locations, while AppSensor [27] uses its own app to understand the contextual usage (e.g., time of day, location) on smartphones. In addition to studying usage patterns of mobile applications, our focus is to understand *how it varies with different factors, such as network performance, device type, and application type, and the possible generalizations from our observations.* To meet this goal, our approach to design and implement Insight is quite distinct from prior published approaches.

The user's experience within a mobile application transcends multiple dimensions. In certain cases it depends on the interaction of the network properties with application features. It also depends on the application's design and how different features of the application appeal to each user. Unlike standalone measurement systems that capture various performance parameters only when users activate these processes, Insight focuses on application-level experience of users *when the users are interacting with specific applications.*

In order to capture this application experience of users, Insight is implemented as a toolkit library and developers can install this library into their application. This approach to analyze apps is quite similar to a few commercial analytics solutions [5, 7, 9, 10] that focus on providing business analytics to the users (e.g., number of downloads, geographical locations of users etc.). Insight goes further than these commercial toolkits to correlate network-layer and device-related factors that impact application usage experience of users. For example, Insight performs light-weight network latency measurements during the application sessions. Thus, by tracking in-app purchases made by users and the network quality statistics of sessions in which these purchases were made, developers can infer how user revenue may be impacted by quality across different networks.

We performed a long term deployment of Insight in two popular (commercially available) mobile applications: (i) a multi-player mobile role playing game, called Parallel Kingdom or PK [77] (on Android and iOS), with more than 1 million users across more than 120 countries which used Insight for more than 3 years and (ii) a third-party study application for students called StudyBlue or SB [93] which used Insight (Android only) for more than 1 year and has more than 160,000 Android users.

Following are examples of some useful insights obtained from our study. We observed that even on device models with similar CPU, memory and battery capacity, the battery drain caused by the same mobile application had a high variance (upto 3×). Also, high battery drain caused by the application led to lower session lengths across both PK and SB. This points out the need for device specific optimizations for the application to remain energy efficient across a range of devices. For example, controlling the screen brightness on a Kindle Fire device reduced the average battery drain by 40% while using SB. Also, higher network latencies due to poor network performance reduced user interactivity across both applications. The impact was higher for the PK (upto 40% reduction in interactivity), which requires real-time communication with the servers. Furthermore, poor network performance led to shorter user sessions and loss in application revenues for PK (upto 52% under bad network conditions).

1.5 CONTRIBUTIONS

Diagnosing causes of poor user experience while using wireless networks is a very challenging problem. Poor user experience can be caused by issues at the wireless network infrastructure, client devices or mobile applications. Our research projects are motivated by the need to build systems and tools for different vantage points that can aid in solving this problem. Furthermore, the availability and accessibility of each vantage point impacts the most applicable solution. To this end, we build systems that capture and manage important parameters across the network stack (RF, network and application) at WiFi Access Points, client devices and mobile applications. Through real-world deployment of these systems, we uncover the nature of problems experienced

by users and show how we can mitigate these problems. Specifically, the contributions of this dissertation are described as follows:

1. We designed and implemented the WiSe infrastructure to perform a long-term systematic study of WiFi experiences in home environments and present a detailed characterization. While the research community has a broad understanding of WiFi performance in different settings and we know that users are able to connect and get reasonably "good" performance most of the time, there are instances where these networks appear to be frustratingly slow or unavailable. We present our unique approach of performing *inline* wireless measurements on the WiSe WiFi Access Points, and do so across 30 homes for 9 months. We also develop a lightweight and passive wireless performance metric, called Witt, that can be used to quickly measure the wireless experience of a WiFi AP and its clients. In our measurements, we observed that WiFi experience was poor for 2.1% of the time. Across different locations, the major causes of poor performance (e.g., channel congestion vs. poor signal) was dependent on the environment and the nature of the wireless deployment. There were intermittent cases of WiFi to WiFi and non-WiFi (e.g., microwave ovens) to WiFi device interference with more than 50% degradation in throughput. We also observed that a majority of home Access Points used static configurations, indicating inefficient spectrum utilization and vulnerability to sources of wireless interference.

2. We designed and implemented COAP, a vendor-neutral cloud-based centralized framework to configure, coordinate and manage individual home Access Points using an open API implemented by these commodity WiFi Access Points. We motivate many COAP based applications to improve residential WiFi experience. Traditionally, majority of home WiFi APs use static configurations (e.g., WiFi channel) as observed through the WiSe deployment or update configurations based on local observations. Through real-world deployment of COAP Access Points, we show upto 47% reduction in airtime utilization or channel congestion due to co-operative channel assignments performed by COAP. We also show how

wireless interference issues due to channel congestion or hidden terminal style interference can be mitigated, resulting in more efficient airtime utilization due to 50% reduction in MAC losses and improved quality for HTTP video flows. Furthermore, prior wireless statistics collected by COAP controller can be used to train models to predict future wireless activity. For example, we were able to predict future microwave oven interference activity in an apartment building using 5 days of training data about microwave oven activity observed by WiFi APs.

3. We designed and implemented Insight, a toolkit library that explores the use of client device and mobile application vantage points to understand mobile application experience. Insight goes beyond existing mobile analytics solutions [5, 7, 9, 10] to understand the impact of network performance, resource consumption (CPU, memory and battery), devices and platform on user experience and application revenues. In this first of-its-kind of academic study, Insight served as a distributed lens from inside two popular mobile applications (Parallel Kingdom [77] and StudyBlue [93]) with greater than one million users over more than three years. Insight helped understand the end-user impact of performance bottlenecks experienced by client devices and mobile applications. We showed how higher network latencies due to poor network performance reduced user interactivity across both applications. User interactivity dropped upto 40% for Parallel Kingdom users due to poor network performance, since it requires real-time communication with the servers. Furthermore, poor network performance led to shorter user sessions and loss in application revenues (upto 52% under bad network conditions).

1.6 OUTLINE

The rest of the thesis is organized as follows. In the first of the thesis, we design and implement the WiSe infrastructure using Access Points as vantage points to measure the wireless performance of dense residential WiFi deployments (Chapter 2). In the second part of the thesis, we focus on developing mechanisms

to mitigate wireless problems in residential WiFi deployments (Chapter 3). We develop the COAP framework to co-ordinate and manage home WiFi Access Points in residential multi-dwelling units. COAP provides a light-weight centralized control plane using the Software Defined Networking (SDN) paradigm. In the last part of this thesis, we explore approaches using client devices and mobile applications as vantage points to identify problems that degrade mobile experience (Chapter 4). In Chapter 5, we compare our work with prior approaches and methods to understand and diagnose problems in wireless environments. We conclude and discuss the avenues for further research in Chapter 6.

2 OBSERVING RESIDENTIAL WIRELESS EXPERIENCE THROUGH WIFI APS

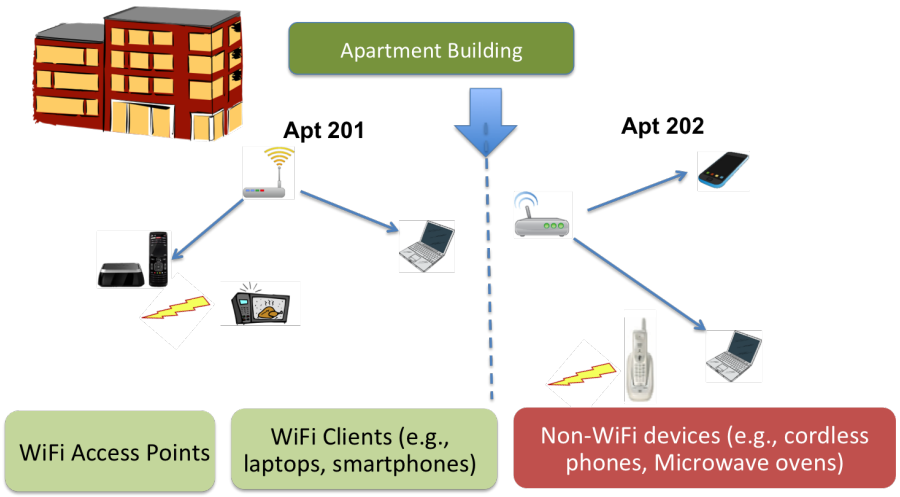

Figure 2.1: Multi-dwelling residential units today consist of a diverse set of coexisting WiFi Access Points (APs).These APs serve as an Internet gateway to WiFi capable client devices and co-exist with many non-WiFi interferer devices.

2.1 MOTIVATION

The capacity of residential broadband networks is increasing over time [31]. At the same time, residential wireless networks are becoming more complex and diverse (Figure 2.1). Residential WiFi Access Points (APs) today serve a diverse set of client devices – smartphones, tablets, laptops and other devices such as Wireless TVs. The emerging trend of Internet of Things (IoTs) is only going to further increase the density of wireless devices in residential environments.

In this chapter, we describe WiSe, a measurement infrastructure for observing residential WiFi performance and diagnosing problems (e.g., interference, congestion, weak signal etc.) using home Access Points as vantage points. We also develop techniques that can leverage passive statistics available at WiFi Access Points to diagnose wireless problems at homes. WiFi Access Points are

ideal vantage points for this purpose since all traffic passes through them and allows us to monitor the performance characteristics of all associated clients (e.g., laptops, smartphones, tablets and video streaming devices).

Passive measurements of wireless systems, usually through carefully placed sniffers (e.g., Jigsaw [34] and Wit [60]) have been popular mechanisms to understand various properties of these systems. Our proposed approach in collecting measurements and understanding WiFi performance is passive as well. Unlike such prior wireless measurement efforts, our work (inspired in part by the BISmark [94] project) places specialized measurement capabilities embedded into regular WiFi APs, and observes various wireless properties inline. This approach has various unique advantages as described next.

Inlined measurements from home WiFi APs. Measurement capability installed in WiFi APs provide a unique vantage point. This is a useful approach for collecting wireless measurements and understanding wireless properties for the following reasons. First, our infrastructure is able to observe all traffic to and from its clients without missing any data, unlike external passive sniffers. Second, APs can sometimes be re-configured by users to operate on different channels or with different transmit power levels. Since our measurement infrastructure sits inside these APs, we are both aware of these changes and can ensure that such changes do not affect our measurement process. Third, we can use different built-in functions in the APs to capture various types of information, e.g., CRC errors, CCA threshold settings, busy time of channel as reported by the AP's WiFi NICs, etc. The availability of this information directly from the AP itself proves to be particularly useful.

Perhaps, the most important benefit on inlined measurements is that we can provide better incentives to residents to deploy the measurement infrastructure. In our deployment, we provided WiFi Access Points for residents for free. The requirement to receive our AP was to deploy them *as their primary WiFi AP in their apartments and homes,* and a willingness to participate in our measurement framework. If for some reason, their AP has any problems, users are incentivized to fix them (and inform us if necessary) almost immediately. In our prior experience, users are not always as meticulous in ensuring that passive sniffers stay online and functional for long running measurements.

Figure 2.2: We built the WiSe Access Point platform using OpenWrt based APs (left). Each WiSe Access Point consists of two WiFi cards (right). The primary card acts as an Access Point to serve WiFi client devices in homes and also capture different channel and link related parameters. The secondary WiFi card is used for additional passive wireless measurements.

2.2 WISE INFRASTRUCTURE AND MEASUREMENT FRAMEWORK

We now describe the different components of the WiSe infrastructure, the wireless analytics capabilities and the deployment performed for this study.

WiSe Access Points: We developed OpenWrt [12] based APs (Figure 2.2) using the ALIX 2d2 platform [75] (having a 500 MHz AMD Geocode CPU, 256 DDR RAM and slots for flash disk, Mini PCI and USB accessories). This is very similar to the hardware configuration available used on commodity wireless APs. OpenWrt is an open source Linux distribution for embedded systems, which is also used by some commercial AP vendors. Each AP is equipped with two Atheros 9220 Mini-PCI WiFi NICs. The "primary" WiFi NIC is setup in the Access Point mode to allow users to connect their WiFi based devices to the AP. The "secondary" wireless NIC is used as back-up wireless NIC to perform various wireless measurements for the purpose of this study. The nodes were set to use a default transmit power of 17 dBm (50 mW). We also benchmarked our routers against commercial routers with similar configurations and achieved comparable performance under different workloads.

Type	Description
AP statistics	Record aggregate statistics local to the AP such as airtime utilization, overheard beacons from external APs, total received packets, packet counts with CRC errors.
Client statistics	Record aggregate downlink statistics per associated client (e.g., Total packets sent, received, retried, client's signal strength at AP).
Non-WiFi devices	Report non-WiFi devices detected by the AP (type, start time, duration, RSSI) using Airshark [84].
Per-packet summaries	Record packet summaries for all links overheard by the AP. Each packet summary contains: received timestamp, packet length, PHY rates, retry bit and RSSI (average overhead < 1%).
Flow statistics	Report aggregate flow level statistics (e.g., sent, recv, total bytes).

Table 2.1: The data gathered by the WiSe infrastructure about residential wireless activity by using each connected WiSe AP as a vantage point.

Measurement controller: A measurement controlling server collects periodic wireless measurements from the APs. The controller runs on a standard Linux server (3.00 GHz dual core CPU, 4 GB DRAM) with a public hostname. We deployed the controller in our lab's server cluster with public hostname/IP.

Measurements

Table 2.1 provides the list of measurement capabilities implemented at the WiSe APs. We started with a basic set of measurement capabilities and over time, incrementally added all of these capabilities into the WiSe APs to collect various statistics. We now briefly describe these different measurements performed over the course of this study.

Basic Statistics. Each WiSe AP periodically collects aggregate statistics (in 10 second intervals) such as overheard WiFi packets and beacons from

neighboring APs present in the vicinity. These aggregate statistics include the total number of CRC errors for received packets and the per link packet transmission summaries for the overhead links. Local packet statistics collected the WiSe APs include aggregate statistics per local WiFi link (using kernel statistics), the nature of wired traffic by finding the TLDs (Top Level Domains) corresponding to the flows.

Airtime utilization. To measure the airtime utilization, we use the aggregate "busy time" statistics maintained by the Atheros NICs that we used with our APs. Whenever the energy level is detected to be higher than the CCA (Clear Channel Assessment) threshold (due to WiFi or non-WiFi sources), the channel is marked as busy, otherwise its marked as free. The driver records cumulative statistics about the total channel busy period observed over a period of time. We use this metric as an indicator of channel utilization because it accounts for all the factors the factors that cause the energy to be higher than CCA: packets received correctly as well as received in error (due to weak signal or collisions), energy from non-WiFi sources etc. In addition, the driver also records airtime utilization statistics due to local packet transmissions as well as utilization due to overheard WiFi packets only.

Per-packet Summaries. To perform WiFi interference analysis (§2.5), each WiSe AP records a short 10 byte per-packet summary for the WiFi data packet headers of its own links as well as the overheard data packets from other WiFi links on the same channel. Each packet's summary (grouped by link) contains the MAC timestamp (from driver) for the packet's transmission start time (32-bit timestamp with microsecond granularity), packet length, PHY rate, retry bit and RSSI. These statistics are periodically reported back to the controller to provide fine grained information about the WiFi activity on the channel. Across our deployment, the average WiFi packet size was between 900 bytes to 1000 bytes. So, it translates to only 1% average overhead per data packet.

Non-WiFi device detection: In residential settings such as apartments, an AP can be in the vicinity of multiple non-WiFi devices (e.g., Microwaves, Cordless Phones, Baby Monitors etc.) located in same as well as neighboring

Location (AP IDs)	Count	Deployed Since
Bldg 1 (APs 1 - 14)	14	Sep 2012
Bldg 2 (APs 25 - 30)	6	Dec 2012
Others (APs 15 - 24)	10	Oct 2012

Table 2.2: Summary of the deployment of WiSe Access Points

apartments. Some of these devices can cause trouble to nearby operating links through packet losses and increased channel utilization [45, 84]. In our prior work, we designed a tool called Airshark [84] which demonstrates the feasibility of using commodity WiFi NICs available in the market (e.g., Atheros 9280 chipsets) to detect non-WiFi devices. Airshark uses subcarrier-level energy samples received from the WiFi NIC as input to detect the presence of different types of non-WiFi devices. For this project, we ported Airshark to run on our WiSe Access Points to detect the presence of non-WiFi devices in residential environments.

Since Airshark requires the WiFi NIC to be placed in monitor mode, we implement this functionality over the secondary WiFi NIC of the AP and have it scan all WiFi channels periodically. We also use the information to study the impact of non-WiFi interference in our deployments.

Deployment

To understand the wireless characteristics in real home environments, we distributed our Access Points to volunteers who used them as the primary wireless APs inside their homes (Table 2.2) to connect all their WiFi devices. For this study, we deployed a total of 30 Access Points in Madison, Wisconsin (figure 2.3) and actively ran this deployment for more than 9 months. During this period, all APs were configured to use a randomly selected channel on different days to capture the impact of different channels on our APs. Channel changes were done sometime late at night when the APs had no associated clients. We now discuss the details of our deployment.

Figure 2.3: Deployment of WiSe APs in the downtown area and some suburban locations in Madison, Wisconsin. The stars indicate the two apartment buildings and the other points indicate locations with deployment of single APs.

Multiple APs within two residential apartments. We collaborated with the property managers of two apartment buildings (Bldg 1 and Bldg 2) in the downtown area of our city to sign up volunteers, who wanted to be a part of our study in exchange for getting free wireless APs. Both these apartments were multi-storied with many 1-2 bedroom apartment units on each floor. In the case of Bldg 1, we focused our deployment on a specific portion of the building (5 floors) having 12 units per floor. We gave away 14 APs to Bldg 1 residents such that each of these floors have multiple APs (usually between 2 and 4). Bldg 2 is a 10-story building with dormitory style housing (1-2 bedrooms per apartment unit) in which the property manager provides its own wireless service to residents by deploying its own off-the-shelf APs. Hence, working with the property manager and the local ISP, in this building we deployed 6 APs on 2 consecutive floors (3 per floor) in the designated areas, that replaced their existing APs. We were told that these building APs were still the primary form of Internet access in this residential building. Thus, the two buildings had slightly different approaches to wireless AP deployment and provided us with some diversity in our measurements.

Across different types of user homes. We also distributed our APs to volunteers and colleagues staying at different low to high density residential

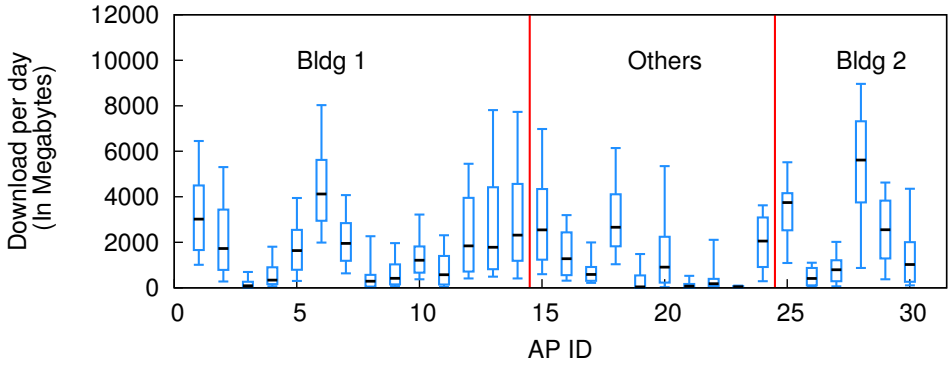

Figure 2.4: Distribution of the daily data download over the WiFi network per WiSe AP (Sep 1, 2012 - Jan 31, 2013). 10th, 25th, 50th, 75th and 90th percentiles are shown in this figure.

units spread across different locations in our city (Others). This was done to capture the wireless network properties across different types of home environments.

Volume of observed traffic. Over the course of our deployment, we observed wide-ranging daily WiFi usage characteristics across our users (Figure 2.4). The median WiFi usage across users varied between a low 30 MB per day to over 5.6 GB per day. The 90th percentile usage for some users was as high as 8-9 GB per day. The usage of these wireless networks can be driven by a number of factors such as the kind of wireless devices used, wired access link capacity, the nature of the traffic/service used and user behavior.

User privacy in our measurement study: An infrastructure can potentially be highly intrusive to users. However, in this work, we have only needed to observe IEEE 802.11 frame header information for the most part, that do not carry any private information of users. As described to our Institutional Review Board, we informed our participating users (and in some cases the relevant ISPs) that we do not capture user identifiable information in the study. In a few aspects when information such as flow types are analyzed (e.g., Netflix video), we have taken care to anonymize user and home identities (using an identity

decoupling technique) in the measurements that this information cannot be mapped back to any individual user or home.

2.3 QUANTIFYING WIRELESS EXPERIENCE THROUGH A METRIC

In order to analyze and understand the large volume of wireless performance data that we have observed, we wanted to categorize them through the use of a metric. In defining this metric, we set out some modest goals as outlined in §1.2 with our intent being two-fold: (a) to quickly identify periods of good and bad performances for the WiFi links, and (b) to be able to explain to residents or ISP personnel, how often there are performance problems of some type.

There can be many different ways to approximate the "goodness" of a wireless network. In this work, we wanted to pick something that is passively observable in real-time, captures the wireless experience alone, and is application-agnostic. We call our defined metric, Witt, and explain its construction next.

WiFi-based TCP Throughput metric

The idea of our proposed metric is fairly simple — it measures the *likely* TCP throughput between a client device and its AP (or a server located on the same uncongested LAN as the AP) , given the current wireless conditions that exist. The metric is clearly, a property of the client and AP combined and the protocol used (802.11g vs. 802.11n). In this work, we also consider the average value of the metric for the entire AP as a single aggregate. In such cases, we take the average of Witt for all its *active* clients. An "active client" is defined as one which has sent at least a minimum level of traffic in recent time window. We use a threshold of at least 500 packets in the last 10 second window — we want to bias the metric towards client devices that are imposing a higher load at the current time, than the ones that are less active.

Given that TCP-based flows are a dominant fraction of Internet traffic, this metric likely captures most of the user's experience when the wireless link is the bottleneck (short of analyzing performance on an application-by-application

Causes	Indicator
Non-WiFi devices, Rate Anomaly, Heavy external traffic	Airtime utilization
WiFi and non-WiFi Interference, AP and Client Transmit Power + Signal Environmental/Location effects	Losses + Data Rates
Local WiFi clients sharing an AP	Local contention

Table 2.3: The list of potential factors can degrade a WiFi link's performance and potential indicators of these issues.

basis). We believe that such a metric would actually capture a lot of wireless properties on the network. Factors that imply poor wireless conditions, e.g., low signal strength, high degree of interference from various sources leading to losses, increased latency on the WiFi path due to reduced PHY rates or multiple re-transmission attempts, high airtime utilization leading to a reduced ability to send traffic, all will reduce the likely TCP throughput estimate, and vice versa. Thus, we focus on estimated TCP throughput as a direct measure of the link's performance.

Causes and indicators related to wireless performance. Table 2.3 presents a short summary of such causes for poor performance experience by wireless links. The table also shows that the impact of multiple such causes can be captured by using a set of key indicators observed locally by the APs.

For example, the impact of non-WiFi activity as well as presence of WiFi traffic using low PHY rates (rate anomaly) can be observed by APs through an increase in *airtime utilization*, i.e., the fraction of airtime occupied by only external WiFi and non-WiFi transmissions . Similarly, an increase in packet losses for a link indicates the presence of factors such as WiFi or non-WiFi interference at the receiver, poor signal quality at the client, location dependent performance problems etc. High local contention at an AP caused by the presence of multiple clients with high traffic demand can reduce the available throughput capacity per client. In the next section, we use these indicators to build the Witt metric.

Parameter	Airtime	CRC errors	MAC retries	Signal (at AP)
Minimum	17.6%	1.2%	6.2%	-71 dBm
10th percentile	20.5%	6.4%	13.3%	-70 dBm
50th percentile	38.3%	17.2%	24.9%	-61 dBm
90th percentile	47.5%	41.6%	56.7%	-52 dBm
Maximum	72.5%	98.8%	86.4%	-49 dBm

Table 2.4: Parameter value ranges in our ground truth measurements.

How to measure Witt?

A key objective of our metric is that it should be easy to obtain through passive observations at the AP and should not impose additional traffic. Hence, we built a simple model of Witt based on likely factors that will impact the metric. To do this, we first collected targeted ground truth data, built and tested our model of separate parts of the data, and then used it for analyzing wireless network performance. We first describe our ground truth measurements and then our model, leading to the metric.

Ground truth measurements. To collect ground-truth measurements of WiFi-based TCP throughput under different conditions, we placed 4 of our own clients (laptops) at 8 different deployment locations within the apartment buildings (excluding lab experiments). These laptops co-existed with the actual users of the WiSe APs and performed TCP downlink throughput runs (using iperf between APs and clients) that lasted 20 seconds each. This setup allowed to us create a scenario with the WiFi link being the bottleneck.

In the case of the actual home deployments, the clients ran throughput measurements in intervals of 5 to 10 minutes over the course of a week through which we collected hundreds of these measurements. Further, these clients connected to different APs within the apartment buildings to emulate different types of link conditions. These experiments were automatically conducted at different times of the day and hence, covered a diverse set of link characteristics, channel and operating conditions. Table 2.4 shows the distribution of different

Feature	Correlation Coefficient
Airtime	0.321
CRC errors	0.345
Local contention	0.463
Signal strength	0.536
Effective rate	**0.882**
Effective rate + Airtime	0.915
Preferred "Link exp" model (Eqn. 2.2)	**0.958**

Table 2.5: Correlation of metrics with the observed TCP throughput (802.11g). The best individual and overall metrics are highlighted.

parameter values from our ground truth measurements.

Creating the Witt metric. To understand the impact of different factors on the observed throughput, we correlated the TCP throughputs from our clients with various wireless statistics recorded by the WiSe APs in 10 seconds intervals (Table 2.1). The following are some of the example statistical features that we tested to predict Witt: (i) airtime utilization, (ii) CRC error rate, (iii) client signal strength, (iv) local contention, and (v) effective rate — we define the last two precisely next.

- *Local contention (*c*):* the relative amount of other client traffic transiting through this AP as a fraction of total traffic passing through this AP. It captures the fraction of time the AP spends on transmitting (and receiving) traffic of its other clients, and hence, reduces the ability of this client to receive (and send) traffic.

- *Effective rate (*r*):* captures the net effect of packet losses and choice of PHY rate used on an AP-client link. It uses aggregate kernel statistics provided by the wireless driver about the number of successful (s_i) and total packet (p_i) transmissions at each PHY rate ($r_1, ..., r_n$) used by an AP-client pair. It is defined as:

$$r = \frac{1}{\sum_i p_i} \sum_i s_i \cdot r_i, \ 1 \leqslant i \leqslant n \tag{2.1}$$

	Coefficients (β_0, β_1)	**95% Confidence interval for** β_1
802.11g	(0.167, 0.422)	(0.403, 0.441)
802.11n	(-0.493, 0.733)	(0.720, 0.746)

Table 2.6: Parameters of the linear model for predicting Witt.

Table 2.5 shows the correlation between these features available at the AP (individually and also in various combinations) and the observed TCP throughput from our ground truth measurements. A high correlation value indicates greater capability of the metric to predict the saturation throughput achievable by the link. Among all possible combinations tested (not all are shown in Table 2.5), the best combination of features that has the highest correlation is given by Equation 2.2. We call it the "link experience" model — which depends on a combination of airtime utilization from external sources (a), local contention (c), and effective rate (r). Together this model intuitively captures various aspects that govern the experience likely to be expected for a wireless link. Note that all these features are based on aggregate statistic per link (10 second intervals in our current system) and can be easily obtained using kernel statistics reported by wireless drivers.

$$\text{link\_exp} = (1 - a) * (1 - c) * r,\ 0 \leqslant a \leqslant 1,\ 0 \leqslant c \leqslant 1 \tag{2.2}$$

Finally, to map "link experience" to Witt, we create a simple linear model between the two (Equation 2.3). We tested the fidelity of Witt by dividing our ground truth data into training and testing data sets, each corresponding to different days in our entire dataset. The parameters for our linear model as well as the error estimates for the slope, β_1 obtained using the training data, is indicated in Table 2.6. The small range of the the 95% confidence interval shows that the linear model is a reasonable estimate for predicting the throughput.

$$Witt = \beta_1 * \text{link\_exp} + \beta_0 \tag{2.3}$$

Witt	>=16 Mbps	8 - 16 Mbps	4 - 8 Mbps	1 - 4 Mbps	<1 Mbps
Rating	(V. Good)	(Good)	(Moderate)	(Poor)	(V. Poor)

Table 2.7: Using Witt to rate link quality.

A subjective rating. Based on the Witt value, we also use a subjective rating to classify links and APs into a range from *Very Good* to *Very Poor* based on the range of throughput observed (Table 2.7). Given that the highest Witt values estimated in our infrastructure approached 19 Mbps for 802.11g networks and 34 Mbps in 802.11n networks under best conditions, anything very low (1-4 Mbps) or lower was deemed poor or very poor. Further, our rating moved from across categories mostly with doubling of Witt value to indicate the ability of the APs to support higher TCP throughput.

Benchmarking the Witt metric

To evaluate the performance of the Witt metric, we present a couple of benchmarks in this section to compare the performance of the "link experience" model with other alternatives.

Effective Rate vs. Link Experience. As discussed earlier, effective rate exhibited the highest correlation amongst the individual features in estimating the TCP throughput. We used our ground truth TCP throughput measurements and compared them with the predicted TCP throughput values using the linear regression model (Equation 2.3) for effective rate and the link experience model. Figure 2.5 compares the actual and predicted TCP throughputs for a few representative WiFi links from our ground truth experiments. It shows that the link experience model resulted in a better fit compared to effective rate. Only using effective rate as a feature can result in an overestimation of the link's TCP throughput (e.g., in the presence of high channel utilization) which leads to higher prediction errors. The link experience model also accounts for the potential reduction in TCP throughput due to factors such as high airtime and/or local contention which results in a better prediction.

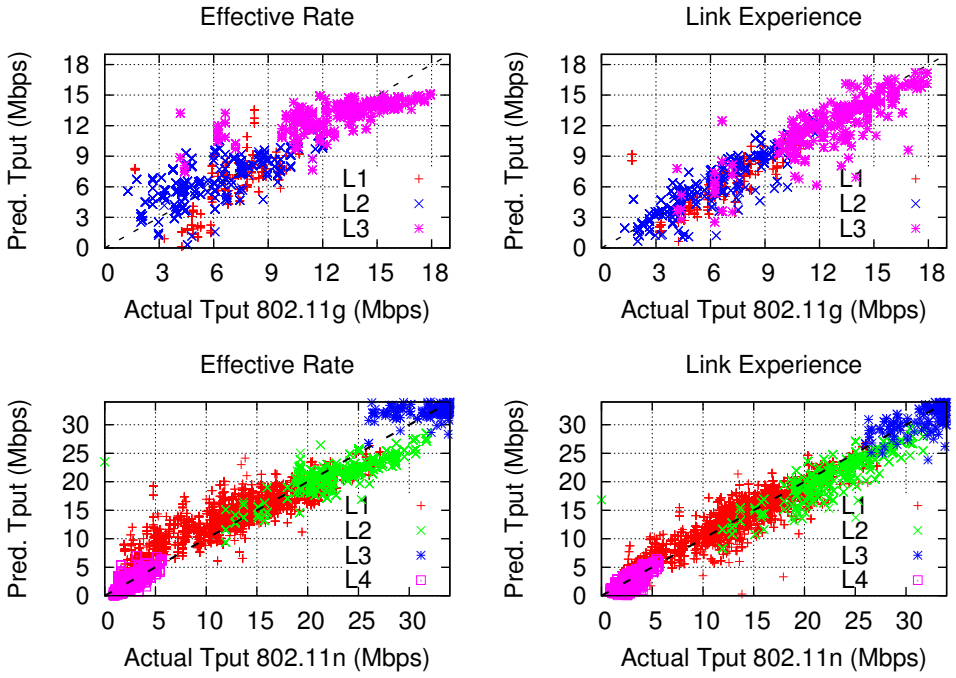

Figure 2.5: Scatter plot showing the actual vs. predicted TCP throughput values for the 802.11g (top) and 802.11n (bottom) ground truth experiments by using "Effective Rate" and "Link Experience". Points near the "x=y" line indicate instances with accurate prediction of TCP throughput. Each set of points corresponds to a different WiFi link.

CDF of prediction errors using different features. Figure 2.6 shows the CDF of the errors of our proposed link experience model to create the Witt metric in estimating the WiFi TCP throughput and compares that to the other alternatives. The figure shows that our proposed formulation performs effectively and better than the baseline metrics with an estimation error under 1.5 Mbps and 2 Mbps for more than 80% of 802.11g and 802.11n instances respectively. In summary, Witt provides a quick estimate of likely TCP throughput that an average WiFi link can achieve based on observed traffic characteristics at the AP and without requiring any active measurements.

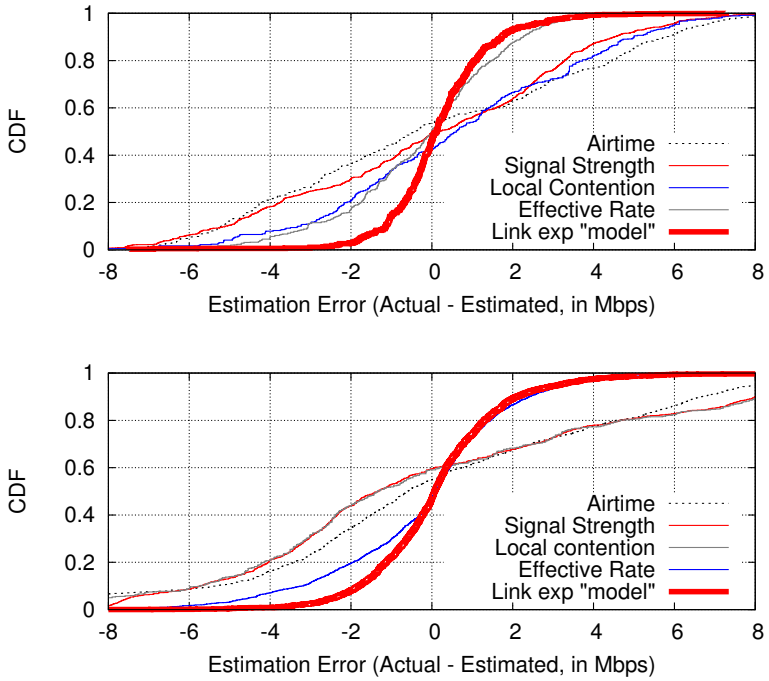

Figure 2.6: CDF of errors between the actual and predicted TCP throughput values obtained by using different metrics for 802.11g (top) and 802.11n (bottom).

2.4 USING WITT TO CLASSIFY WIRELESS EXPERIENCE IN THE WILD

The Witt metric allows us to classify links into different categories based on their performance from the large volume of data collected through our measurements (over 100 GB). In this section, we analyze on the impact of various factors (local and external) that affected the performance of WiSe APs. To study periods with WiFi activity at the APs, we focus on the periods when a WiSe AP has at least one *active* client (§2.3).

How did the link performance vary across APs over time?

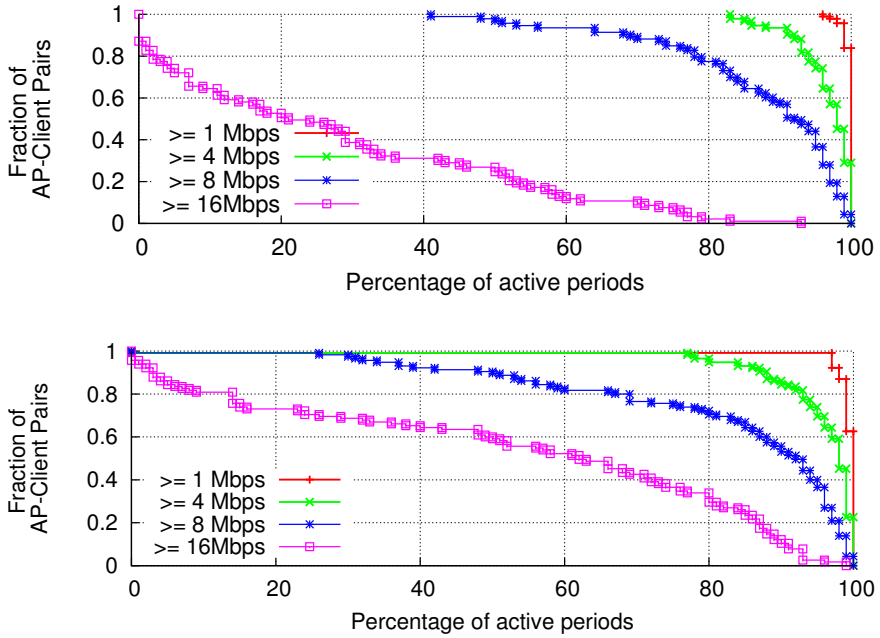

Figure 2.7: Distribution of Witt across AP-client pairs using 802.11g (top) and 802.11n (bottom) in our deployment that were active for at least 20 days.

During the course of our study, a diverse set of clients associated with the WiSe APs (e.g., laptops, handhelds, entertainment devices etc.). We measured their Witt values (§2.3) during active periods grouped their results based on their Witt values. Figure 2.7 shows the distribution of the performance experienced by AP-client pairs who were active for at least 20 days in our deployment. While a majority of our deployment consisted of 802.11g based WiSe APs, others used 802.11n based APs. For the different Witt threshold values used to characterize link performance in Table 2.7. For example, a line corresponding to >=16 Mbps shows the fraction of these links (an AP-client pair) whose Witt value was above 16 Mbps for a given percentage of their active periods.

The graph shows that a majority of links performed well for most of the time. For example, around 80% AP-client pairs experienced Witt >= 8 Mbps during more than 80% of their active periods. But, there were occasional periods for

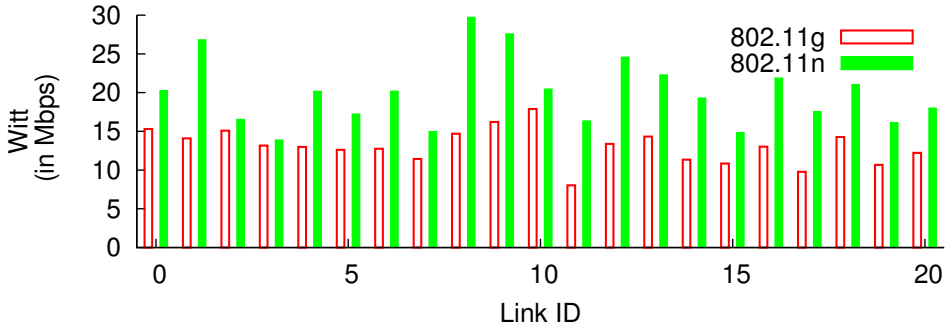

Figure 2.8: Comparison of average Witt for AP - client pairs over 802.11g and 802.11n (3 days each).

many AP-client pairs where the link performance was "Poor" (< 4 Mbps). About 8 % of these pairs experienced Poor performance (<4 Mbps) during more than 10% of their active periods, indicating some periods of poor performance which may be observed by users. In the later sections, we discuss the causes of poor performance across our different deployments.

802.11g vs. 802.11n. To study the performance gains of using 802.11n vs. 802.11g for a given link, we configured the 802.11n based APs to use 802.11g for a few days. Figure 2.8 compares the average Witt values on 802.11g vs. 802.11n for clients that support both protocols. The Witt values for most AP-client pairs increased between 12% to 100% due to usage of higher PHY rates (e.g., 65 Mbps) and use of 802.11n specific MAC layer optimizations such as frame aggregation. Interestingly, some clients experienced similar performance on both protocols (e.g., AP-Client pairs 2 and 3), indicating that using 802.11n did not ensure higher throughput for all clients. This can happen due to packet losses based on the link quality or the prevalence of interference from co-existing 802.11b/g WiFi transmitters on the same channel [90].

Causes for "Poor" wireless experience. During the course of 80 days from November 12, 2012 to January 31, 2013, we detected a *total of 186 and 2031 minutes* of the "Very Poor" and "Poor" instances respectively, across all 30 WiSe APs

Indicators				Bldg 1		Bldg 2	
A↑	**S↓**	**L↑**	**R↓**	**V. Poor**	**Poor**	**V. Poor**	**Poor**
✓	×	×	×	0%	18.4%	0%	1%
×	×	✓	×	24.2%	**49.5%**	25.2%	**78.1%**
✓	×	✓	×	**61.8%**	26.7%	2.1%	1.4%
×	✓	✓	×	2.3%	1.1%	20%	15.8%
×	✓	✓	✓	9.4%	0%	**51.6%**	3.4%
Others				2.3%	4.3%	1.1%	1.3%

Table 2.8: Distribution of causes responsible for "Poor" and "Very Poor" periods in Bldg 1 and Bldg 2. Each "cause" is composed of 1 or more of the following indicators and corresponding threshold values: High Airtime ($A\uparrow, > 60\%$), High MAC Loss Rates ($L\uparrow, > 50\%$), Low signal strengths ($S\downarrow, < -70\mathrm{dBm}$) and Low PHY rates ($R\downarrow, <= 12\mathrm{Mbps}$). Others correspond to all remaining combinations of factors such as high contention, signal strengths etc.

(average of 2.3 and 25 minutes, respectively per day across all APs). Thus, the "Very Poor" periods are rare but the "Poor" periods can occur intermittently depending on the link and the location. Overall, these cases accounted *for 2.1%* of the active periods during the 80 days.

In §2.2, we had discussed that our deployments consisted of two apartment buildings with multiple WiSe APs deployed within the same building. We aggregated the instances of poor wireless performance across WiSe APs in each of these buildings (Bldg 1 and Bldg 2). Table 2.8 breaks down the periods of poor wireless performance in our two apartment deployments to understand the causes of poor wireless performance at these APs. Each row in Table 2.8 is a combination of indicators of poor performance: high airtime (>60%) [53], high wireless MAC layer packet losses (>50%) indicating the fraction of wireless transmissions that were retried, low signal strengths (<-70dBm) and low PHY rates (<= 12Mbps). We use the signal strengths from clients observed at the APs as estimate of link quality at clients since we do not have access to the clients.

The table shows that the presence of both high airtime and wireless losses (A $\uparrow$ +L $\uparrow$) were the main cause (61.8%) of "Very Poor" performance in Bldg 1. This can happen due to the network congestion at the WiSe APs and clients leading to high packet losses. In Bldg 2, the major cause of "Very Poor" performance (51.6%) was poor signal strengths (< -70dBm) which lead to high packet losses as well as usage of low PHY rates by the rate adaptation algorithm (S $\downarrow$ +L $\uparrow$ +R $\downarrow$). The impact of other factors ("Others"), such as period of high local contention (>0.5) from other clients associated to an AP was quite low at both locations (<= 4.3%). The prevalence of low local contention at the wireless hop is due to the fact that it was not common for multiple clients associated to the same AP to generate high traffic demand during the same time interval (10 second intervals). In some cases where there were multiple *active clients* at the same AP, bottlenecks on the wired link or low traffic demand led to lower contention on wireless hop.

The most likely cause for the above observations about poor wireless performance is the *nature of the wireless deployments in the two buildings*. Bldg 1 has private APs per apartment unit resulting in a dense wireless deployment. Thus, some APs experience occasional high airtime utilization (> 60%) due to neighboring sources of traffic. Bldg 2 provides centralized wireless service to its residents and thus, some users can occasionally experience poor signal quality based on the client device and location. The impact of losses due to low signal strengths was much lower in Bldg 1 (<12%) due to good signal coverage within each apartment. But, performance issues can still arise due to the dense nature of these deployments. Finally, high WiFi loss rates (L $\uparrow$) was the major cause for "Poor" performance in both Bldg 1 and Bldg 2 (49.5% and 78.1% respectively). Some potential causes of these losses are packet reception issues at some clients or packet collisions at the receivers due to external WiFi/non-WiFi sources. Through our interference analysis from WiFi sources (§2.5) in Bldg 1, we were able to attribute at least 19% of these lossy cases during "Poor" periods to strong interference from external WiFi transmitters.

Variability in wireless experience over time. Figure 2.9 shows the Witt for 2 AP-client pairs (Bldg 1) across a period of 80 days since 12th November 2012.

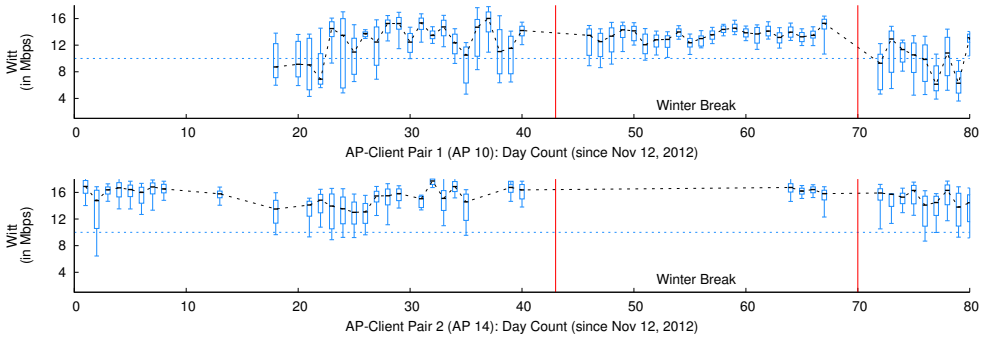

Figure 2.9: Distribution of Witt for 2 different AP-client pairs in our deployment from Nov 12, 2012 to Jan 31, 2013 and shows their variation in performance over the span of this period. 10th, 25th, 50th, 75th and 90th percentiles are shown in this figure.

These were amongst the most actively used clients in our deployment and were chosen to show the diversity in link performance over time across different locations and clients. Amongst the 2 clients, the one at AP 14 experienced consistently good performance over this period due to low neighboring wireless activity (median Witt around 14 - 16 Mbps). On the other hand, the client at AP 10 experienced a higher fluctuation in performance as shown by the wide range of the observed median Witt across days. One of the reasons for this behavior was high airtime utilization caused by some neighboring WiFi transmitters at AP 10 (§2.5). The "Winter Break" corresponds to the period between Dec. 25, 2012 and Jan. 21, 2013 during which the wireless activity in Bldg 1 was lower compared to the other days[1]. During this period, the median Witt value for AP 10 was consistently high and stable (around 14 Mbps) compared to other periods of time due to less utilized wireless channel conditions. This observation indicates the high impact of neighboring wireless traffic at this AP.

[1] A large fraction of residents in the building are university students, and many of them were out of town during the break.

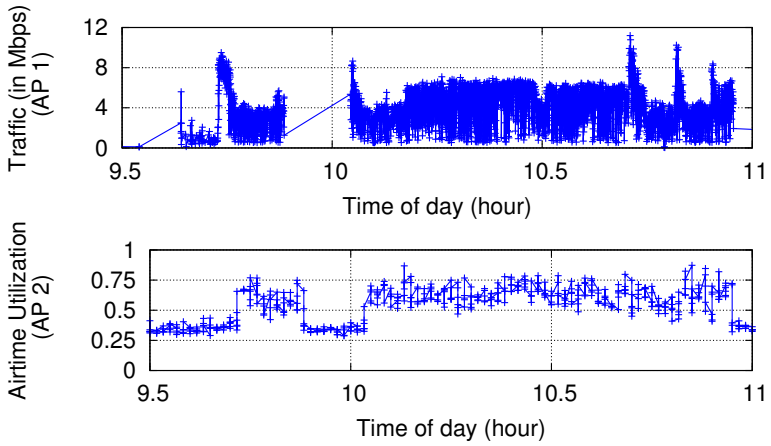

Figure 2.10: Time-series airtime utilization at AP 2 with and without the presence of traffic from an external AP 1 that used low PHY rates. AP 2 was inactive during this period.

2.5 WIFI LINK INTERFERENCE

WiFi links within homes can cause interference to each other in two major ways: contention from transmitters using low PHY rates and hidden terminals. We now discuss the prevalence of these issues in our WiFi deployments.

Contention from low data rate senders

The presence of transmitters using low PHY rates during the active periods of APs can cause their Witt to suffer. This is due to the rate-anomaly [96] problem caused by the loss in channel availability. Figure 2.10 shows an example of airtime utilization at a WiSe AP (AP 2) over a period of 2 hours in the presence and absence of traffic from an external AP (AP 1). During AP 1's activity, AP 2's airtime utilization (10 second average) increased from an average of 30% to around 65 - 70% due to the usage of low PHY rates by AP 1. This was one of the major cause of "Poor" performance in Bldg 1(§2.4).

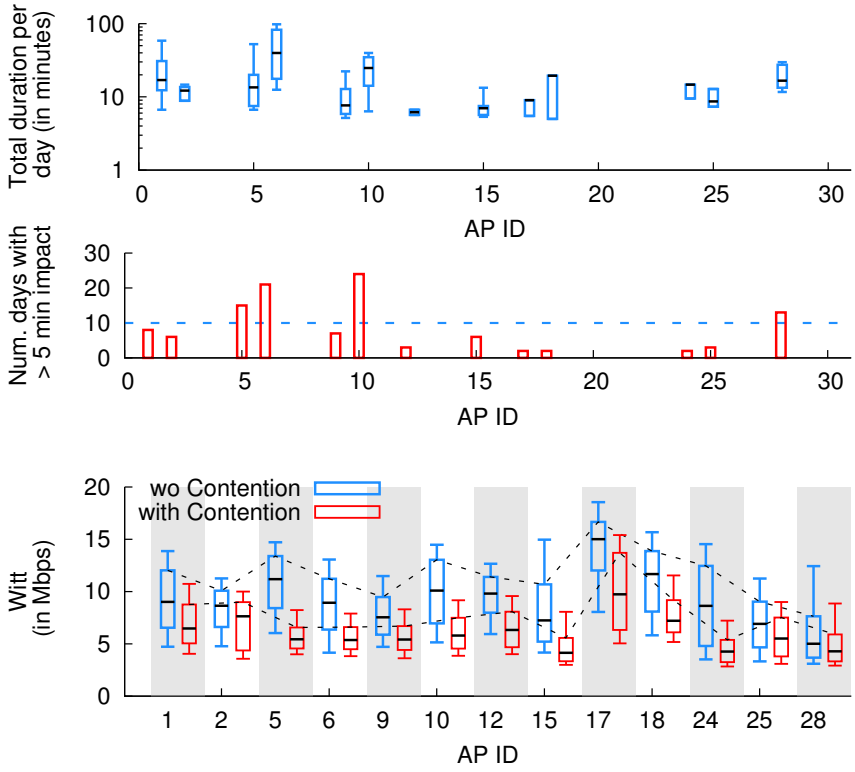

Figure 2.11: (top) Candlestick graph showing the distribution of active periods per day (Y-axis in logscale) during which the WiSe APs experienced contention from external transmitters (using average PHY rate <= 15 Mbps), (center) Bar graph showing the total number of days during which WiSe APs experienced this contention for a minimum duration of 5 minutes (from 12th Nov. to 21st Dec, 40 days), (bottom) Witt during active periods for clients with and without contention from such transmitters. This data is from 12th November to 21st December 2012 (40 days). 10th, 25th, 50th, 75th and 90th percentiles are shown here.

Prevalence and impact of contention from low rate senders. We analyzed the presence of such transmitters and their impact on the WiSe APs from 12th November to 21st December 2012 (40 days).

Figure 2.11 (top and center) shows the duration of *active periods* per day (in minutes) and the number of days respectively, during the 40 day period, over which the WiSe APs experienced contention from external WiFi senders that transmitted at least 500 packets while using low average PHY rates (<= 15 Mbps). Thus, it only shows the impact on WiSe APs while they were sending actual traffic to their clients. Figure 2.11 (top) shows that some APs (e.g, APs 1, 6, 10, 28) experienced contention from such senders lasting over multiple minutes (median between 20 - 48 minutes). Figure 2.11 (center) shows that 4 WiSe APs faced at least 5 minutes of contention from such low PHY rate traffic during 10 or more days while they were transmitting data to their clients. This happened due to the presence of nearby external APs at these locations (both Bldg 1 and Bldg 2) that consistently used low PHY rates for some clients across multiple days. Figure 2.11 (bottom) compares the distribution of Witt at the impacted APs during active periods with and without (5 minutes before and after) such contention. Some APs experienced consistent reduction in Witt during the periods. For example the 75th percentile value for AP 10 reduced from 12.5 Mbps to 6 Mbps during such periods. These external APs was the main cause for the low values of Witt at AP 10 as observed in §2.4.

In addition to activity from external low PHY rate transmitters, the period of channel contention experienced (Figure 2.11, top) is also partially dependent on the activity at the AP itself. For example, AP 6 (Figure 2.4) was the most active AP in our deployment which resulted in the higher periods of contention (median of 40 minutes) experienced by it. While, it is not possible for us to determine the signal strength properties of these external links that used low PHY rates, we analyzed our traces and found that many of these transmitters sent large data packets while using conservative rate adaptation. These links switched to low bitrates after a single failure resulting in high usage of low data rates and thus, airtime usage.

Packet losses due to WiFi sources

High packet losses were a major cause for the "Poor" cases in our deployment. Amongst external factors, hidden terminal (HT) style interference at a wireless

receiver from nearby links can reduce a link's Witt by increasing packet losses. We leveraged our deployment of 14 WiSe APs in Bldg 1, to collect timestamped (microsecond level) WiFi packet summaries (§2.2) for all observed WiFi traffic from multiple vantage points. These packet summaries from neighboring APs are time-synchronized and merged at the controller using prior techniques from [34, 89] using common data packet summaries present in both traces.

We use these synchronized and merged packets summaries from multiple APs in Bldg 1 to compute "hidden terminal events" in "epochs" of 15 seconds. We mark an epoch as a hidden terminal event for a WiSe AP when the loss rates for one of its links is 40% higher for packets *overlapped in time* by the interferer compared to packets *not overlapped by any other transmitter* [89]. When an epoch is marked as a hidden terminal event, the main cause is packet losses at the receiver caused by the overlapping packet transmissions from the interferer. To minimize false positives, we used a constraint requiring a minimum of 1000 WiFi packets for a link and a minimum of 100 packet overlaps from a potential interferer per epoch to check the presence of a conflict. Thus, our results are a conservative estimate of the interference experienced by the APs in Bldg 1.

What was the impact of HT interference on APs in Bldg 1?

We ran the interference detection measurements across APs in Bldg 1 for a period of two weeks (Figure 2.12). Across the 14 WiSe APs in Bldg 1, our analysis detected the occurrence of HT interference for 7 APs. Figure 2.12 (center) shows the number of days during which we detected at least 5 minutes of HT interference across different APs. During the two weeks, APs 6, 10 and 11 experienced HT interference between 5 - 8 days, indicating that some APs were repeatedly impacted by nearby WiFi interferers across multiple days. The median duration of interference per day (Figure 2.12, top) varied between 3 to 7 minutes but some APs (2, 6 and 11) experienced higher periods of interference (maximum of 23 - 87 minutes) during this two week period. Figure 2.12 compares the Witt of the impacted APs during periods with and without HT interference (within 5 minutes of HT event). The 75th percentile values dropped between 0.5 Mbps (AP 2) to 4.3 Mbps (AP 8) indicating high variation in impact of interference across some APs. AP 2 experienced minor

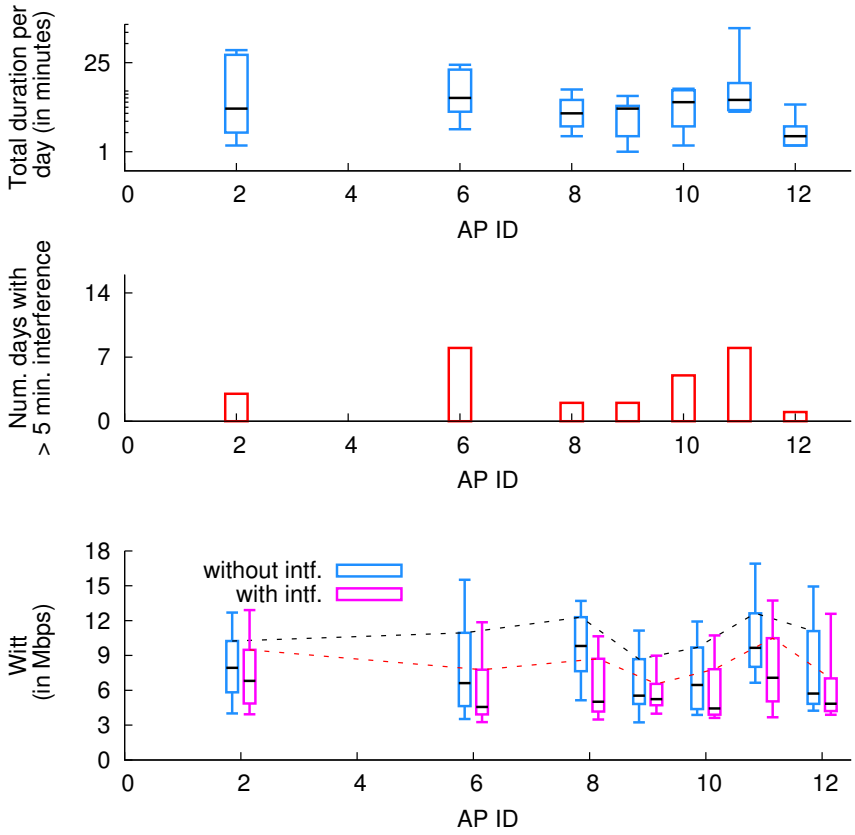

Figure 2.12: (top) Distribution of time per day during which the WiSe APs in Bldg 1 experienced hidden terminal interference from external links. (center) Bar graph showing the total number of days during which WiSe APs experienced hidden terminal interference for a minimum duration of 5 minutes per day over a period of 2 weeks. (bottom) Distribution of Witt with and without the presence of HT interference (within 5 minutes of the interference event). Min, max, 25th, 50th and 75th percentiles are shown.

reduction in throughput due to interference because, even though the loss rates for packets overlapped by the interferer increased by 40%, the proportion of these packets were low (< 20%) compared to the total transmitted packets.

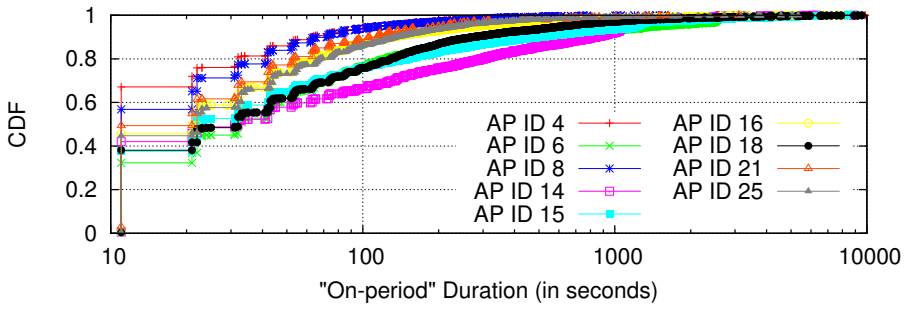

Figure 2.13: Distribution of "on-periods" across different WiSe APs. We represent a consecutive period of 10 second intervals with more than 100 data packets each as the *on-period* of the APs.

Our analysis shows the HT interference can occur intermittently, mostly for short periods of time. The occurrence of such HT interference is a property of both the receivers' and interferers' traffic. We observed high burstiness of WiFi links in home environments. For example, in our deployment only about *10% of total periods of continuous activity at the* WiSe *APs lasted more than 3 minutes* at most APs (Figure 2.13). This is one of the reasons for the small periods of interference in homes. The length of these bursty periods are dictated by the nature of the underlying traffic and can be highly variable depending on the type of traffic. For example, we observed highest periods of continuous activity during video streaming sessions (e.g., Netflix) which can download data in the order of 50 MB while buffering [82]. We found that, at APs 6 and 11, the periods of highest HT interference (39 and 87 minutes respectively) coincided with the usage of Netflix. The APs are more sensitive to such issues during these continuous periods of high activity, while short periods with bursty traffic are less likely to experience these performance issues from external sources due to the lower volume of the transmitted traffic.

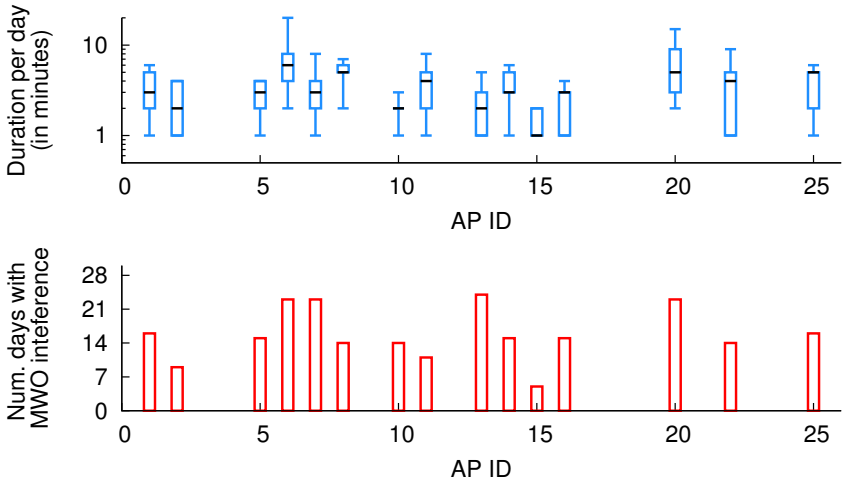

Figure 2.14: (top) Distribution of duration per day during which microwave interference reduced Witt by 20%. (bottom) Number of days from a 30 day period, during which the APs experienced at least 1 minute of such interference.

2.6 NON WIFI DEVICE INTERFERENCE

Another factor that can degrade the performance of WiFi links in homes is interference caused by commonly available non-WiFi devices, such as microwave ovens, Cordless Phones etc. Unlike WiFi transmissions, these devices do not sense the medium before transmitting energy into the spectrum. Different non-WiFi devices can impact WiFi links differently based on their transmit power, transmission protocol as well as their distance from the WiFi transmitters and receivers. By running Airshark [84], WiSe APs detect the presence of non-WiFi devices operating in their vicinity and report them to the controller (§2.2). In this section, we report the properties of non-WiFi interference in homes and quantify their impact on the Witt across the APs.

How did non-WiFi interference impact nearby WiFi links?

In this section, we focus on the impact of microwave oven devices on nearby links, since they were the most ubiquitous non-WiFi interferers in our deployment. Microwave ovens have a duty cycle of 50% with alternative periods of

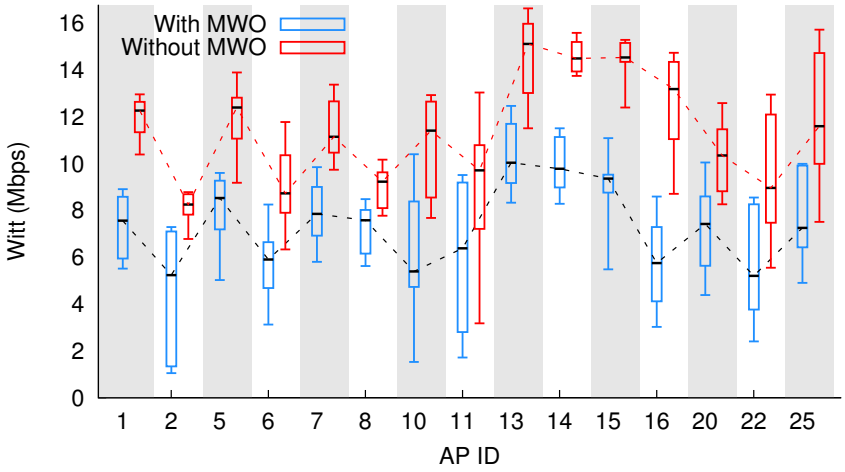

Figure 2.15: Candlestick plot comparing the estimated average Witt for WiSe APs with active WiFi links during and after the completion of microwave oven activity. 10th, 25th, 50th, 75th and 90th percentiles are shown in this figure. The dotted line compares the 50th percentile values for both cases.

approximately 8ms of active and quiet periods (60Hz cycle). We compared the Witt for the active WiSe AP-client pairs with and without microwave oven activity (within 5 minutes of the activity). This allowed us to quantify their impact on these links by comparing their performance during these two periods. Since microwave ovens mainly impact channels 8 -11, we analyzed this data for a period of 30 days during which the APs were configured to use channel 11.

Figure 2.14 shows the duration per day and the number of days (from the 30 day period) during which different APs experienced at least 20% degradation of Witt in the presence of microwave oven activity. During this period, WiSe APs 6, 7, 13, 20 experienced microwave interference on more than 20 days. Most APs experienced short periods (median 1 - 5 minutes) of microwave oven interference during active periods. While these periods are short, they can cause significant reduction in Witt at some APs (e.g., AP 2, Figure 2.15).

Figure 2.15 compares the estimated average Witt for different WiSe APs during active periods with microwave oven activity and active periods without microwave oven activity (within 5 minutes). The dotted line compares the

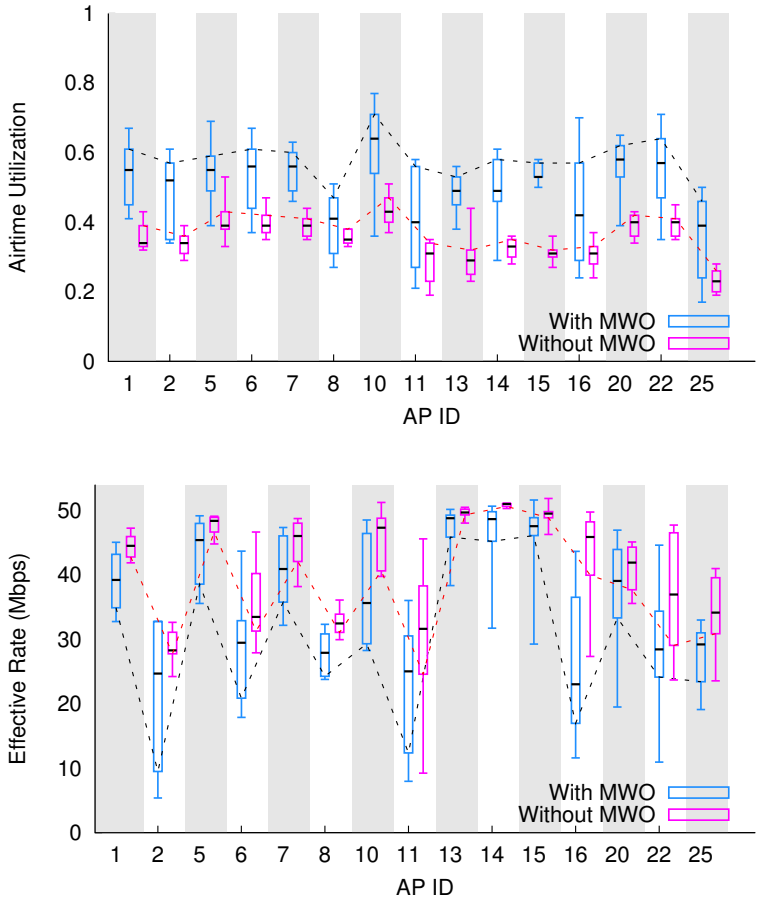

Figure 2.16: Candlestick plot comparing the airtime utilization (top) and effective rates (bottom) for WiSe APs with active WiFi links with and without microwave oven interference. 10th, 25th, 50th, 75th and 90th percentiles are shown in this figure. The dotted line compares the 25th and 75th percentile values for airtime utilization and effective rate respectively.

50th percentile values of the Witt for both periods to focus on the performance during periods of high interference. Some APs (e.g., 2, 5 and 16) experienced high performance losses during periods of microwave activity. For example, the 25th percentile Witt dropped from 7.8 Mbps to 1.5 Mbps for AP 2 (in Bldg

1), indicating a performance drop of 81% due to microwave oven activity.

Impact on airtime and effective rate. To provide greater insight into the causes of the low Witt measured at some APs during periods of interference, Figure 2.16 shows the impact of microwave ovens on factors such as airtime and effective rate (§2.3). Figure 2.16 (top) shows the average airtime utilization (10 second average) across APs during the two periods. Some APs reported significant increase in airtime utilization in the presence of microwave ovens. For example, the 75th percentile value of airtime utilization for AP 16 increased from 0.33 to 0.70, indicating an absolute increase in airtime utilization of 37 percentage points due to the microwave oven. Other APs, such as AP 10, 11 and 22 also experienced high airtime utilization in the presence of microwave ovens (over 60%).

Figure 2.16 (bottom) shows the impact of microwave ovens interference on the effective rates. At some APs (e.g., APs 13 - 15), there was little reduction during most instances of interference. Thus, these APs were mostly impacted by higher airtime utilization at the sender. Other APs, such as APs 2 and 16 experienced high reduction for the effective rates due to microwave oven interference. For these APs, both high airtime utilization and packet losses contributed to a lower Witt. This analysis shows the diversity of impact of microwave ovens on WiFi links in home environments.

Using different mitigation techniques based on the interferer type (e.g, channel hopping vs. fixed frequency) to avoid non-WiFi interference can be helpful for APs that experience high degradation of their links' performance. For example, switching the AP's WiFi channel can be helpful to avoid interference from fixed frequency non-WiFi devices (e.g., microwave ovens, Analog Phones etc.) but does not solve the problem due to frequency hopping non-WiFi devices (e.g., FHSS cordless phones). Temporarily increasing the WiFi AP's transmit power is a better solution for the latter scenario. In the next chapter (§3.4), we also show how leveraging context information about non-WiFi device activity (e.g., time of day) can also be helpful in avoiding interference from these devices when they are more likely to be used.

Unique channels	Num. APs (Overall)	Percentage (Overall)	Num. APs (Bldg 1 only)	Percentage (Bldg 1 only)
1	171	56.1%	99	61.1%
2	61	20%	33	20.4%
>=3	73	23.9%	30	18.3%

Table 2.9: Number of unique channels used by neighboring external APs as observed by the WiSe APs over a 1 month period. The table shows the overall values as well as the APs specifically observed in Bldg 1.

2.7 ANALYZING ACCESS POINT CONFIGURATIONS

Our deployment of WiSe APs across homes showed that WiFi links in dense deployments occasionally experience poor performance due to external sources. Changing channels can partially help in dealing with repeated performance issues on a channel (e.g., non-WiFi interference on a single channel).

To study the usage of channel selection algorithms by other home APs, we configured the WiSe APs to periodically scan all WiFi channels to overhear beacons from neighboring APs, especially to include external APs, on different channel. Table 2.9 shows the number of distinct WiFi channels used by these external neighboring APs observed by all WiSe APs as well for Bldg 1 only over a period of one month. It shows that around 56% of the overall 305 APs observed, used a single WiFi channel for the entire period, indicating the fact the majority of home APs used a static WiFi configuration, and are never re-assigned by residents after they are deployed. These channel configuration patterns were prevalent across different locations (as shown for Bldg 1), indicating that this property was not biased towards a particular location.

Using a static channel assignment may not be an issue in low density deployments or if the link's experience is good for most of the time. But, as discussed in §2.4, some APs observed lower Witt values for various reasons. Such APs can benefit from channel re-assignments based on the performance experienced on the current channel.

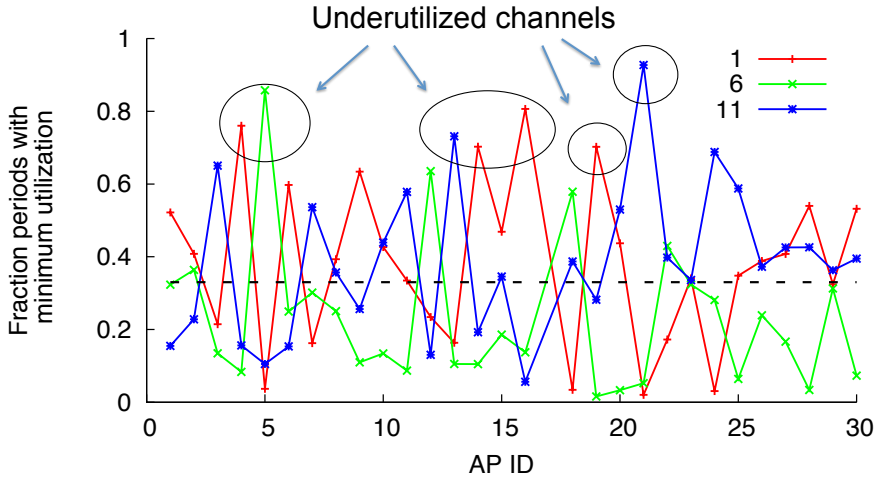

Figure 2.17: Fraction of periods with the lowest airtime utilization on a particular channel (amongst channels 1, 6, 11) across different APs (6pm - 1am) across two weekdays.

Inefficient spectrum usage due to static Access Point configurations. To compare relative activity across different channels at the 30 deployed APs, we used the traces to compute the fraction of total time with lowest airtime utilization amongst channels 1, 6, 11. These values were obtained from the secondary wireless card on each AP which hopped across channels 1, 6 and 11 in a round robin fashion (500 ms intervals). Figure 2.17 shows that some locations experienced highly uneven activity across channels (e.g., APs 5, 16 and 21). At these locations, one channel was least utilized for 80% of the time, indicating opportunity for more efficient channel assignments. This causes home WiFi APs to miss opportunities to use better WiFi channels to prevent the aforementioned problems.

2.8 SUMMARY OF WISE

In this chapter, we have tried to address the problem of understanding WiFi performance in dense residential environments. We built WiSe, a measurement

infrastructure that leverages home WiFi Access Points as vantage points and conducted a long term measurement study using a deployment of 30 WiSe APs. In summary, the following are some important observations from our analysis.

- The majority of the WiFi clients (>=80%) in our deployment of WiSe APs experienced "Good" wireless performance during most of their active periods while some clients (8%) experienced poor performance for greater than 10% of their active periods. Overall, the clients' experience was poor during 2.1% of the total active periods across APs.The location and type of deployment (e.g., Bldg 1 vs. Bldg 2) influenced the causes of "Poor" instances. In a dense deployment of private APs (Bldg 1), high airtime utilization from neighboring APs was the major cause of performance degradation while low performance due to weak signal strengths were more prevalent in a centralized home deployment (Bldg 2). We also observed high variability in link performance over the day as well as during across a period of time (e.g., high Witt values during periods of low wireless traffic in the building).

- The impact of interference (WiFi and non-WiFi) is dependent on both the link's and interferer's traffic. Majority of the interference durations were short (1 - 7 minutes) due to the bursty nature of traffic at homes. But, some links experienced extended periods of interference (tens of minutes) during periods of continuous activity due to either high airtime utilization (at sender) or packet losses (at receiver).

- Even though most interference periods were short, some had a high impact on the APs. For example, microwave ovens caused high degradation of Witt at some APs (>50%).

- Majority of APs (56%) observed at homes used static channel configurations over time (30 days) indicating that they rarely or never get configured once they get deployed. This led to inefficient usage of the spectrum and underutilized WiFi channels at many locations. It also indicates that majority of home APs do not perform any adaptations during periods of poor WiFi experience.

3 OUTSOURCING COORDINATION AND MANAGEMENT OF HOME WIRELESS ACCESS POINTS THROUGH AN OPEN API

3.1 MOTIVATION

Networking at homes continues to get complex over time requiring users to configure and manage them. Central to home networking infrastructure, are wireless Access Points (APs), that allow a plethora of WiFi-capable devices to access Internet-based services. In many dense urban environments, a large number of APs (from different AP vendors) and their associated clients are in range and cause interference to each other. Individual home users neither have the sophistication nor the patience to frequently tune their wireless APs with optimal settings. In our long term measurement study using 30 WiSe home Access Points (Chapter 2), we observed the following major wireless performance issues:

- **WiFi Interference.** Home APs can experience high airtime utilization/channel contention due to factors such as high external traffic and/or use of low PHY rates by legacy clients. During periods of active traffic from such transmitters, the average airtime utilization at the neighboring APs increased upto 70%. Such activity from a single AP can increase the congestion experienced by nearby APs operating on the same channel. Also, some links can occasionally experience hidden terminal (HT) style interference from nearby APs, resulting in packet losses at the clients.
- **Non-WiFi device interference.** Unlike the 802.11 protocol, some non-WiFi devices (e.g., microwave ovens, cordless phones) do not backoff to existing wireless activity on the same channel and can cause packet losses at the receiver due to overlapping transmissions with WiFi packets. We observed that the microwave ovens caused more than 50% degradation in link performance across multiple locations.

- **Inefficient and static channel assignments.** More than 50% amongst around 300 APs observed during a 1 month period used a single static WiFi channel. This causes APs to miss multiple opportunities to use better WiFi channels to prevent the aforementioned problems. Also, it indicates that a majority home APs do not react to intermittent interference issues that can reduce wireless throughput.

In each of these cases, If a home AP is able to determine the wireless *context* (e.g., traffic information, congestion, non-WiFi activity) at its neighboring APs and WiFi channels, it can determine the best remedial measure. Individual APs, however, do not have the full view of wireless activities, since each AP can only observe nearby activity on its current channel. If the entire set of nearby APs can aggregate this information at a , the latter can then instruct the participating APs on potential adaptations.

A vendor-neutral API and a cloud-based management service for home wireless APs. Enterprise WiFi deployments also suffer from similar wireless pathologies. Most enterprise WiFi solutions today deploy a centralized controller service located either on the premises [36] or in the cloud [64]. Our approach is to also provide a *centralized controller service for home WiFi APs* in multi-dwelling units to enable co-operation and co-ordination between them. But, residential WiFi deployments differ from enterprise WiFi networks due to following reasons, which makes it non-trivial to use same mitigation techniques and enterprise-style controllers for home APs.

- Residential wireless deployments typically consist of co-existing heterogeneous WiFi APs with each AP individually owned by the residents. Enterprise WiFi networks usually consist of Access Points provided by a single vendor and use proprietary controllers.

- Enterprise WiFi networks can use a tightly coupled control as well as data plane with very low latencies between the controller and the WiFi APs (order of micro-seconds) [36, 88]. There is no concept of centralized data plane in the case of the residential WiFi networks. Furthermore, it is only

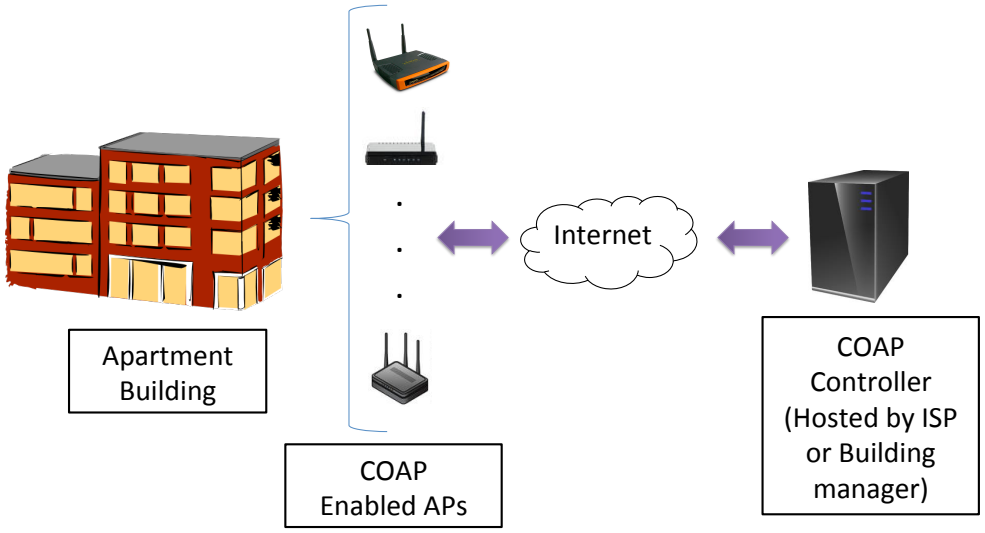

Figure 3.1: An example of COAP deployment within a residential apartment building consisting of diverse COAP capable home APs and a cloud based controller.

possible to create a loosely coupled control plane for both residential DSL and cable networks due to the higher latencies (order of milliseconds) between the APs and the cloud or the first aggregation point where a controller can be placed (e.g., ISP's central office) for a single multi-dwelling unit.

- Access Points in enterprise WiFi networks are carefully placed to obtain optimum coverage while residential multi-dwelling buildings consist of an ad-hoc placement of APs.

In this chapter, we propose *an open, vendor neutral API for home APs* that can be supported by different commodity Access Point manufacturers to allow *third-party cloud-based controller services to be designed, implemented, and deployed for RF management of home APs*. The concept of Software Defined Networking (SDN) for WLAN management using open APIs for managing APs is gaining popularity in enterprise WLANs [95]. Our proposed API, COAP, extends the OpenFlow SDN framework [63] and uses the open source

Function	Description
APConfigManager: Module to receive configuration commands from the controller and execute them at the AP	
SetParameters(channel, power)	Configures the AP to a particular channel and/or transmit power.
SetAirtimeAccess(slotDuration, transmitBitmap)	Manage airtime access of APs by throttling/slotting packet transmissions (§3.3).
BasicStatsReporter: Module to report aggregate wireless statistics to the controller	
GetNeighborInfo()	During idle periods, the AP scans all WiFi channels for neighboring APs' beacons on each channel. Logs the each AP's MAC address (hashed) and RSSI and reports this information to the controller.
GetAirtimeUtilization()	Captures the current channel's airtime utilization over the input time duration.
GetClientInfo()	Get information about associated clients (e.g., device type, signal strength).
GetLocalLinkStatistics()	Get packet transmission statistics per local link (e.g., total packets sent, received, retried).
GetTrafficInfo()	Report meta-data about the current traffic: traffic type, source, session duration (§3.4).
DiagnosticStatsReporter: Module to report more detailed wireless statistics for diagnosis purposes	
GetNonWifiDevices()	Report neighboring non-WiFi activity (using Airshark [84]): device type, duration etc.
GetPacketSummaries()	Get fine grained MAC layer packet statistics for overhead links (for debugging purposes).

Table 3.1: The different functions implemented by the COAP APs. Most functions are fairly simple to implement. In Appendix B, we further discuss the OpenFlow protocol extensions required for COAP.

Floodlight controller [13] to provide wireless management capabilities. They are complementary to recent proposals [91, 102] that use SDNs to manage residential wired broadband networks.

3.2 COAP FRAMEWORK AND API

We now present the details of the COAP framework that centrally manages and enables cooperation between APs across neighboring homes using a cloud controller (Figure 3.1). An important aspect of the framework is that it COAP only requires a software upgrade for commodity home APs and doesn't require any client device support or modifications.

Framework details

To participate in the COAP framework, APs need to expose a vendor-neutral open API (Table 3.1) to communicate with the COAP controller. COAP is complementary to SDN proposals ([91, 102]) that manage residential wired broadband networks. To implement COAP, we extend the popular OpenFlow [63] SDN framework to implement the modules shown in figure 3.2. We implement the COAP controller modules over Floodlight [13] (an open-source Java-based OpenFlow SDN controller) to communicate with the OpenFlow module at the APs. Other management frameworks such as CAPWAP [51], Reference Design Kit [14] and TR-069 [28] can also be used to implement the COAP API.

Our implementation consists of around 9000 lines of code (LOC) for the controller, 4000 LOC for the COAP APs and 800 LOC to implement OpenFlow wireless extensions. Following are additional implementation details about the COAP framework:

Access Points. In our current implementation for OpenWrt [12] based Access Points, we use the open-source *click* [6] framework to implement the statistics gathering capabilities of the *BasicStatsReporter* and *DiagnosticStatsReporter* modules for the COAP APs. We use it to parse the packet headers of the local and neighboring WiFi links to obtain anonymized aggregate (or detailed) link level statistics. We use Airshark [84] to provide the non-WiFi device detection capability using commodity WiFi cards. The OpenFlow module at the COAP APs interfaces with Airshark and click-based modules (figure 3.2) using *netcat* and a standard messaging format. The netcat interface allows AP vendors to replace the click and Airshark modules with their own implementation for reporting statistics (e.g., using SNMP) described in Table 3.1.

The *APConfigManager* module interfaces with the OpenWrt configuration tool (*luci*) to perform AP level configurations (e.g., channel, transmit power and traffic shaping). It is important to note that WiSe is not dependent on the OpenWrt platform. It can interface with other configuration protocols (e.g., netconf [52]), depending on the AP platform.

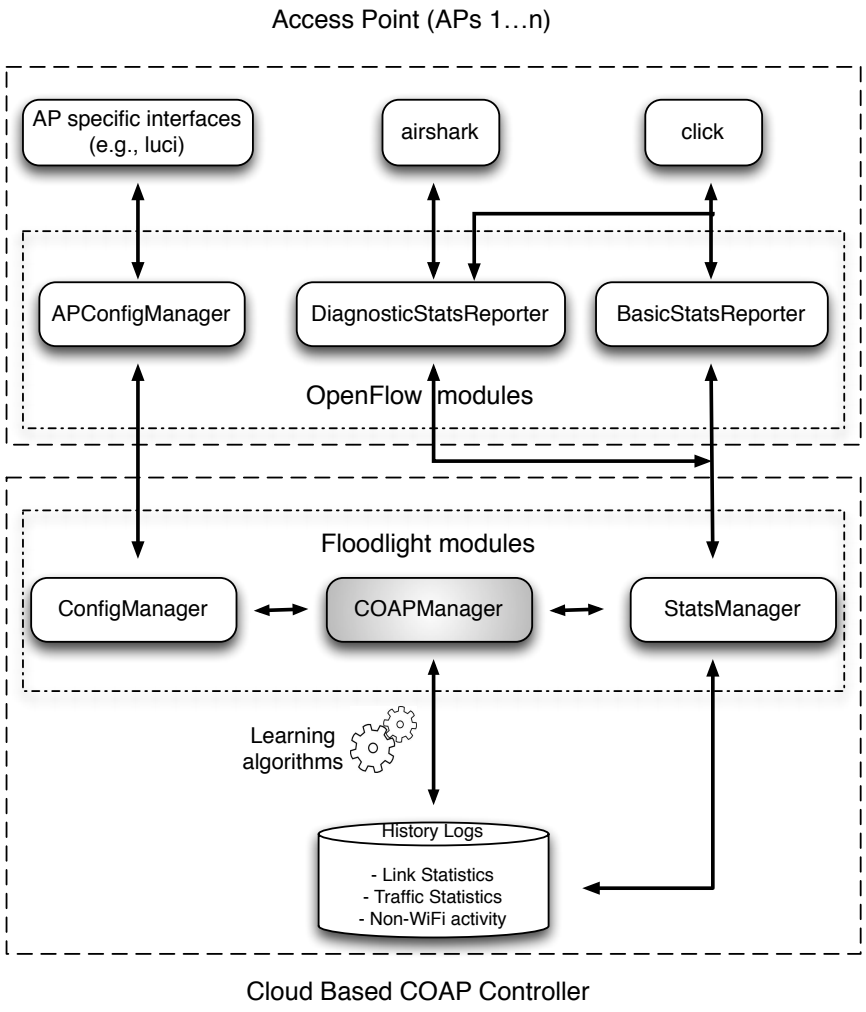

Figure 3.2: Access Point and controller components of the COAP framework. The "COAPManager" is the main module which manages other modules at the controller.

We have instrumented the ath9k wireless drivers for our 802.11n based APs to support APIs related to airtime management, *SetAirtimeAccess(slotDuration, transmitBitmap)* for interference mitigation (§3.3). For example, to throttle the airtime access of an AP, we disable the AP's transmit queue (using the

AR_Q_TXD register) to block packet transmissions for the required period of time. Since this is a software-only modification, the underlying implementation of this feature can be driver specific and transparent to the COAP API.

Controller. The COAP controller is implemented over the Java based open source OpenFlow controller, Floodlight, and currently runs on a standard Linux server. Floodlight allows developers to use an existing module or implement their own modules within the framework. All COAP controller modules, *StatsManager, COAPManager and ConfigManager*, are implemented as modules within Floodlight.

All server modules and tasks are managed by the *COAPManager* module, through which it manages and configures the connected APs. In the COAP framework, the COAPManager module allows administrators to implement custom policies based on their management requirements. The controller also maintains logs about previous wireless activity (WiFi and non-WiFi) to learn and build models to predict wireless usage characteristics (§3.4).

Wireless extensions for OpenFlow. The OpenFlow communication protocol currently consists of capabilities to exchange switch related statistics (e.g., statistics per switch port, flow, queue, etc.). We augmented this feature to exchange COAP related wireless statistics. (Table 3.1). To transmit wireless configuration updates from the controller to the APs (e.g., switch channel, throttle airtime), we extended the OpenFlow protocol to use a mechanism analogous to the one used to send OpenFlow switch configuration updates. In Appendix B, we further discuss these OpenFlow protocol extensions.

Deployment and Traces. We deployed 12 OpenWrt based COAP APs in a large apartment building (Bldg 1 used for WiSe project) to gather statistics as well as experiment with the basic management capabilities. We complemented this effort with a local testbed to perform additional experiments for this work. We also analyze traces from the WiSe deployment to motivate COAP based applications due to its larger scale and duration.

The COAP framework enables administrators to implement diverse applications at the controller and leverage co-ordination and co-operation between neighboring Access Points to improve overall wireless performance. To get the maximum benefits from the framework, a single controller should be used for managing APs in a multi-dwelling unit (e.g., an apartment building). This is because, aggregating results from multiple APs within the same building can allow the controller to create a better wireless performance map for the building by combining information from multiple Access Points. We now discuss these applications through a combination of real world deployments as well as controlled experiments.

Application #1: Channel Configuration

In enterprise WLANs, AP configurations (e.g., channel, transmit power) are usually managed by a vendor specific central controller. In home networks, such configurations are usually performed manually or done by each AP individually based on local observations. Furthermore, such configurations were observed to static for more than 50% of home APs which can lead to inefficient utilization of the spectrum as discussed earlier in §2.7.

In the COAP framework, the central controller can perform these functions for the home Access Points. For example, by using information about the recent wireless activity on different channels (e.g., total airtime utilization due to WiFi and non-WiFi sources) from an AP's neighbors, the central controller can perform better channel configuration per-AP compared to local-only observations. To motivate this application, we first analyze the feasibility of gathering channel utilization statistics from neighboring APs to augment an AP's local view of the spectrum.

How did the AP density vary across different locations?

We analyzed the traces from the earlier WiSe deployment to determine the number of neighboring APs observed across different locations. Figure 3.3

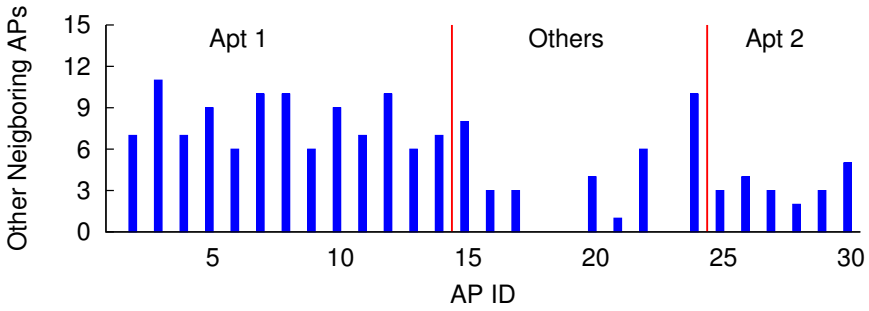

Figure 3.3: The number of neighboring APs with RSSI > -55 dBm across different locations.

shows the number of other APs observed by each AP (across all channels) with a high RSSI ($> -55\text{dBm}$) across 2 apartment buildings and other diverse locations. It shows high (Bldg 1) to moderate (Bldg 2) density of APs with the number varying between 2 - 8 APs across most locations. As discussed next, information about wireless activity collected from these neighboring APs can be exploited by the controller to augment a single AP's view of the spectrum.

Can the controller leverage neighboring APs for better channel selection?

A residential WiFi AP can only observe the wireless activity for the channel it is currently on. This might cause the AP to miss opportunities of using a better channel with lower congestion and/or wireless interference. We studied the feasibility of *estimating the airtime utilization for other channels at an AP by using information from nearby APs (RSSI > -55 dBm) on those channels.* The underlying intuition is that nearby residential WiFi APs will observe similar wireless activity.

Figure 3.4 shows the CDF of the Pearson's correlation coefficient between the timeseries per-channel airtime utilization values (1-minute averages) observed by pairs of neighboring APs. It shows that more than 60% of the AP pairs whose mutual RSSIs were higher than -55 dBm (using beacon RSSI) exhibited a high correlation coefficient of more than 0.8. Furthermore, almost all of these AP pairs exhibited a correlation higher than 0.6. In contrast, only 15% of all possible AP pairs exhibited a correlation of more than 0.8 between their

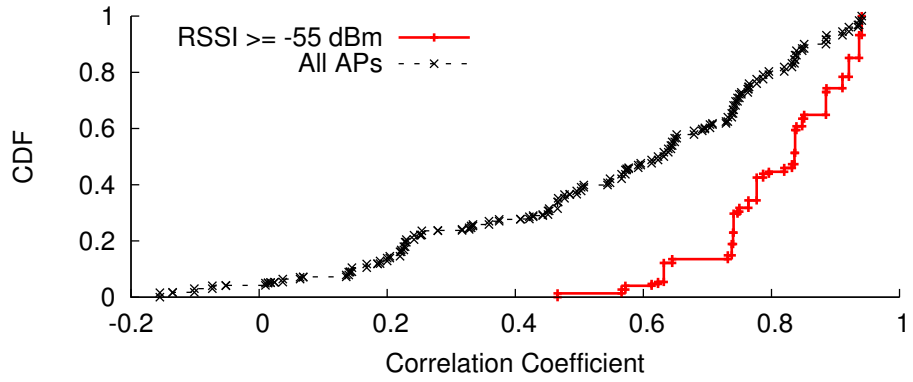

Figure 3.4: CDF of the Pearson's correlation coefficient for time-series per-channel airtime utilization observed by pairs of neighboring APs.

channel utilization values. This result shows that it can be possible to exploit the *spatio-temporal locality* of wireless activity between nearby APs to estimate airtime utilization on other channels. Thus, sharing information about observed wireless activity can enable the COAP controller to perform better centralized channel configuration [29] for the participating COAP APs.

What were the performance improvements due to centralized channel configuration?

To study the benefits of using the COAP framework for channel assignment, we used the COAP controller to perform dynamic channel assignments and reduce the airtime utilization (and thus, contention) in our residential deployment of 12 APs. Due to the sparsity of our AP deployment, we used a secondary wireless card on our APs to collect airtime utilization across all channels in a round robin fashion. In this manner, we simulated the usage of neighboring APs to collect airtime utilization information. Over a span of 6 days, we employed 2 different channel configuration strategies on alternate days (1 day per configuration): per-day random channel assignment, and dynamic channel assignments using COAP to to reduce airtime utilization per AP. In the second scenario, we used airtime utilization statistics during the most recent 5

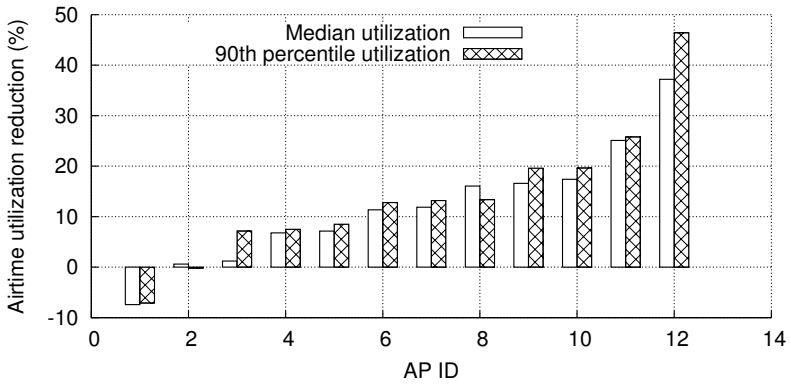

Figure 3.5: Percentage reduction in median and 90th percentile airtime utilization in our deployment of 12 COAP APs using dynamic channel configuration.

minute interval to select the best channel per-AP. These channel configuration updates were only applied during periods without high WiFi activity.

Figure 3.5 shows the reduction for median and 90th percentile airtime utilization values across our deployment of 12 APs for the COAP-based dynamic channel configuration over the random channel assignment scheme, which is the case for majority of homes today due to lack of centralized management. *It shows that the dynamic scheme performed better than a random channel assignment scheme for 10 out of the 12 APs. Four out of the 12 APs (APs 9 - 12) experienced 20% or more (upto 47%) reduction in airtime utilization.* This shows the potential gains that can achieved through a centralized approach to home WLAN management. Further proliferation of WiFi-capable devices and high volume traffic (e.g., video) will further increase the utility of COAP based approaches for balancing and reducing channel contention.

Application #2: Airtime Management

For the COAP framework, we developed the *SetAirtimeAccess(transmit_bitmap, slot_duration)* API (Table 3.1) which provides a lightweight and flexible mechanism to the COAP controller to manage the airtime access of the COAP APs. We

propose this API for solving two problems that can be occasionally experienced by home APs (§2.5): (i) High channel utilization caused by neighboring APs, (ii) hidden terminal (HT) style interference resulting in packet losses at the client. The intuition is to mitigate these scenarios by controlling the packet transmission or channel access patterns of the interfering APs. For example, the impact of hidden terminal (HT) style interference can be mitigated by preventing overlapping transmissions between the two interfering APs. Similarly, channel congestion experienced by high priority flows (e.g., video) can mitigated by reducing the airtime access of nearby APs with low priority flows (e.g., bulk downloads). Traffic classification techniques [68] can be used to detect flow types and identify the more sensitive flows (e.g, video).

API details

We use two mechanisms to manage the airtime access of COAP APs (Figure 3.6): $\mathtt{Throttle}(AP_x)$ and $\mathtt{Slot}(AP_x, AP_y)$. For this purpose, the controller divides time into small "slots" (10 - 20 ms). The controller can enable/disable transmissions from a COAP APs on a *per-slot* basis to manage its airtime access. $\mathtt{Throttle}(AP_x)$ is used to limit the transmissions of an AP to reduce the airtime utilization and thus channel contention at the neighboring APs. $\mathtt{Slot}(AP_x, AP_y)$ is used by the COAP controller to configure two APs to transmit in *non-overlapping time slots*. This is useful for mitigating HT style interference during which overlapping packet transmissions cause high packet losses at client devices. We note that this mechanism is only applied to data packets and thus doesn't impact any control or management packets (e.g., beacons) transmitted by the WiFi APs.

In contrast to enterprise-centric centralized approaches (CENTAUR [88]) which explicitly schedules the flow data to mitigate interference, it is not possible to do across home APs due to the lack of centrally a controlled data-plane But the loosely-coupled centralized control plane still allows the cloud-based COAP controller to co-ordinate transmissions across neighboring home APs. This approach is motivated by the RxIP [61] mechanism for co-ordinating airtime sharing between APs. But, RxIP uses a distributed token based co-

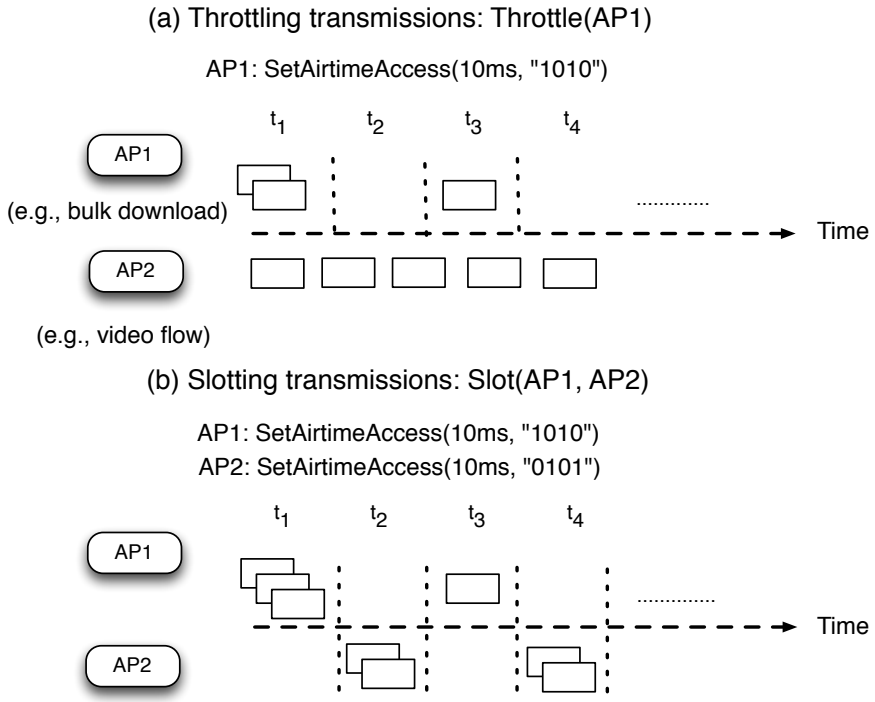

Figure 3.6: Two mechanisms to manage airtime access of COAP APs: $\mathtt{Throttle}(AP_x)$ (top) and $\mathtt{Slot}(AP_x, AP_y)$ (bottom). $\mathtt{Throttle}$ limits the airtime access of a single COAP AP while $\mathtt{Slot}$ coordinates the airtime access of two interfering COAP APs.

ordination protocol which increases the control messaging overhead in dense residential environments.

Controlling airtime access. To use this API, COAP APs are configured with the following two parameters: *"transmit bitmap"* and *"slot duration"*. The slot duration (e.g., 20 ms) determines the time granularity to use for airtime access management (enable/disable transmissions). The per-AP transmit bitmap specifies the slots in which the AP is allowed to transmit. For example, a transmit bitmap of "1010" indicates that the AP is only allowed to transmit in time slots 0 and 2 and disabled in slots 1 and 3 (Figure 3.6, top). This pattern

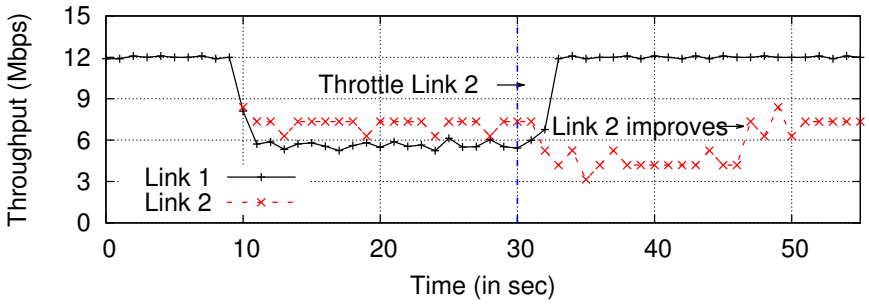

Figure 3.7: Mitigating congestion: Concurrent traffic from a poor link (Link 2) reduces the TCP throughput of Link 1; throttling link 2 to causes 2x improvement for link 1.

for transmissions is continuously repeated at the APs. This example represents a throttle value of 50% since the AP is only allowed to transmit data packets during 50% of the time slots.

To synchronize APs for the $\texttt{Slot}(AP_x, AP_y)$ mechanism, the COAP framework uses a passive mechanism from [85] to synchronize the APs' clocks to $<$ 100 microseconds error which is much smaller than the slot duration of 10 - 20 ms. This slot duration is large enough for WiFi frame aggregation while not impairing the most delay sensitive traffic, such as VOIP (§3.5). Based on traffic type, it is possible to use a larger slot duration. For example, bulk downloads are not delay sensitive and it is possible to use larger slot durations during such flows. The time synchronization error necessitates a lower bound on the slot duration so that the error doesn't impact link performance during hidden terminals. Algorithms to dynamically select the most optimal slot duration based on flow type is a subject for future research.

We'll now explain the usage of this API with the following two example scenarios.

Example Scenarios. Figures 3.7 and 3.8 show results from a couple of controlled experiments in our testbed. In the first scenario (Figure 3.7), the iperf throughput of a good link (Link 1) with a throughput of 12 Mbps degrades in the presence of overlapping transmissions from a poor link (Link 2) which uses

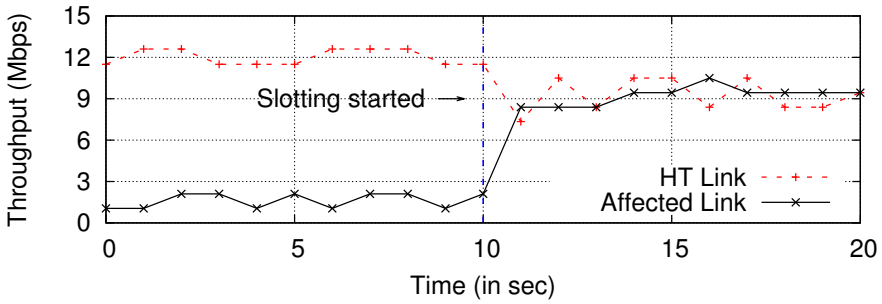

Figure 3.8: Mitigating hidden terminal interference: Overlapping transmissions from a hidden terminal degrades a link performance; slotting the two transmissions improves the affected link's performance by 4×.

low PHY rates (at t = 10). At t = 30, the controller throttles link 2 by limiting its transmissions to alternate 10 ms time slots (50% throttle), This improves the performance of link 1 from 6 Mbps to 12 Mbps due to the sufficient airtime available to get the desired throughput. At t = 45 seconds, the link quality of link 2 improves, allowing to get higher throughput (from 4Mbps to 7.8 Mbps) with the 50% airtime throttle. Thus, as opposed to approaches to control a link's airtime utilization by controlling its maximum throughput, this API provides a stricter control over an AP's transmissions which is independent of the varying link + PHY layer properties (e.g., signal strength and data rates).

In the second scenario (Figure 3.8), a link experiences poor throughput (2 Mbps) due to interference caused by another AP which acts as a hidden terminal (HT) and transmits data at 12 Mbps. The controller protects the affected flow by configuring the two APs to transmit in alternating and non-overlapping time slots (using transmit bitmaps of *"1010"* and *"0101"*). The protection provided by slotting increases the throughput of the affected link to 9 - 10 Mbps while the HT link still obtains a throughput of 8 - 10 Mbps.

Testbed evaluation

We setup a testbed deployment consisting of 802.11n based 6 COAP APs and 6 clients to emulate two targeted problem scenarios and evaluate the benefits of using the airtime management APIs: (a) HT style client-side interference

and (b) channel congestion experienced by some APs. Out of these 6 links, 3 links consisted of video traffic and we emulated bulk data downloads on the remaining 3 links using iperf traffic. This represents a mix of heavy traffic sources in homes today. We setup a video server to generate video traffic in our testbed and and used the open source Strobe Media Playback (SMP) media player [73], a popular browser based HTTP based video streaming client to generate TCP based video traffic for the video clients. The SMP client was also used to log various metrics related to quality of the video (e.g., frame drops, stalls, buffer period etc.). We benchmarked the performance using a set of 3 metrics for evaluation: (i) the TCP throughput achieved by a link, (ii) MAC layer packet loss rates and (iii) the frame drop rate indicating the percentage of total video frames dropped by the SMP player.

HT interference scenario. In the first experiment, we setup 3 hidden terminal scenarios by creating hidden terminal (HT) like conditions between the 3 video clients (streaming a 10 Mbps HD video) and the 3 bulk clients with 10 Mbps traffic. Figure 3.9 shows the performance metrics for the video (clients 1 - 3) and bulk flows (clients 4 - 6) with DCF and using the *Slot* mechanism to mitigate interference. In the DCF scenario, all three video flows experience high MAC layer losses (40% or higher) due to interference. Furthermore, 2 clients experience a high frame drop rate (20 - 40%) due to the degraded throughput of the links (< 8 Mbps).

By using the $\mathtt{Slot}(AP_x, AP_y)$ mechanism to configure interfering links to transmit in non-overlapping time slots, the performance of the video links improve considerably (> 10 Mbps) while the bulk flows still obtain similar throughput. Furthermore, the MAC losses for the video links reduced by more than 50% indicating more efficient airtime utilization while the frame drop rate reduced to 0 - 4% across all video links indicting good video quality. In the COAP framework, we incorporate techniques from prior PIE [89] and WiSe (Chapter 2) that correlates packet losses and activity of nearby links to detect the presence of HT style interference.

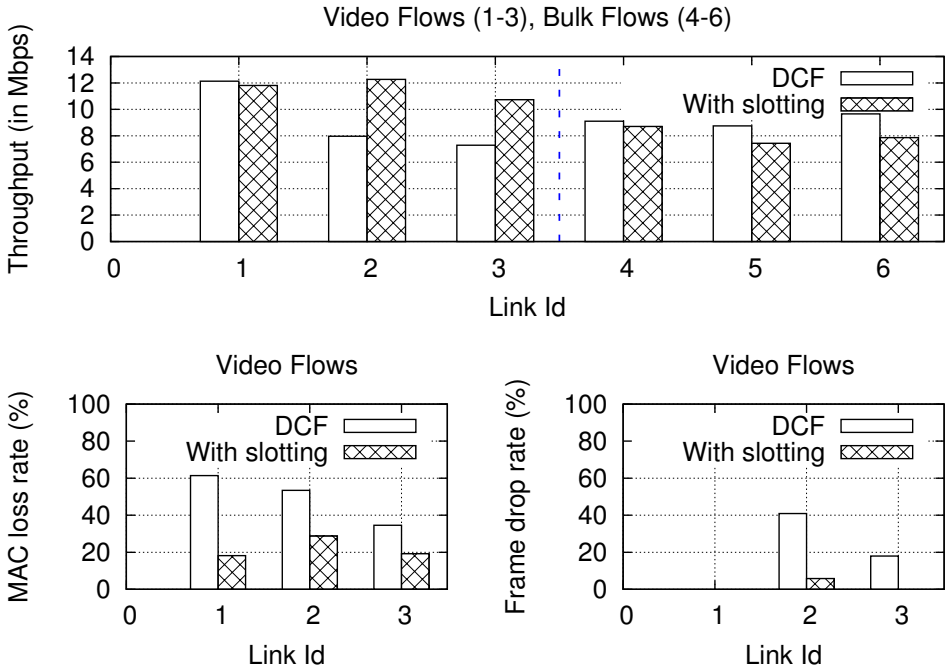

Figure 3.9: TCP throughput (top), MAC layer loss rate (bottom-left) and frame drop rate (bottom-right) for 3 video links experiencing HT style interference with and without the airtime management. Using non-overlapping time slots for transmissions reduces MAC losses of video flows by more than 50% and frame drop rates to < 4%.

Mitigating channel congestion. We emulated a congested channel scenario by placing 2 medium bit-rate (5 Mbps) HTTP video links, 1 high bit rate (10 Mbps) HTTP video link and 3 bulk flows (10 Mbps) on the same WiFi channel (Figure 3.10). In the DCF scenario, the channel congestion caused the high bitrate video link (link 2) to experience high drop rates (40%) while the medium bit rate video links (link 1, 3) experienced good performance due to the lower throughput requirements.

To reduce congestion, the COAP controller throttled ($\mathtt{Throttle}(AP_x)$) the 3 bulk flows (4 - 6) to 50% airtime access. While the throughput of the bulk flows went down by 50% the performance of the video link 2 improved (4% frame drops) due to the higher throughput of the link (11 Mbps). This shows

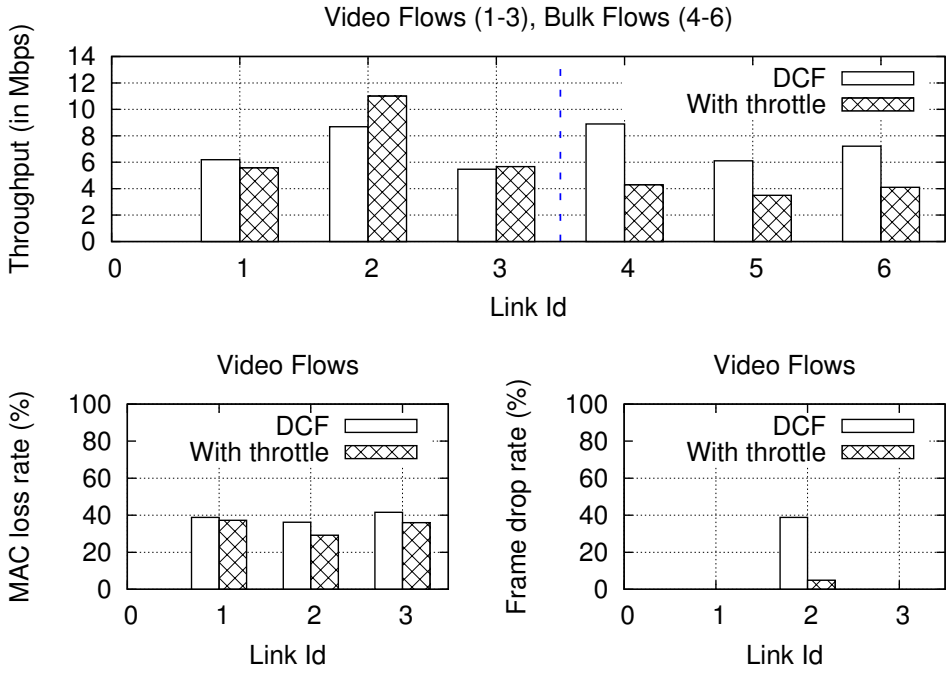

Figure 3.10: TCP throughput (top), MAC layer loss rate (bottom-left) and frame drop rate (bottom-right) for 6 links (3 video, 3 bulk flows) experiencing channel congestion with and without the airtime management. Throttling the bulk flows to 50% airtime access improves the performance of video link 2.

how the COAP framework can improve the performance of the more sensitive flows (e.g., video) leveraging co-operation between neighboring APs for airtime access. The COAP framework collects information about client and traffic type (§3.4), which can be used as hints to identify the more sensitive flows.

3.4 CONTEXT BASED ACTIVITY PREDICTION

The controller can also analyze logs about prior wireless activity collected from COAP APs to employ *learning techniques* and mine "context-related" information. We present two examples in this section: (i) Predicting future non-WiFi activity using microwave ovens as an example, (ii) using client device type and traffic information to predict traffic characteristics.

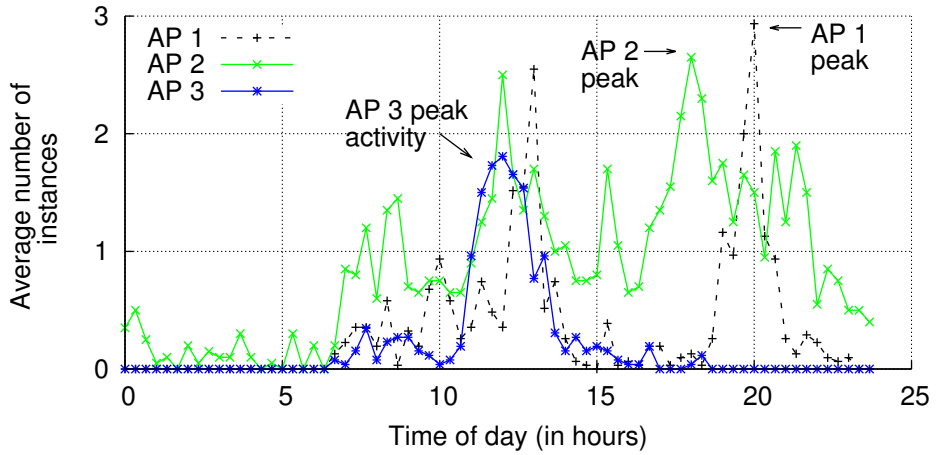

Figure 3.11: "Activity vectors" for microwave oven activity observed by three different APs (2 weeks).

Modeling and learning about non-WiFi activity

Prior work such as TIMO [45] use MIMO based techniques to mitigate the impact of non-WiFi interference. In this work, we investigate the feasibility of an orthogonal approach to avoid non-WiFi interference by predicting future activity. The intuition here is that certain non-WiFi interferers (e.g., microwave ovens) get used at specific periods of the day. Our goal is to determine whether it is possible to learn from past instances and predict future activity using commodity WiFi APs today.

In the COAP framework, the controller maintains time-series statistics about non-WiFi activity observed by COAP APs using commodity WiFi cards [84]. If a non-WiFi interferer is regularly active during specific time periods of the day, the controller can intelligently perform pre-emptive configuration of the COAP AP that gets consistently impacted by the device. We now analyze the "predictability" of future non-WiFi activity using microwave ovens as an example.

Building "activity vectors". To represent a particular non-WiFi device type's activity (e.g., microwave oven) observed by an AP, we create a per-AP

"activity vector" for each non-WiFi device. Each *activity vector* is represented as a vector (V) to represent the non-WiFi activity during the 24 hours of a day:

$$V :< c_1, c_2, ..., c_n > \forall 1 \leqslant i \leqslant n \ , c_i \geqslant 0.0$$

Each element (c_i) in the vector records the average number of daily instances of non-WiFi activity observed during a time of day "bin period" (e.g., 8pm to 8:20pm - using 20 minute *bin periods*). The activity vectors are constructed by aggregating information about the non-WiFi activity over a span of multiple days. Figure 3.11 shows the activity vectors and peak-activity periods for microwave oven activity observed by three different home APs. The activity vectors are able to capture time periods of high microwave activity (e.g., 11am - 2pm for AP 3). Furthermore, there is large diversity in the temporal activity at these APs (observed across different locations). Our goal is to analyze the fidelity of these activity vectors over a period of time across different locations.

Predicting non-WiFi activity.

We analyzed microwave oven activity traces collected from our deployment of 30 residential WiSe APs (Chapter 2) to build the aforementioned activity vectors over consecutive days using different "time spans" (1, 2, ... 30 days). This allowed us to capture their aggregate activity characteristics over different time periods. For a given time span of k days, using per-AP activity data (total of d days), we obtained a sequence of activity vectors ($V_1^k, ..., V_{d/k}^k$). Using all time spans ($k = 1...30$), we obtained 30 different sequences of activity vectors:

$$(V_1^1, V_2^1, ..., V_d^1), ..., (V_1^{30}, ..., V_{d/30}^{30})$$

We used these sequences of activity vectors to determine the predictability of future non-WiFi activity (V_{i+1}^k) based on the most recent record of non-WiFi activity (V_i^k). For each *time span, k* ($1 \leqslant k \leqslant 30$), we computed the per-AP Pearson's correlation coefficient between consecutive activity vectors and

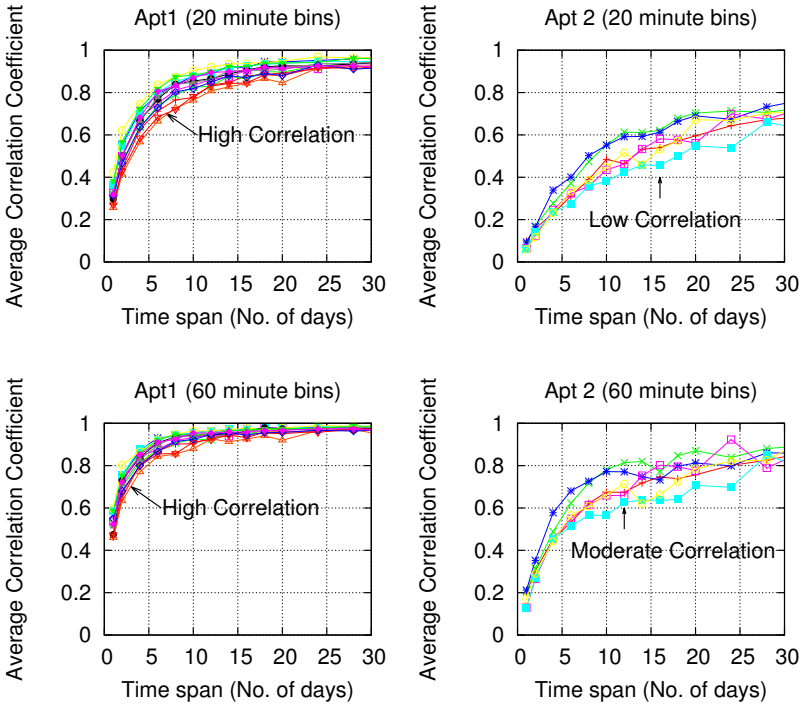

Figure 3.12: Average correlation between consecutive microwave oven activity vectors for APs in apartment buildings Bldg 1 and Bldg 2 as a function of the "time span". The top graphs use "activity vectors" with 20 minute bin periods while the bottom graphs use 1 hour bin periods.

averaged them:

$$\frac{1}{n-1}\sum_{i=0}^{(d/k)-1} \mathrm{corr}(V_i^k, V_{i+1}^k)$$

A high correlation for a given time-span (e.g., k = 10 days), indicates high similarity for non-WiFi activity between the consecutive 10 day periods and indicates the fact that using information about non-WiFi activity from the previous 10 days can be used as an indicator for future activity.

Results. For the correlation analysis, we experimented using activity vectors with different *bin periods* to capture microwave oven activity at different time granularity. We discuss the results for two different bin periods - 20 minutes (figure 3.12, top) and 1 hour (figure 3.12, bottom). The results are divided between the APs located in two residential building – Bldg 1 and Bldg 2. In Bldg 1, the APs were deployed within individual apartment units across 5 floors of the building. In Bldg 2, the APs were deployed in common areas and shared by the different tenants within the building.

The results provide some interesting insights. With 20 minute bin periods (figure 3.12, top-left), the correlation between consecutive activity vectors for short time spans (1 - 2 days) across both buildings was less than 0.5. This indicates that short time spans were not able to capture the diversity in microwave oven activity. Using time spans greater than 5 days resulted in moderate (0.6) to high correlation (0.8) for microwave oven activity at all APs in Bldg 1. Using a time span greater than 10 days did not provide additional gains. By using more coarse grained bin periods of 1 hour (figure 3.12, bottom-left), it was possible to capture this information in a shorter time span of 3 - 4 days. *This indicates that a span as low as 5 days could capture enough information to predict future microwave oven activity nearby APs in Apt 1.*

Interestingly in Apt 2, by using 20 minute bin periods (figure 3.12, top-right), the trends did not converge even after 20 days as indicated by the lower correlation (0.6) for the activity vectors across all APs. One of the major reasons for this behavior is the location of these APs in common areas of the building. Furthermore, there were microwave ovens present in the common areas of the building. Thus, these APs captured microwave activity from different apartments as well as common areas, resulting in high variability in activity across different time spans. This resulted in lower predictability of future activity due to the high diversity of microwave oven activity as well as poor detection accuracy for farther devices. Using more coarse grained bin periods of 1 hour (figure 3.12, bottom-right) resulted in better predictability – correlation coefficient of 0.6 to 0.8 for time spans > 10 days. This analysis shows how the COAP controller can incrementally learn about such spatio-temporal properties about non-WiFi activity at different time granularity (based on the AP deployment and multi-dwelling unit). Depending on the

nature and extent of non-WiFi interference impact, *this information along with the activity vectors can be used by* COAP *controller for pre-emptive configuration of* COAP *APs (e.g., WiFi channel, transmit power) to avoid non-WiFi interference.*

Learning client and traffic context to predict future activity

Users access a diverse set of traffic services from their WiFi-capable devices such as video, VoIP, bulk transfers, gaming applications etc. Since home APs act as a single gateway to all WiFi-based devices within homes, they can capture useful coarse-grained *contextual information* about the nature of wireless activity. They include client device type information (e.g., iPhone, Google TV, laptop) as well as the traffic source ID (e.g., Netflix, Pandora). In the COAP framework, the controller can use such contextual information to predict future traffic characteristics and perform traffic aware configuration of APs.

In this section, we investigate the feasibility of predicting traffic characteristics based on contextual information - client device type, traffic source, time of day etc. This can help avoid scenarios discussed earlier in chapter 2 (§2.5) where long running and high volume flows caused long periods of high airtime utilization at neighboring APs. Following is an example scenario – *"AP 4 has started a high-volume HD video flow on his wireless TV, which is going to last for 30 minutes with 90% probability. Therefore, move adjacent APs to other WiFi channels to reduce channel contention during this traffic."*

Why device type and traffic context can help?

For the following analysis, we use the traffic traces collected from the COAP deployment to motivate the benefits of using device and traffic context to predict traffic characteristics. The traffic sources were determined by finding the TLD (Top Level Domains) names corresponding to the traffic flows. The TLD information is only needed by the COAP controller to distinguish the different traffic sources of the flows. For COAP's deployment, the TLD can be hashed and anonymized to preserve user privacy. The client device types were determined by only using the prefix of the MAC address (first 24 bits) determine the Organizationally Unique Identifier (OUI) as well as leveraging

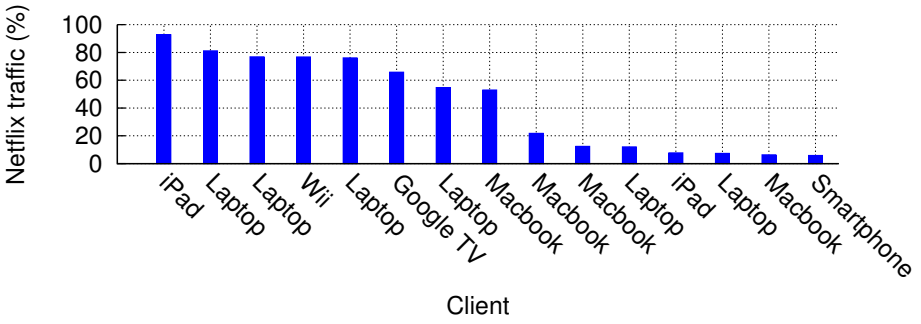

Figure 3.13: Total proportion of Netflix traffic across top clients.

hints from the host name (e.g, many devices such as iPhones use the device type in their host name). We used the traffic traces to extract "active periods" for flows (> 40 packets/second) and obtain the following traffic properties during the active periods:

- *Burst properties.* A *burst* is a time-period during which the gap between consecutive active periods is less than 10 seconds. For bursts, we compute the duration, downloaded bytes, the average and maximum download speed.
- *Session properties.* A *session* is a sequence of consecutive traffic bursts with a gap of less than 5 minutes. These periods indicate active usage of the traffic source. For each session, we compute gaps between consecutive bursts, duration, bytes downloaded and download speeds.

Client and traffic context information can be helpful in predicting traffic properties due to the following reasons:

- ***Bias in traffic usage profile by device type.*** In our traces, Netflix was the highest traffic source by volume. Figure 3.13 shows the proportion of Netflix traffic across the top Netflix client devices. Analyzing the distribution on a per-device basis reveals a good correlation between

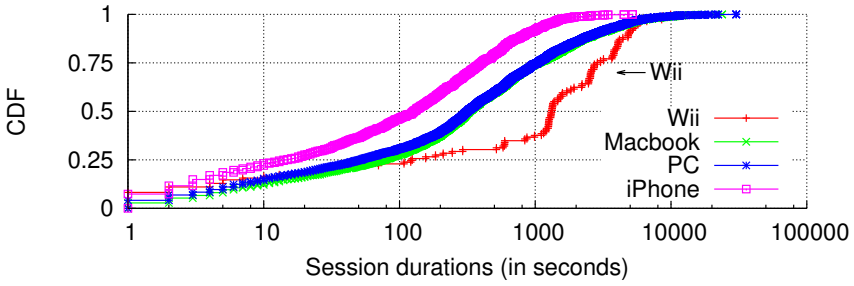

Figure 3.14: CDF of session lengths by client device type for 4 different client devices.

device and traffic usage. The top 5 client devices had a high proportion of Netflix usage (around 80%). Such properties can be exploited to predict traffic characteristics due to greater bias towards a single traffic source.

- ***Impact of device usage characteristics.*** The device type can also impact user behavior (e.g., the time that they spend on a device). For example, a user may use his smartphone for only short periods of time but use a gaming console for longer periods of time. Figure 3.14 exhibits higher session duration for the Wii devices (median of 20 minutes) compared to other devices: median duration of 5 and 2 minutes for laptops (Macbook, PCs) and iPhones respectively.

- ***Platform specific traffic behavior.*** The type of the platform (e.g., client device, OS) can impact traffic characteristics. For example, video streaming services have been observed to use different strategies [82] based on the browser, device type etc. In our analysis, we observed a similar behavior for some popular traffic sources (e.g., Netflix). Figure 3.15 shows the CDF of per-burst maximum download speeds for Netflix based traffic across different client devices. It indicates that while the maximum per-burst download speeds on Wii devices was only around 5 Mbps, it went upto 16 Mbps on PCs and Google TV. Since we do not have additional

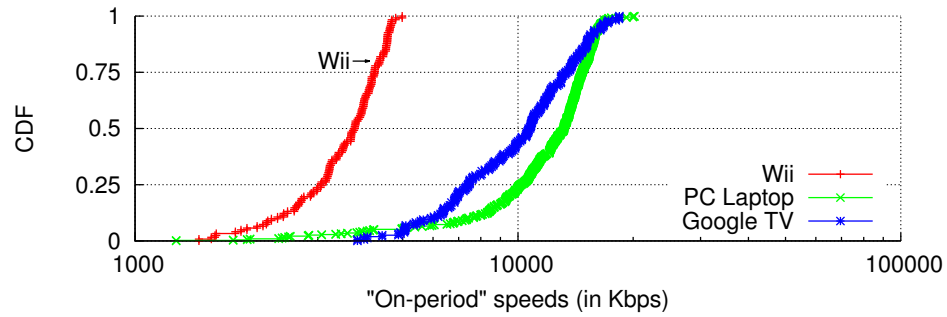

Figure 3.15: CDF of maximum download speeds for Netflix traffic during burst periods on different device types.

information about the video the underlying video traffic, we believe that the lower speeds for Wii devices might be due to the lower bit-rates used by the Netflix application or device limitations.

Predicting traffic characteristics

To quantify the benefits of using *device and traffic context information* for predicting the aforementioned traffic properties (e.g., session duration, downloaded bytes, download speed etc.), we used the machine learning tool, Weka [17] and traffic traces collected by COAP APs (30 day period) to train and evaluate prediction models for different traffic properties (e.g., session duration, downloaded bytes).

We represent the "context" as a collection of the following traffic and device related features which can be obtained at the start of a new traffic flow – *AP ID, client device id, traffic source id, time of day, day of week*. These input features were used to predict the burst and session related properties discussed earlier (e.g., session duration and bytes downloaded per session). We compared the performance of these predictions to the ones when only *AP ID, time of day and day of week* were used as input, i.e., excluding any device and traffic context. We used 10-fold cross validation on the input data to train and evaluate the prediction models.

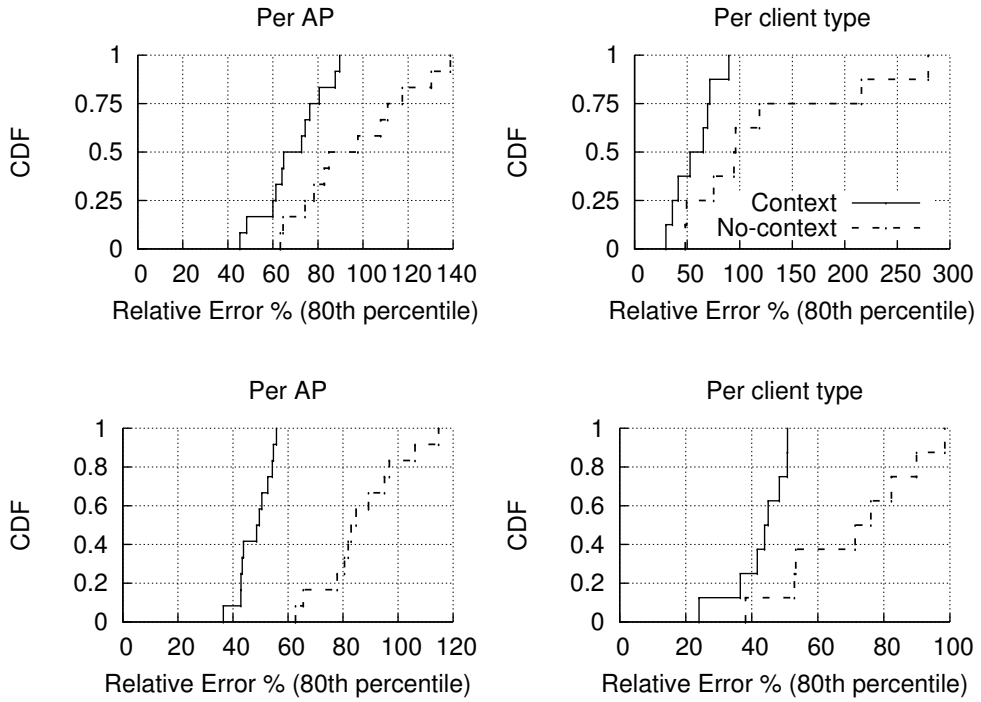

Figure 3.16: CDF of the per-AP and per-device type 80th percentile prediction error values for session lengths (top) and bytes downloaded per session (bottom) with and without using client device + traffic context.

Prediction Performance. We experimented with different algorithms within Weka and found that the REPTree algorithm [76] performed the best. It uses information gain to build a tree based model to generalize the properties of the underlying data.

Figure 3.16 (top) and figure 3.16 (bottom) show the CDF of per-AP and per-client device type (e.g., Wii, iPhone, PC) relative prediction errors (80th percentile values) respectively, for the session lengths and bytes downloaded per session. *It shows that using client device and traffic context information resulted in 2× or lower prediction errors compared to no-context scenario.* Also, these 80th percentile relative prediction errors are below 100% for the context-aware

scenario indicating that it is possible to predict these traffic properties within a factor of 2 to 3 times for the majority of cases. We observed similar results for predicting other traffic properties (e.g., download speeds). Thus, client device type and traffic context can provide valuable information to the COAP controller for predicting future wireless activity and perform potential adaptations, such as channel assignments and airtime management (§3.3).

3.5 ISSUES AND DISCUSSION

Performance overhead. The COAP framework imposes a minor traffic overhead (average overhead < 1 KBps), mostly for collecting statistics. The average CPU overhead of COAP on our router platform (512MHz CPU) is also low (< 10%) and it requires only a few megabytes to maintain in-memory state.

Incentivizing deployment. Greater participation within the COAP framework benefits everyone though better spectrum usage and interference mitigation. Furthermore, the controller can provide user-feedback about the causes of poor-WiFi performance (e.g., faulty clients) using collected statistics. The design of such user-incentive mechanisms is a topic for future research. ISPs can also drive the adoption of the COAP framework since it provides them capabilities to better understand and mitigate wireless problems in residential areas. Interest from ISPs can encourage commodity Access Point vendors to include COAP APIs within home APs.

Airtime management API and 802.11e. The Quality of Service (QoS) extensions proposed in the IEEE 802.11e standard uses different MAC layer congestion window parameters across 4 traffic classes to provide higher airtime access priority to delay sensitive traffic (e.g., VoIP and video). In the COAP framework, we let the applications use this standard based on their requirements. COAP's airtime access API is complementary to this standard and provides additional protection to sensitive flows (e.g., HTTP based video traffic) by coordinating neighboring APs' packet transmissions. Furthermore, this API can also protect traffic from hidden terminal interference at clients which is not possible with 802.11e QoS extensions.

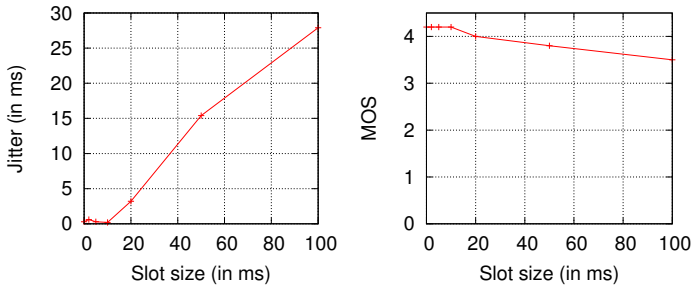

Figure 3.17: Impact of slot duration on VoIP performance (using an Internet based VoIP testing service [98]).

Selecting a slot duration for airtime management API. The slot duration should be large enough to minimize impact of clock synchronization errors for $\mathtt{Slot}(AP_x, AP_y)$ mechanism as well as allow for 802.11n frame aggregation (>= 4 ms). Also, it should be small enough to avoid performance degradation of latency sensitive flows (e.g., VoIP).

We use a passive mechanism for clock synchronization from WiFiNet [85] which uses timestamped common beacons overheard by APs to periodically determine clock offsets between AP pairs. Using an interval of 5 seconds results in clock synchronization error of less than 100 microseconds while not imposing a high communication overhead with the COAP controller. Using a smaller interval reduces errors but requires more frequent communication with the controller.

Using a large slot duration for airtime management may degrade the performance of latency-sensitive flows (e.g., VoIP) that are active at the same time. To quantify the impact of slot duration on VOIP performance, we benchmarked VoIP performance metrics [98] (using G.726 codec) as a function of the slot duration. Figure 3.17 exhibits a sharp spike in jitter values as well as drop in Mean Opinion Score (MOS) for slot durations greater than 20 ms. A MOS value of less than 4 indicates medium to poor quality VoIP calls. *Thus, a slot duration upto 20 ms provides a good trade-off between minimizing impact of*

synchronization error (< 100 microseconds) and protecting latency sensitive flows (< 1% slot synchronization error).

3.6 SUMMARY OF COAP

In this chapter, we make the following main contributions:

- We provide an overview of the COAP framework which uses an open API and a cloud-based SDN controller to manage commodity WiFi APs in dense multi-dwelling units (§3.2). The cloud-based controller enables co-ordination and co-operation between neighboring residential APs to improve WiFi performance.
- Through a combination of real-world AP deployments and controlled experiments, we motivate the benefits of the COAP framework by describing two applications that benefit from co-operation between nearby residential APs: (i) efficient channel assignments resulting in *upto 47% reduction* in airtime utilization (§3.3), (ii) mitigating wireless interference issues due to channel congestion or hidden terminal style interference resulting in more efficient airtime utilization due to *50% reduction in MAC losses* and improved quality for HTTP video flows (§3.3).
- We discuss how learning about prior non-WiFi and WiFi activity (e.g., client and traffic properties) using data collected by COAP cloud controller can help in predicting future activity (§3.4). For example, *using 5 days of microwave oven activity* data from APs in an apartment building lead to high predictability of future activity. Also, using client device and traffic type information resulted in *at least* $2\times$ *better accuracy* in predicting wireless traffic properties (e.g., session lengths). Such information can enable the COAP controller to pre-emptively configure nearby home APs to avoid interference.

4 CAPTURING MOBILE APPLICATION EXPERIENCE IN THE WILD

4.1 MOTIVATION

Mobile applications are the dominant means of accessing various services on handheld devices such as smartphones and tablets today. Most of these mobile applications are typically developed by third-party developers for different mobile operating systems such Android [46], iOS [23] and Windows Phone [65]. The success and adoption of an application can be impacted by a large variety of factors, e.g., network latency and bandwidth experienced, responsiveness of the application to user requests, or even how accessible certain UI components are in the various display screens. In this chapter, we focus on developing solutions to understand users' overall *application experience* when they access mobile applications. From a revenue standpoint, it is also important for application developers to understand different fine-grained aspects of their application from its large and diverse user base. Many questions may be of interest to the developers, but they may not have enough expertise to answer them. For example, (i) Do tablet users spend more time than smartphone users? (ii) What parameters are positively correlated with generation of application revenues? (iii) What factors cause the users to spend more (or less) time using the application?

This problem has resulted in the development of several commercial mobile analytics solutions such as Flurry [7], KISSmetrics [9] and many others. These tools provide application developers with business analytics such as number of downloads, user locations, client devices used and crash statistics. Some prior research efforts in this area have focused on controlled user studies [41, 87] to understand client device and application usage patterns. Most of these studies used instrumented smartphone devices. This approach is not feasible for third-party mobile application developers since they do not have device-level access. Over the last few years, a number of interesting standalone measurement systems have also been developed and deployed on mobile platforms, most notably the 3GTest [49] system. 3GTest (available for the iOS, Android, and

the Windows platforms) is a standalone app that can be installed by any user on their mobile device. When this application is activated, it collects network performance measurements for that user at their current location and reports these measurements to a central server. But, this information may not be useful for other application developers since they require measurements during periods when their application is accessed by users.

Understanding the application experience: To empower developers with more knowledge about their applications once deployed in the wild, we developed a toolkit library, called Insight. Insight is unique in that it provides application developers with analytics relevant to their application — it enables them to continuously capture *a configurable set of dynamic behaviors that users experience when they use an application.* Developers can easily embed Insight into their application by importing a library. This "target" application interfaces with Insight using a standard API. In addition to passive statistics collected by commercial analytics toolkits (e.g., usage trends like session lengths, popularity of platforms), Insight also collect information about network quality and application footprint. This information helps developers understand the factors (e.g., network latency, client device) affecting the application usage and revenues. A key aspect of this measurement capability is that it reflects the actual application experience, at the times and the locations where users actually use these applications. This system, therefore, continually crowd-sources data across vast geographical areas and for long periods of time. Further, since the measurements are taken continuously over extended periods of time, it allows us to carry out longitudinal studies and helps us understand the long term trends in usage and performance characteristics.

For us, this approach has provided an intriguing measurement platform that reveals the user's experience across different applications. Over the last three years, Insight has been installed by two different commercial developers with a total install base of more than 1 million users worldwide. Through this, it has served as a distributed lens inside applications to reveal various properties in the continuously changing wide-area wireless and mobile environments.

Usage of Insight by developers: Given the ease of developing and releasing applications on emerging mobile applications today, a vast majority of programmers are unlikely to have enough performance debugging experience and foresight. Insight is targeted towards these developers. When using Insight, these developers simply need to add a small number of lines of code. For instance, to log an in-app event of spending 5 units of virtual currency, a developer simply inserts the following line of code into his application: *Insight.captureEventValue("currency",-5);*. Insight automatically records the required information, securely transmits it to the Insight measurement server and subsequently makes the information available for analysis.

The key contributions of this chapter can be summarized as follows: (i) We design Insight, a framework that allows mobile application developers to collect application analytics and understand the various aspects about the usage of their applications. (ii) Through the long-term deployment of Insight on 2 applications (a MMORPG and an educational app), we study and contrast the properties of these two apps, (iii) We couple application analytics with different factors (e.g., network performance) to gain insight into various factors affecting applications, e.g. usage and revenue generation.

The rest of the chapter is organized as follows. We begin with a detailed description of the Insight framework and data collection methodology in §4.2. We then study the application usage and footprint across devices in §4.3. We correlate network performance with application usage and revenues in §4.4. we discuss the impact of user retention on user activity and application revenues in §4.5.

4.2 THE INSIGHT FRAMEWORK

In this section, we present an overview of the measurement framework and discuss the two commercial mobile applications used in this study on which Insight was deployed.

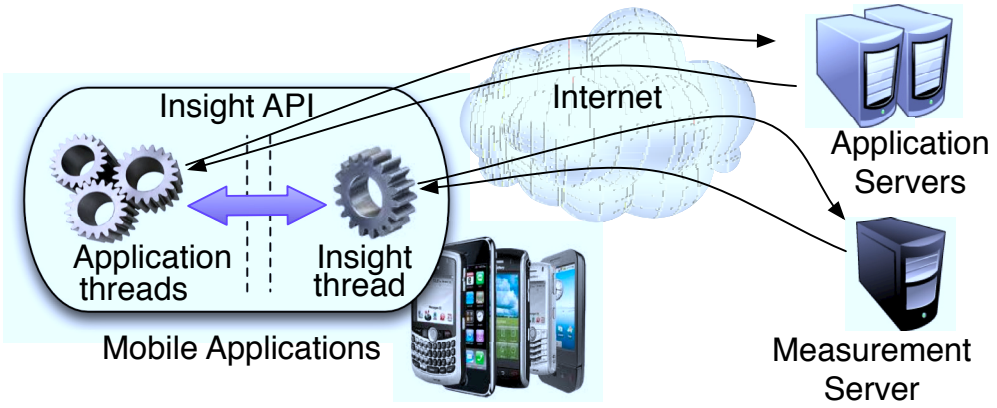

Figure 4.1: Illustration of the Insight measurement framework.

Measurement Framework

With Insight, our goal is to provide a flexible mechanism to collect various statistics from mobile applications. We achieve this goal by providing Insight to application developers as a library (e.g., *insight.jar* for Android) that can be embedded into their applications. Figure 4.1 illustrates this overall measurement framework. The application code interfaces with this library using a standard API exposed by Insight. When a user starts the application on a mobile device, it spawns a separate thread that starts collection of *generic data*. The generic data logged consists of the information about application usage (e.g., session lengths), application footprint across devices (e.g., CPU usage) and information about the network performance (e.g., end-to-end latencies). Additionally, Insight's API also enables developers to log *app-specific data i.e.*, that could be of interest to the application under consideration. For example, a game developer might capture the virtual currency transactions within the application sessions.

While Insight's approach provides many advantages, such a design also imposes several constraints. Working with the mobile application developers helped us realize the following constraints:

- The API should be intuitive and simple to use with minimal instrumentation required inside the application.
- Measurement framework must be *light-weight i.e.*, Insight's processing, and memory requirements must be minimal, and must not affect application performance.
- Network measurements have to be totally "non-intrusive" with no effect on the application's performance. For example, we did not have the luxury to perform network bandwidth measurements using the framework.
- The measurement framework should be flexible. It should allow the developers to easily enable/disable any portion of the measurements (e.g., network measurements).
- Measurement granularity must be flexible, further, we might only be able to collect data at a coarse granularity to reduce measurement overhead. For example, in PK, the collection of location statistics was restricted to 1 measurement every 5 minutes.

Framework details

Insight periodically collects measurements from mobile applications to generate analytics that provide useful information to the application developers (Figure 4.2). It also performs *light weight* active measurements to understand the network performance. An important aspect of the library is that it *only runs when the application is used in the foreground* and does not perform any background data collection. Insight uses a light-weight measurement library, which consists of around 3000 lines of code. The library is written in Objective-C for iOS and Java for the Android SDK. Insight's measurement server implementation consists of around 2500 lines of C# code. We now elaborate on the data currently logged by Insight. Some of these capabilities have been added over time to enhance Insight's view of the application.

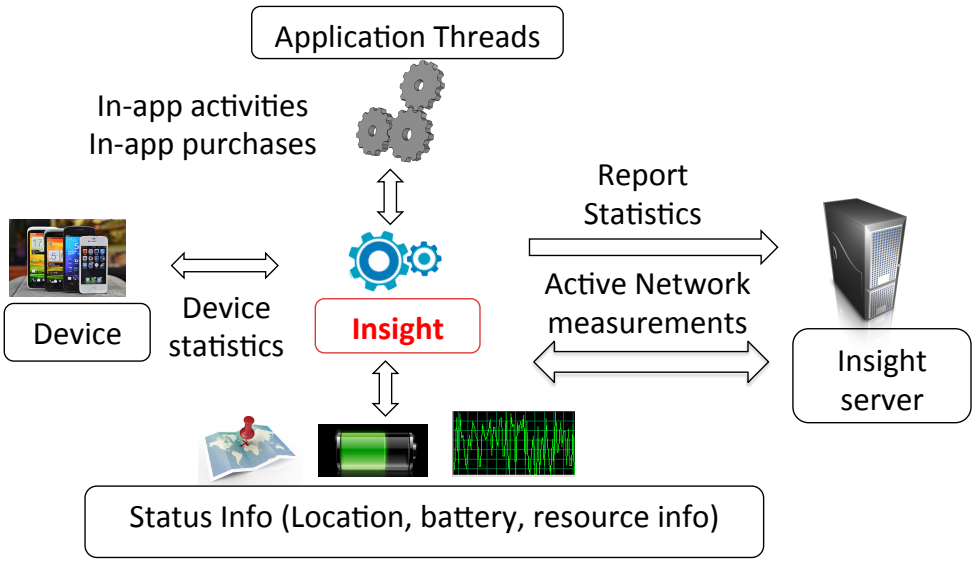

Figure 4.2: Statistics collected by Insight during an application's usage through passive and active measurements.

Metrics	HTC Dream	HTC Evo 4G
Average CPU usage	$< 1.5\%$	$< 1\%$
Average memory usage	< 100 KB	< 100 KB
Average network usage	< 5 KB per min.	< 5 KB per min.

Table 4.1: Insight overhead measured on 2 Android devices showing the minimal resource and network overhead added by the library. Insight runs only when the application is in the foreground.

Insight's measurement overhead. Table 4.1 shows that the processing and memory of Insight is minimal. For example, Insight added < 2% CPU overhead on our test Android devices. Also, Insight's network overhead is small since it uses the network connection to only perform coarse grained network measurements. Further, application developers can selectively enable/disable a subset of Insight's measurements for their applications. We now elaborate on the data logged by Insight.

Generic data logged by Insight. We now describe the generic data logged by the current implementation of Insight. It is important to note that developers have the flexibility to selectively enable or disable any of the following measurements. So, it is upto the developers to determine the set of metrics that they wish to capture.

1. Session Data. The duration of an application's usage in the foreground is denoted as a session. A session ends if the application is closed or pushed to the background.

2. Device Information. Insight also records the device related information (e.g., model, platform, screen brightness, available network interfaces, signal strengths) per session.

3. Network Data. Insight collects periodic network performance data by performing light-weight measurements using both TCP and UDP. When a user starts the application, Insight initiates a TCP connection with our measurement server (co-located with the actual application servers). The server sends probes to the applications at a configurable interval using TCP and UDP packets with increasing sequence numbers. These probes elicit immediate responses from the application allowing computation of both UDP and TCP based network latencies or Round Trip Times (RTTs). These RTTs allow us the measure the quality of the network link (e.g., cellular or WiFi) between the server and user device. A high value of network latency (or RTTs) indicates the presence of a poor network link. We chose a 30 second time-interval as it provides reasonable measurement accuracy while minimally affecting a mobile phone's resources [62].

4. Location Information. While the app has access to precise user location through GPS/WiFi, for our study we anonymize this information into coarse-grained blocks (100 sq. km. regions, state level, or country level). We use reverse geocoding to determine the user's country and region information from the coarse-grained location.

5. Application footprint on device. Insight also measures the resource overhead of the application on different mobile device models by capturing a set of metrics. These metrics include the CPU and memory utilization by the application as well as the "battery drain rate" on the device while the application is in the foreground. These metrics are obtained by using information exposed by the

platform (e.g., /proc filesystem and APIs exposed in Android SDK). The battery drain rate is calculated per session (while the application is in foreground) by dividing the change in battery level during a session with its duration. For example, if the battery level changes from 80% to 75% during a session length of 20 minutes, the battery drain rate is (80 -75)/ 20 levels/minute, i.e., 0.25 levels/minute. To make this analysis more accurate, we use some filtering as described in §4.3. The overhead of this measurement is minimal on the device as well as the application as it is passive and performed infrequently (every 30 seconds by default). Currently, we have implemented this capability only for Android.

App-specific data logged by Insight. In addition to the above generic data, Insight also provides an API to the applications to log any additional data. The application code calls this API periodically to record app-specific statistics. This data is later combined with the other statistics (e.g., device, network performance) collected by Insight to study their correlation with the in-application activity.

Test applications

We have deployed Insight within two third party applications which are available for free in Apple's AppStore (iOS platform), and in Google's Android Market (Android platform). Figure 4.3 shows a screenshot of these applications.

Application #1 (MMORPG): Parallel Kingdom (PK). Parallel Kingdom [77] is a Massively Multiplayer Online Role Playing Game (MMORPG) that places the character (user in the virtual gaming world) on the game map according to their real world location. The game uses GPS/WiFi capabilities of the mobile device for tracking the real world user location. In the game, each player picks up, trades or upgrades various items, fights monsters and interacts with other players on the map or through a chat facility. As the game progresses, a player learns various skills, claims territories and obtains unique capabilities. Players can spend real money to buy the virtual currency in the game ("Food"). The game has added numerous features over time through the releases of

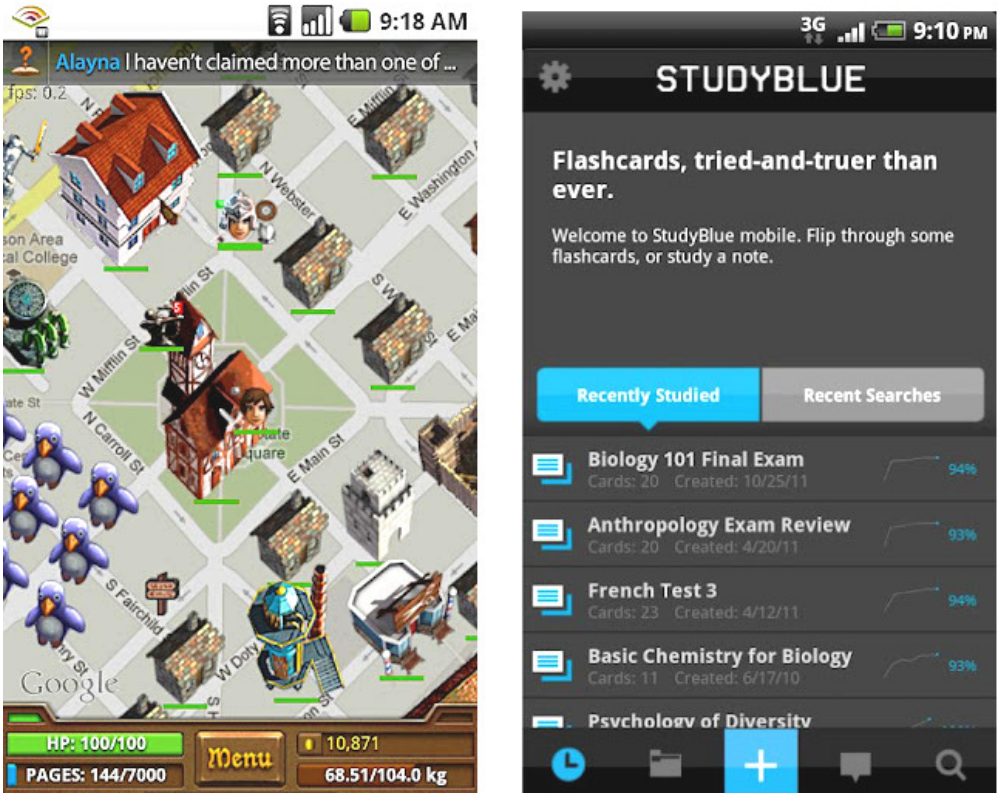

Figure 4.3: Screenshots of Parallel Kingdom (left) and StudyBlue (right), the third party applications that used the Insight framework.

new "Ages" and minor updates. The game (starting with Age 1) was officially launched on October 31, 2008 and has since then become a popular game on both the Android and iOS platforms. Since the First Age, there have been many other minor releases and three major releases (Age 2, Age 3 and Age 4). The game has over 1 million players worldwide. It was ranked amongst the top 4 Android applications in its category in Dec 2009 [1] and was a finalist for the 2011 Mashable awards in the best mobile game category [2].

PK specific data logged by Insight: In addition to the generic data, PK uses Insight to log the following:

1. *Expenditure of Money.* "Food" is a special resource in the game, which

	PK (game)	SB (education)
Unique users	$> 1,000,000$	$> 160,000$
Sessions	> 61 million	> 1.1 million
Countries	> 120	> 25
RTT measurements	> 350 million	> 25 million

Table 4.2: Application stats tracked using the Insight framework. The data collection process for PK started on 31 October 2008 while it started for SB on 29 April 2012.

serves as the virtual currency. Players spend (or burn) Food to buy items from merchants, upgrade buildings etc. Players can purchase food with real money to expedite progress in the game. Thus, tracking food consumption is important for PK developers as it is their main source of game revenues.
2. Action data. Game related activities such as gathering resources, fighting other players and monsters, trading items were also logged for each session.

Application #2 (Study Tool): StudyBlue (SB). StudyBlue [93] is a smartphone/handheld device based digital study tool for students. SB acts as a platform and enables students to create, study and share flash cards. SB also provides features such as online quizzes, study guides etc. We integrated and launched Insight with SB's Android application on 29th April 2012. Currently, SB has thousands of daily active users who are mostly based in the US ($> 90\%$).

SB specific data logged by Insight: SB uses Insight to log the following application specific events:
1. Study Events. Study events include activities within the application such as taking quizzes, reviewing study material etc. They are related to consumption of content by users.
2. Edit Events. They include events related to content creation within the app. Edit events include creating new flash cards, updating/editing existing "decks" of flash cards. This activity is important for SB developers, since higher content creation makes their platform more valuable to other users.

Analyzed datasets

Using Insight, we have collected data for PK over a period of more than 3 years during which the game received 3 major upgrades and many minor updates. Compared to this data set, the data from the StudyBlue application (Android only) is smaller in size, for just over a period of 1 year. We use these two datasets to provide some application specific comparisons between an MMORPG and an educational app. Table 4.2 presents a summary of the tracked statistics.

Privacy implications

To conduct this study, we had a special arrangement with our partnering software companies. In each case, we developed and provided the Insight library to the companies, and they integrated our software into their application. All collected data was always resident in the servers of the individual companies. Through a special NDA in each case, we were provided with an anonymized feed of the various data we collected. Among data that we present in this chapter, the only potentially sensitive data is user location. For the purpose of this project, we anonymized this information by mapping them to a coarse-grained block — a 100 sq. km. area, state level, or country level, as was adequate for the analysis presented in this chapter. An accurate latitude-longitude level accuracy was not necessary. Further, our research team never directly interfaced with the users or their data, and instead were limited by special arrangements we negotiated with the two companies.

4.3 USING INSIGHT FOR APPLICATION ANALYTICS

In this section, we study some trends related to the usage of the PK and SB apps as well as the impact of device-related factors on the usage of both applications. The observations presented in this section show how Insight helps developers understand various aspects of their application by capturing a set of key metrics.

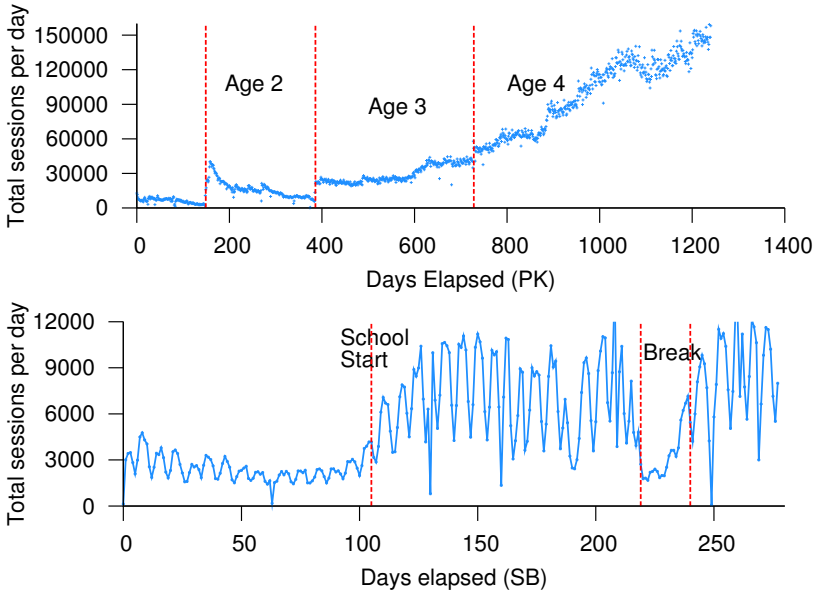

Figure 4.4: Number of sessions/day for PK (MMORPG game) and SB (study tool).

Analyzing user population

We now study the properties of the user base and the application access patterns across the two applications.

What are the long term trends in daily active usage?

Figure 4.4 shows the number of daily sessions for PK (top) and SB (bottom) since the start of data collection for both these applications. The graphs shows different trends for sessions per day across the two applications. Both applications observe thousands of sessions per day although it is much higher of PK due to its wider user base.

For PK, the number is either steadily increasing or decreasing for most of the time except for a few days when there are sudden spikes. The sudden spike in days 149, 387, 729 correspond to the major releases of the game (called Age 2, Age 3 and Age 4 respectively). Other big spikes in the number of daily active users occurred around June 28 2010 (day 603) when the game was released

on the iPad and iOS 4 platform and on day 886 due to a major update to the game. The release of new releases and features positively impacted the number of daily active users of the game. Also, starting from Age 3, there is more consistent increase in the game's daily active usage. Some important factors causing this behavior are – more frequent improvements in gameplay through updates, increase in the game's popularity in its category over time and increase in developers' efforts to advertise the game to attract new players.

For SB, the number of daily sessions is more tightly correlated with the time of week as well as the time of year. For example, the number of daily sessions for the application spike during the weekdays (Monday - Thursday) and the troughs correspond to Fridays and Saturdays of each week. Also, the number of daily sessions for the application was much lower during the summer (days 20 -104) and winter (days 224 - 240) breaks. For SB, this behavior happens due to the nature (study tool) and specific demographic user base (students) of the application. This behavior is not observed for PK due to its more diverse user-base and generic nature of the application (game). *Application such as SB, with highly variable but predictable usage trends can reduce costs by dynamically provisioning cloud-based servers based on the time of the day/week.* For example, the cost per Amazon EC2 cloud instance [4] halves by going down each level (e.g., from "Medium" to "Small" instance).

When do the users access these applications? How is this behavior different across applications?

Figure 4.5 (left) and Figure 4.5 (right) show the number of online based players (from US Central time zone) in a 24 hour period for PK and SB respectively. It shows that the peak number of online users occur at different times for the two applications. For example, We find that the number of PK players reaches a maximum during late nights. But the number of online SB users peak between 1 -2 PM and then falls during the evening before rising again between 10 PM - 12 AM. The most likely reason for this behavior is that users prefer to use these applications at different times of the day. For example, SB's peak usage is in the afternoon and nights when students study more. The

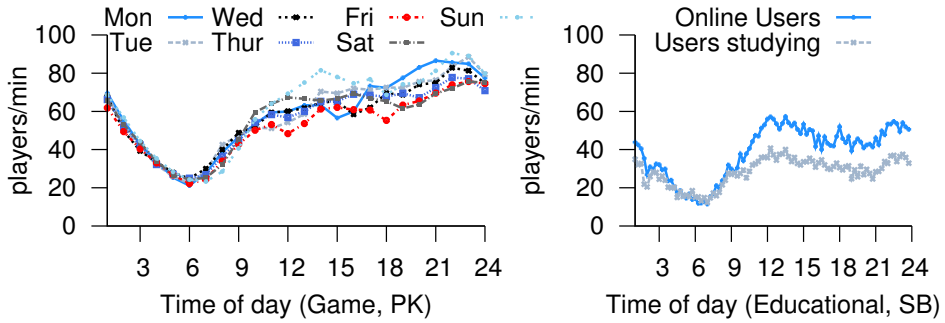

Figure 4.5: Average number of online players per minute (CST) across 24-hours for PK (left) and SB (right).

graph also shows the number of actual users actually studying while using the application. Almost all online users at early mornings (3 - 6 AM) appear to be burning the midnight oil for studying! At other times, its users also perform other activities such as sending messages, creating flash cards, searching data etc. We observe PK gets used the most at nights (9 PM - 11 PM). Further, for PK we also find higher population on Sundays, and a lower population on Fridays.

We find that for both the applications, the number of online users are at a minimum around 6 AM in the morning. This diurnal pattern was consistent across different countries. Application developers can use this insight for several purposes. For example, early mornings (around 6 AM) are favorable to schedule server downtimes and upgrades, install bug fixes or push updates, as they have smallest number of online users across the whole day. For applications such as SB, the time of week information can also be used to schedule such server upgrades and downtimes due to its strong correlation with application usage.

Summary. Across the two applications (PK and SB), we observed differences in the longitudinal and diurnal usage due to their different nature and user demographics. Such application specific behavior can be leveraged by application developers in optimizing costs related to their server deployments as well as creating incentives (e.g, through updates) to attract new users and retain the existing ones.

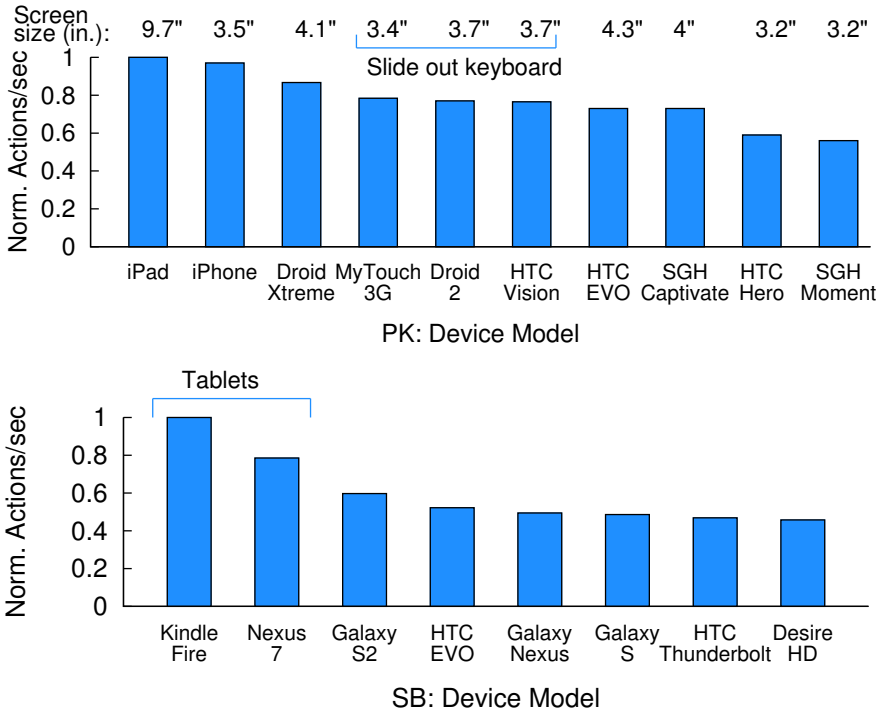

Figure 4.6: Normalized number of actions per unit time (maximum of 1) for different tablets phone models for PK (top) and SB (bottom). Usability in terms device type, screen size and slide out keyboards are shown.

Understanding application usage

We now study how different context (e.g., device, time of day etc.) impacted the application access properties across PK (game) and SB (study tool) as well as the differences in the session properties across these different types of applications.

How does device usability (platform, display screen size, availability of slide out keyboards) affect user interaction?

Both applications (PK and SB) are used across a wide variety of smartphone and tablet devices and employ very similar UI for both the tablet and

smartphone platforms. We analyzed how "usability" of different platforms may affect the application usage. In particular, we analyzed the effect of (i) size of the display screen and (ii) availability of slide out QWERTY keyboards on "user interactivity". We analyzed the effect on "user interaction" by measuring the average number of user actions/sec on popular device models over a period of one week. For both applications, we chose the top smartphone and tablet devices at the time having diverse form factors (screen sizes and slide out keyboards).

Figure 4.6 shows some interesting results. We find that tablet device (iPad, Kindle Fire, Nexus 7) users performed the highest number of actions/sec, on both applications owing to a larger screen size (e.g., 9.7" for iPad and 7" for both Kindle Fire and Nexus 7) and both applications involve a large number of touch events. For example, in PK users perform touch related activities such as exploring the map, moving the character and attacking monsters. Players can visualize a larger area on the map on tablets compared to smartphones and interact with more objects simultaneously on the map. On SB, users take quizzes which mainly involve touch events. For PK, we also find that the iPhone comes a close second (despite a screen size of 3.5"), indicating the impact of device and user interface differences between iOS and Android.

Amongst Android device models using PK, we find an interesting trend — devices with smaller screen sizes (e.g., 3.2" for Samsung Moment) experienced lower user interactions compared to those with a larger screen size (e.g., 4.3" for HTC EVO). However, devices with slide out keyboards (e.g., Droid 2) exhibit higher user interaction compared to some of the devices with larger screen size (e.g., EVO), despite small screen sizes (3.4" – 3.7"). These Android devices have similar capabilities in terms of CPU and memory. This is due to higher usage of keyboard based activities (e.g., in-game chatting/messaging) on these devices. This shows that a device's form-factor and ease of use can impact application usage behavior. *Overall, the average user interactions varied by around 2× across devices for both applications.* Tracking such user interactivity patterns across different devices in the wild can help developers understand the combined impact of the UI and device on the usage of different features within application (e.g., impact of keyboard based devices on text-based activities). This is a useful

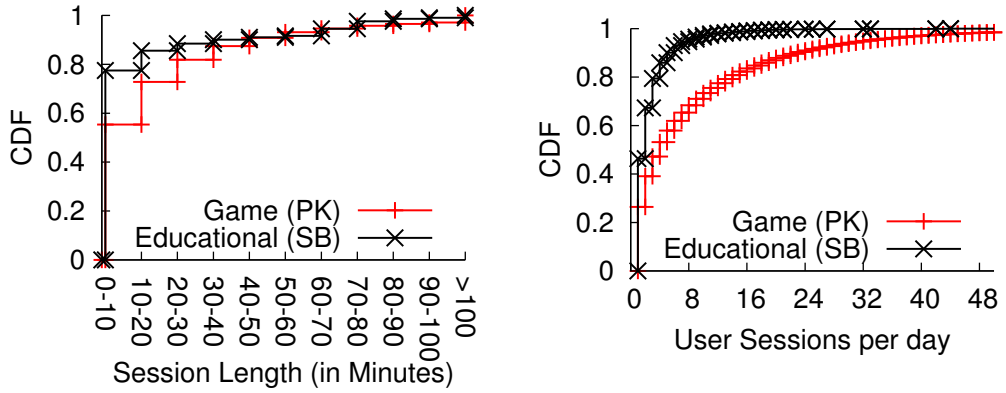

Figure 4.7: (left) Distribution of session lengths, (right) CDF of the number of sessions played by a user per day.

input for developers to make UI/feature modifications (e.g., layouts, button sizes etc).

How did the session properties differ across PK and SB?

Figure 4.7 (left) shows the duration of the session lengths in the game. We find that a large fraction of sessions on both applications are short lived. For example, 55% and 76% sessions are less than 10 minutes long on PK and SB respectively. Further, only 9% of sessions are longer than 60 mins on PK and SB. Figure 4.7 (right) shows the CDF of the number of daily sessions per user. Around 26% of PK users play a single session per day while 27% of its users play more than 10 sessions. SB has fewer user sessions per day with 46% of the users accessing having a single session per day and only 5% of the players accessing more than 10 sessions per day. MMORPGs such as PK tend to have more sessions per day as it common for players to access the game multiple times per day. For example, it is common for PK users to login, play for a short while (e.g., feed the dogs) and logoff.

Figure 4.8 shows the distribution of the time gaps between consecutive user game sessions per day (the difference between start time of a session and end time of the previous session for the same user). *It is interesting to note that 47% of new PK user sessions tend to be within 5 minutes of the end of the previous session.*

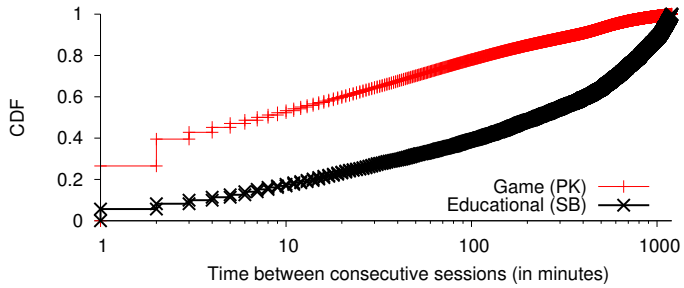

Figure 4.8: Distribution of time periods between consecutive user sessions per day. The X-axis is shown in logscale.

In case of SB, only 14% of the new user sessions are within 5 minutes of the previous one.

These short and closely spaced multiple sessions are partly characteristic of usage of smartphones as has been observed before [41] and is observed for both PK and SB. It is normal user behavior to get distracted [58] and close the current application to use another application such as an email client or stop the application to attend a phone call and then return back to the game[1]. The higher proportion of short gaps between consecutive sessions for PK compared to SB (47% vs 14%) is partly game specific as observed in desktop based MMORPGs [33]. Application developers can implement optimizations based on such user behavior. For example, in the case of PK, instead of discarding a player's state (maps, locations, inventory information etc.) from the phone at the end of a session, it is more efficient to save and reuse it during the next session because almost half of the new user sessions occur within 5 minutes of the previous session. Over time, PK's developers have *improved such caching capabilities to decrease network communication* with the server at the start of each session as well as *improve the start-up time*.

What was the impact of factors that can force users to restart a game session?

[1]Since Insight is active only while the application is in foreground, we did not use it to log the activities on the phone in between the application sessions to find the actual cause for each gap.

Characteristic	% of sessions
High CPU or memory usage	$< 1\%$
High network latency or RTT ($> 1sec$)	7.8%
Network type change (WiFi, cellular)	1.9%

Table 4.3: Observed characteristics for PK sessions that can cause the user to stop and restart the game. This analysis is only for PK sessions which have another user session succeeding it within 5 minutes. A session can fall into multiple categories.

Figure 4.8 showed that 47% of new PK user sessions were within 5 minutes of the end of the previous session. Some of these cases could have been caused due to pathological cases (bad network, high CPU utilization etc.) or availability of another (possibly better) cellular or WiFi network. Such cases can *force* a user to close his previous session and start a new session. Using Insight, developers can keep track of the percentage of sessions exhibiting such pathological behavior and can take actions if such cases become prevalent.

Table 4.3 shows some of the potential causes that may have caused the user to restart the game and the percentage of sessions showing a particular cause. Around 1% of the sessions had high memory or CPU usage (90-100% average CPU utilization or memory leaks characterized by abnormally high memory usage). 7.8% of the sessions had a high average network latency or RTT ($> 1sec$, §4.2). In 1.9% of the cases, the user changed the network (e.g., changed the WiFi network or changed the connection from cellular to WiFi or vice versa). This breaks the TCP connection with the server and restarts the session. Overall, a small percentage of these sessions were impacted by potential causes that can disrupt a session. This indicates the majority of these consecutive sessions can be attributed to user behavior as discussed earlier. In §4.3 and §4.4, we further discuss how high battery drain and poor network performance, respectively, impacted user experience and session lengths.

How did the time of day context impact the application usage behavior (e.g., network type, battery usage etc.)?
Users' application usage behavior (e.g., choosing the network type) can also

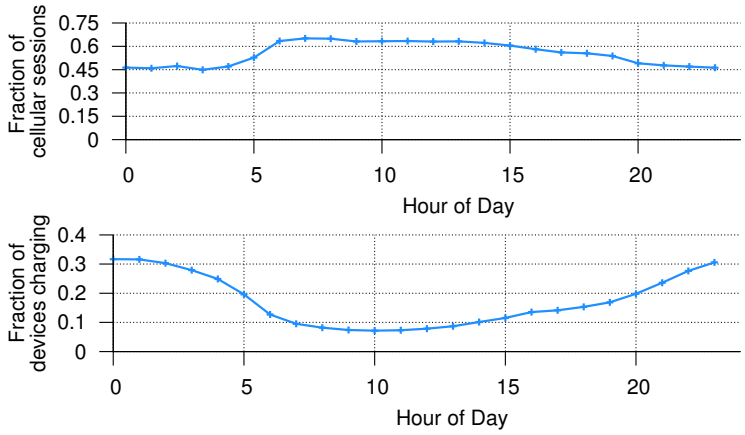

Figure 4.9: For the US based users, the graph shows the fraction of total users using the cellular network (top) and charging their device (bottom) while using the application over the day (normalized to CST).

be impacted by the context (independent of the application or device), such the as the location of the user, time of day etc. We analyzed how the "time of day" context impacted the user behavior for 2 different properties: the type of network used and fraction of active users charging their device during application usage (Figure 4.9). The figure shows that a *larger fraction of US based users accessed the applications over cellular networks during the day compared to late nights* (e.g., 45% at midnight and 63% at 10am). Further, 31% of the users accessed the applications while charging the device during late nights compared to only 7% of the users around noon. Due to these diurnal properties, using the time-of-day context can also be beneficial in the use of network and battery [103] related optimizations by mobile applications.

Summary. We observed a 2× variability in user interactivity across different devices for both applications. Such analytics is important for understanding the impact of device and the application UI on user behavior. Further, high fragmentation of mobile devices in terms of OS, hardware and device factors makes it challenging for application developers to deliver a consistent user experience across these devices. For example, developers of popular

applications [40] need to test their applications extensively across several models before releasing new updates. Tools that assist developers to debug issues related to device fragmentation can be very helpful.

We observed short but multiple sessions for both PK and SB users which can partly be attributed to smartphone usage behavior. In the case PK, the occurrences of closely spaced consecutive sessions led to efforts by developers to reduce the application start-up time for improving the user experience. We also observed a diurnal variation related to network (1.5 $\times$)and battery usage (4 $\times$) in our user population, indicating the utility of the time-of-day context information in optimizing network and battery usage by mobile devices and applications.

Measuring application footprint

Once a mobile application gets deployed in the wild, it is very difficult to track its performance on the diverse types of mobile devices available [20]. We used Insight to monitor the application's overhead in terms of battery usage, CPU consumption and memory usage (§4.2) while it was in operation. Knowledge about application footprint across different device types can allow developers to understand which device types to target and optimize their application for. To make the battery drain analysis more accurate, we filtered the data to remove sessions that had multiple running processes (e.g., background downloads) while the game was in operation. This was done to make sure that the battery consumption reflected closely its usage by the application. We also removed those sessions from this analysis during which the user was charging the phone for any portion of the session.

How does the battery consumption due to the game (PK) vary across different devices?

Figure 4.10 shows the distribution of normalized battery drain rates (levels/min) measured on the popular devices used by PK players. We make the following observations: (i) Across different devices, the variation in drain rates is caused due to use of different components (e.g., screen, network card) even if they have the same battery capacity. For example, the devices LG Optimus,

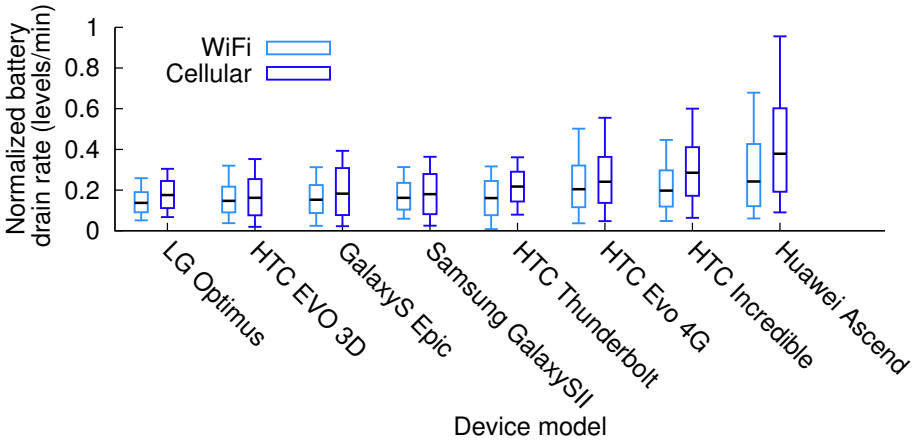

Figure 4.10: Overall normalized battery drain rates (w.r.t. max. drain rate) for popular Android devices over cellular and WiFi (PK). The candlesticks show the 10th, 25th, 50th, 75th and 90th percentiles.

Samsung GalaxyS Epic and HTC Evo 4G have similar memory, CPU, network capabilities and the same battery capacity of 1500 mAh but have different drain rates. Amongst the major causes are different screen sizes and technologies: 3.2"(TFT) for Optimus, 4"(AMOLED) for GalaxyS Epic and 4.3"(TFT) for Evo 4G. (ii) On each device, the variation in drain rates is due to factors such as: different screen brightness, battery age (old vs new battery), variable cellular or WiFi network usage etc. (iii) The battery drain rates are higher on cellular sessions vs WiFi sessions. This is because the cellular connection consumes more energy per bit compared to WiFi [25].

These *diverse resource consumption characteristics across different models (more than 3×)* points to the difficulty by application developers in optimizing the usage of the phone's resources on different device types due to the high diversity [40]. Insight can help developers to track this behavior and identify potential performance issues across devices in the wild. Further, device specific optimizations (e.g., reducing GPS usage or automatically reducing the screen brightness on phones with lower battery levels) can considerably improve the energy efficiency of mobile applications across different devices. But, the gains

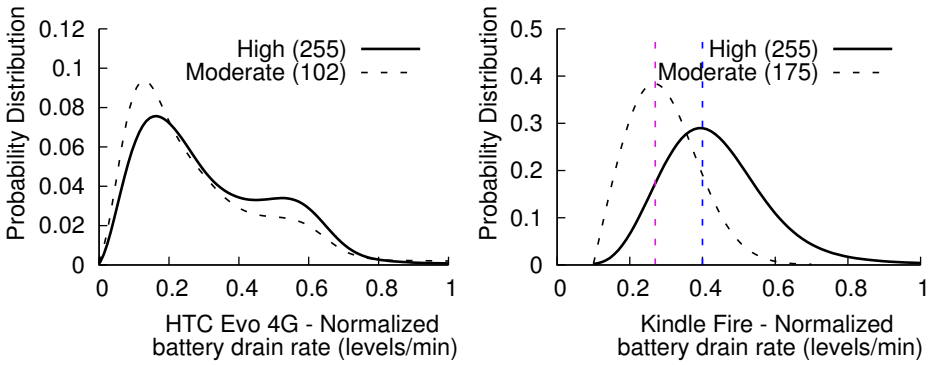

Figure 4.11: Density plot showing the normalized battery drain rates (w.r.t. max. drain rate) at two brightness levels for HTC Evo 4G (left) and Kindle Fire (left) while running SB in the foreground.

from such optimizations can vary from device by device basis as discussed next.

Figure 4.11 shows the distribution of battery drain rates observed at two different screen brightness levels (high, moderate) aggregated across thousands of SB sessions (application in foreground) on HTC Evo 4G and Kindle Fire devices. For the Evo device, the battery drain distributions are similar at both brightness levels. It indicates that controlling the screen brightness on this device may not provide much gains in overall battery savings. This is because factors other than screen brightness that have a higher impact on the overall battery drain. *But, for the Kindle Fire device, the mode of the battery drain distribution goes down by 40% (0.4 to 0.27) by reducing the screen brightness from "High" (255) to "Moderate" (175).* Thus, for the Kindle Fire device, controlling the screen brightness through the application (e.g., by using Android APIs for brightness control) can be an effective way to reduce the battery drain when the battery levels are low. Power-saving techniques such as [99] can also be used to dynamically reduce the brightness of specific screen areas that are not important for the user. We now discuss how battery drain impacted the session lengths for the applications.

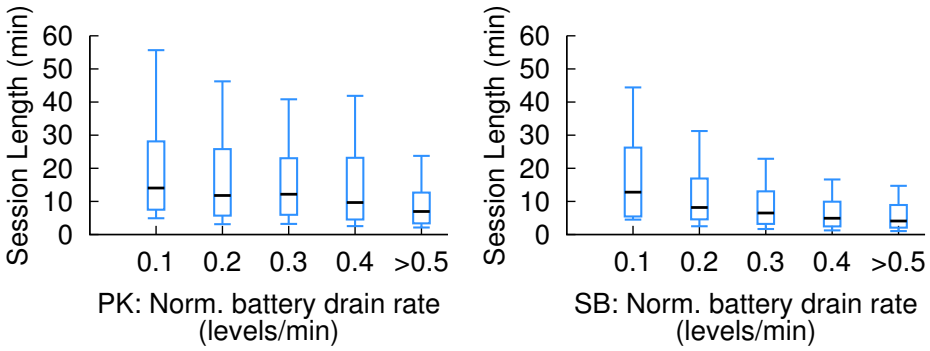

Figure 4.12: Distribution of session lengths as a function of the normalized battery drain rate (bins of 0.1) for cellular based users on HTC Evo 4G for PK (left) and SB (right).

How were session lengths impacted due to the battery drain caused by the application?

High battery drain caused by an application can cause users to reduce the session lengths for conserving the battery. To understand the impact of the battery drain rate caused by the application on the player behavior, Figure 4.12 shows the distribution of session lengths for HTC Evo 4G (a popular device amongst PK and SB users) as function of the normalized battery drain rates. For both applications, *the session lengths decreased with the increase in battery drain (more pronounced for longer sessions)*. For example the 75th percentile session lengths for PK are 29 minutes and 12 minutes for normalized battery drain rates (levels/min) of 0.1 and > 0.5 respectively. The same number goes down from 26 minutes to 9 minutes for SB sessions. The variation in median session lengths is much smaller as the battery drain rates increase, indicating that players with shorter session lengths are less sensitive to the battery drain.

How did user interactivity in PK impact resource utilization on the phones?

Amongst the two applications using Insight, PK (game) was a more resource intensive application due to the higher computational and network requirements. Figure 4.13 shows how increasing user interactivity causes the PK's CPU usage to increase. For example, user actions such as moving the character causes the game to render the screen again for the new location.

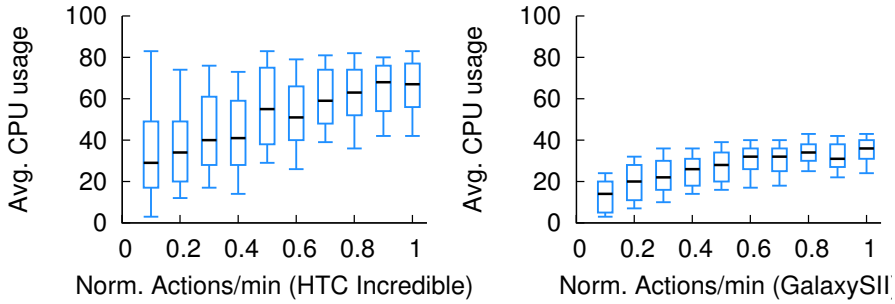

Figure 4.13: The CPU utilization due to PK (game) across two different devices as a function of the normalized rate of actions performed per minute (bins of 0.1).

Higher CPU usage increases the battery usage on the device. The game's median CPU usage goes up from 29% to 67% and 14% to 36% for HTC Incredible and Samsung GalaxySII respectively, when the normalized actions/min go up from 0.1 to 1. Thus, the graph shows that the resource consumption for similar workloads (actions/min.) varies across different phones due to differences in hardware capabilities.

Thus, tracking resource consumption in the wild for resource intensive mobile applications (e.g., games) is important for developers to understand the application's footprint across all client devices. It can also be helpful to identify pathological issues such as excessive CPU utilization, memory leaks etc. that have the potential to cause poor user experience (e.g., due to unresponsiveness of the application in these cases).

Summary. Factors such as a network type, user behavior and device type cause widely varying battery usage across different application sessions. We also observed a drop in average session duration with increasing battery consumption for PK, which is a more resource intensive gaming application. Thus, techniques to optimize battery usage by applications can be beneficial. As shown in this section, the benefits of specific techniques to reduce total battery usage (e.g., controlling screen brightness) can vary across different device types.

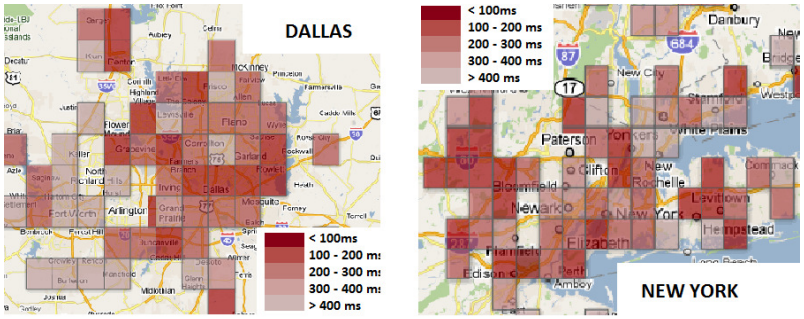

Figure 4.14: Snapshots of median cellular network latencies (RTTs) from two metropolitan regions (Dallas, New York). The RTTs were aggregated into 10km * 10km sized cells and median RTT is shown from each cell.

Thus, a combination of different techniques is required to optimize battery usage across mobile devices.

4.4 IMPACT OF NETWORK PERFORMANCE

Amongst the two applications, PK (an MMORPG) uses network connection (cellular or WiFi) more heavily for real-time communication with the game server. For example, an attack action on a monster needs to be pushed to the server and a response needs to be received by the client. On the other hand, SB requires more intermittent connectivity to the server to download content (e.g., flash cards) or synchronize local updates with the server. As discussed in framework details (§4.2), Insight uses network latency or Round Trip Times (RTTs) as an indicator of network performance. Poor network performance due to causes such as an ISP's network coverage, slower cellular networks (e.g., EDGE, 1xRTT) [80] or poor signal strengths at a user location's can increase the network latencies (RTTs) for a user.

To understand how network performance impacted application usage and revenues, the discussion in this section leverages the network data collected from the apps over a diverse range of networks, devices and locations. Figure 4.14 shows the spatial distribution of cellular network latencies (median per cell, cell size was 10km $\times$ 10km) in two metropolitan locations (Dallas and

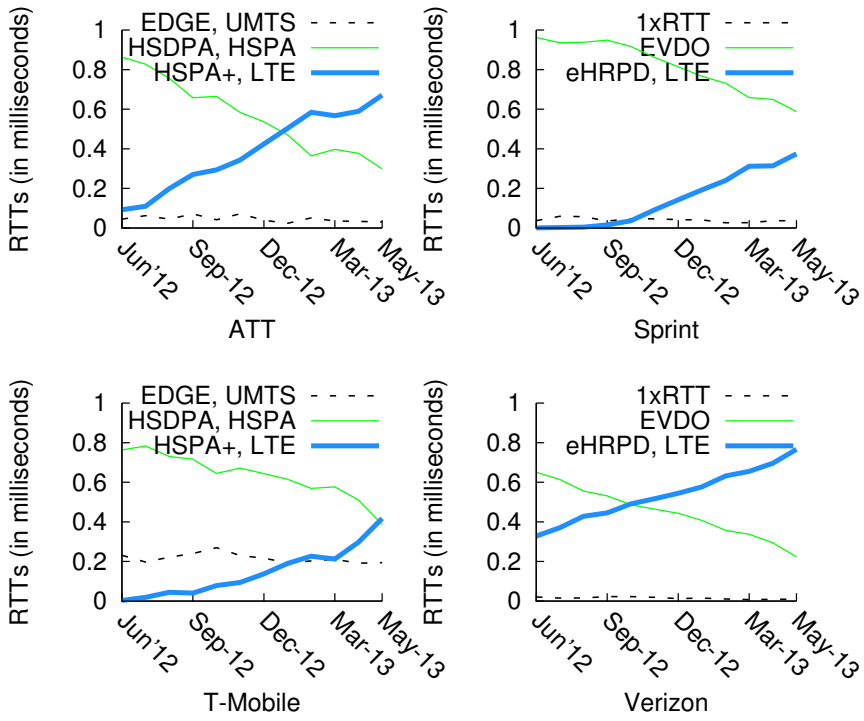

Figure 4.15: Distribution of network traffic from different technology types for cellular carriers from June 2012 to May 2013.

New York) averaged over a month. It shows a considerable difference in cellular network performance within the same metropolitan region. For example, even within New York, we observe that the median RTT can vary from 187 ms to 312 ms indicating uneven cellular performance within a small geographic location.

Usage of latest cellular and WiFi technologies over time. Figure 4.15 shows the distribution of network measurements coming from different technologies (e.g., EV-DO, HSDPA) for the major US cellular carriers over a period of 12 months (June 2012 - May 2013). Over this period, the proportion of network traffic from faster networks (e.g., LTE, HSPA+ eHRPD) is increasing for users across all ISPs. For example, amongst the Sprint users analyzed through Insight, the usage of newer cellular technologies (eHRPD and LTE) increased to 40% by

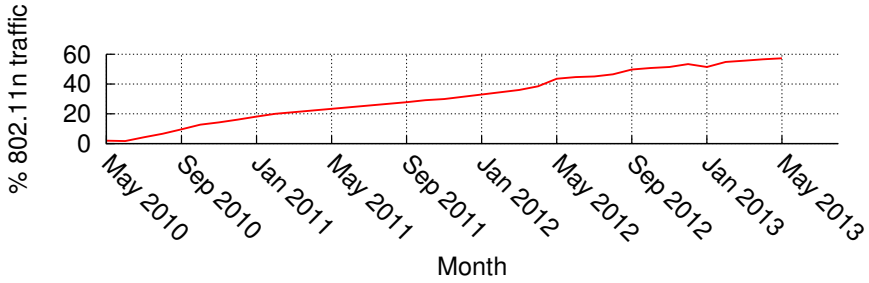

Figure 4.16: 802.11n usage over time from Android based devices.

May 2013. But, there was a small and consistent fraction of users across different carriers that accessed the PK and SB over higher latency cellular connections (e.g., EDGE). Such users might be more sensitive to network performance even during un-congested network conditions. In the next section, we study the impact of network latencies on user behavior across PK and SB applications.

Using Insight, we were also able to track the adoption of 802.11n over time. Figure 4.16 shows that from May 2010 to May 2013, the proportion of WiFi traffic using 802.11n capable devices increased from 2% to 57.2%. This indicates that adoption of newer WiFi standards such as 802.11n is increasing gradually over time with a large proportion of WiFi deployments and mobile devices (around 42.8%) still using the older 802.11bg standards. Thus, mobile application developers should be aware of the presence of these older devices and networks, even though newer WiFi standards such as 802.11ac are getting introduced.

We were also able to track other technology trends amongst the user base (e.g., trends in processing capabilities, available interfaces, memory improvements of smartphones over time).

Impact of network performance on user behavior

We now discuss how Insight can help understand whether network conditions had an impact on application usage.

How were user interactions impacted by poor network performance?

Figure 4.17 (top) shows the normalized number of user actions per minute (max. of 1) as a function of the average network latency experienced during a session. We find that user interactivity (actions/minutes) declined with an increase in network latencies for both applications. But, the decline in user interactions is much more pronounced for PK. *For example, an increase of network latency (RTTs) from 300ms to 900ms caused average user interactivity to drop by 40%.* The same value for SB was only 7% indicating the much lower impact of the network latency on its users. But, at higher network latencies (>= 1.4 seconds) the impact increased on SB users as indicated by a 20% drop in interactivity.

As mentioned earlier, network performance plays a crucial role in MMORPGs like PK because a typical user session comprises many real-time interactions with other users as well as the game server (e.g., moving on a map, attacking monsters etc.). This value may vary for other real-time games or applications, but the results highlight how user perception of poor network performance can vary based on the nature of the application.

How do network latencies influence user's choice of networks (cellular vs. WiFi)?

WiFi based network latencies tend to be atleast a factor of two lower compared to cellular network latencies at most locations [92]. Also, cellular latencies can vary widely based on the type of cellular technologies used (e.g., LTE, EDGE, HSDPA etc.) [80]. We had earlier discussed the reduction in user interactivity during periods of high network latency (more prominent for PK). User perception of higher network latencies (e.g., higher server response times for an action) can be frustrating and degrade their experience. We used Insight network measurements to study the impact of cellular network latencies on their choice of network type (cellular vs. WiFi) to access the two applications.

Figure 4.17 (bottom) shows the average proportion of time spent by PK and SB users on cellular networks while using these apps as a function of their average cellular network latencies. We find that as cellular network latencies increase, users on both applications spend more time accessing the game through WiFi networks (higher for PK users). For example, PK and SB users who experienced low network latencies (0.3 seconds) spent around 55 -

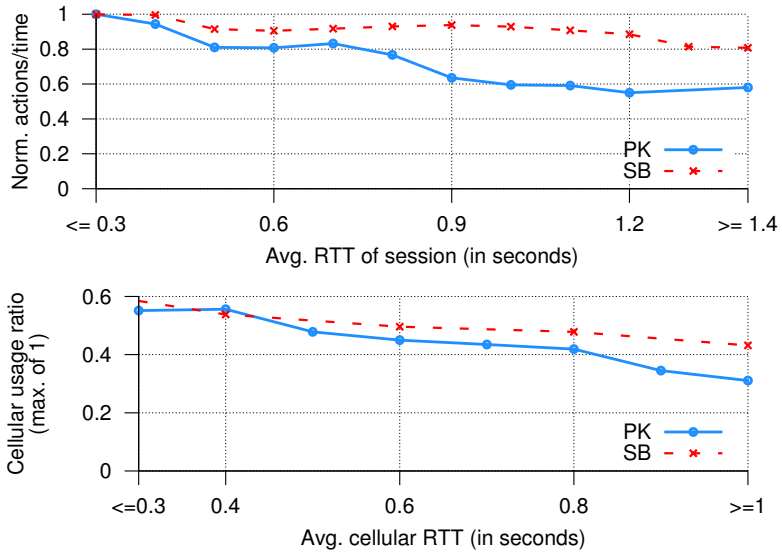

Figure 4.17: (top) Normalized values for actions performed per unit time at different RTTs for US based user of PK and SB (Bin size of 100 ms). (bottom) Percentage time spent on cellular networks (vs. WiFi networks) by US based PK and SB users as a function of average cellular RTTs.

	US	**UK**	**GER**	**JPN**	**CAN**
Cellular	52.7%	24.7%	31.2%	30.5%	23.1%
WiFi	47.3%	75.3%	68.8%	69.5%	66.9%

Table 4.4: Fraction of player sessions accessing PK through cellular and WiFi networks for the top 5 countries.

58% of their game time on cellular networks. *Amongst users that experienced poor cellular network performance (cellular latencies* $>$ 1.0 *sec.), PK users spent only 31% of their time on cellular networks* as opposed to 42% of SB users. This observation indicates that the application's nature (e.g., an MMORPG game) were amongst the important factors that influenced the users' choice of networks used while using their application.

Table 4.4 shows the distribution of user sessions accessing PK using cellular and WiFi networks amongst top five countries (in terms of total user sessions).

Except for USA, the distribution is uneven across other 5 countries with more players accessing the game through WiFi. While this behavior can be affected by many regional factors (e.g. cost of cellular plans, coverage of 3G networks), the observations from Figure 4.17 (bottom) shows that network performance is amongst the important factors affecting this behavior. This is due to the location of PK game servers within the U.S., which leads to higher network latencies for users from other countries.

The analysis in this section showed that across these two applications using Insight, *PK users exhibited higher sensitivity to higher network latencies as exhibited by lower user interactivity and greater preference for WiFi networks under poor network performance*. While developers may have an estimate of the impact of such network related factors on users' experience, its difficult for them to quantify the impact of such factors in the wild. Crowd-sourcing or sampling such information from users during their sessions can help expose these application-specific dependencies between network performance and user experience. Such applications or games with multiple server locations can benefit from frameworks [62] that use periodic estimation of network latencies within the application to optimize the selection of servers and other players to reduce network latencies experienced by users. For developers with more limited resources, these measurements can guide the placement of additional servers (e.g., choosing Amazon EC2 instance locations) near regions that experience high network latencies.

Summary. An increase in network latencies impacted user behavior, especially for PK (an MMORPG), as indicated by a decrease in user interactivity (by 40%) and lower usage of higher latency cellular network connections. Such applications can benefit from frameworks that use client assisted network measurements to optimize the placement of cloud based servers to reduce user-perception of high network latencies.

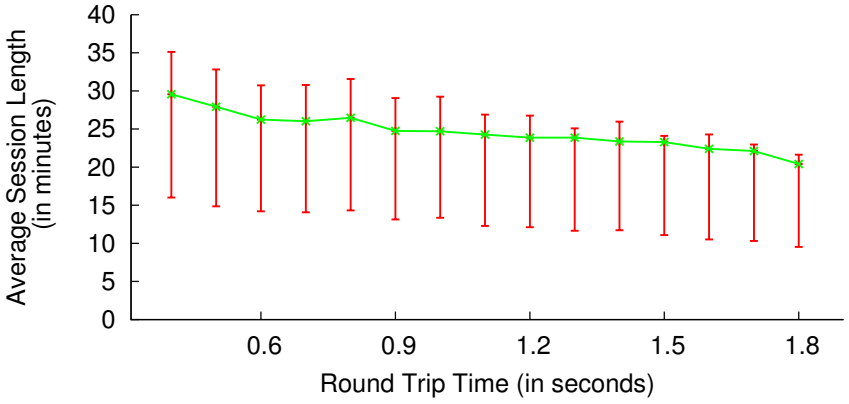

Figure 4.18: Average session lengths for US based cellular players at different RTTs (Bin size of 100ms). The bars show the 25th - 75th percentile values of the session lengths.

Impact of network performance on application usage and revenues

In this section, we analyze the impact of network conditions on PK's usage (session length) and revenues.

How did network performance affect the session lengths?

In Figure 4.18, we show how average user session length for PK varies with average cellular network latency for the session (collected using Insight's active measurements). The latencies are grouped into bins of 0.1 seconds. We find that network performance has a considerable impact on application usage—session lengths decrease with increase in network latencies *i.e.*, users tend to play the game for a lesser time as the network conditions worsen. *An increase of network latencies from 0.3 to 0.9 seconds reduced average session lengths from 30 minutes to 25 minutes (16% reduction).* We observed lower correlation between network latencies and session lengths for SB (not shown here). This is due to the more occasional use of the network and shorter session lengths (Figure 4.7) for SB users compared to PK, so its users are less prone to the impact of poor network performance.

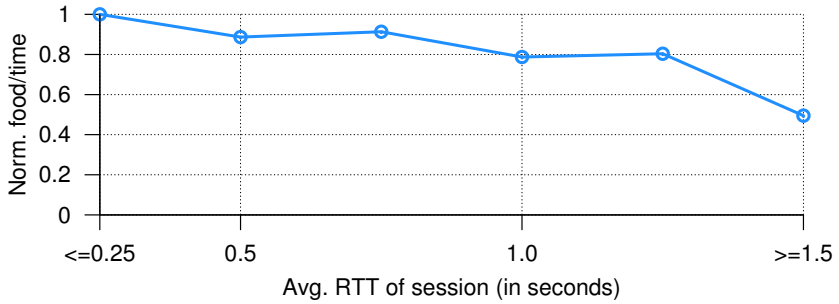

Figure 4.19: Normalized values for food burnt per unit time at different RTTs for US based PK users (Bin size of 250 ms).

How were game revenues from PK impacted by network performance?

In PK, users can purchase virtual currency (Food) with real money which acts as a direct source of revenue for its developers. For PK users, Figure 4.19 shows a high correlation between normalized food consumed per minute and the average network latency—food consumption (and hence the money spent) decreases with the increase in network latencies, thereby affecting application revenues. For example, high latency (e.g., 1.0 secs) sessions had around 20% lesser food consumption compared to low latency (e.g., 300 ms) sessions. *Under bad network conditions (latency $>=$ 1.5 secs), food consumption decreased by 52% indicating sessions with potential revenue losses for the developers.*

Summary. Poor network performance (higher network latencies) resulted in revenue losses (upto 52% lower revenue generate rate) as well shorter average session lengths for PK users (16% shorter). Thus, efforts by cellular ISPs to upgrade their deployments and reduce congestion (e.g., adding WiFi hotspots) is important for application developers and users due to rapidly growing usage of these networks. Also, we observed around 42% of the WiFi traffic was still being received over older 802.11bg networks. Thus, developers should also account for the presence of devices with slower network connections.

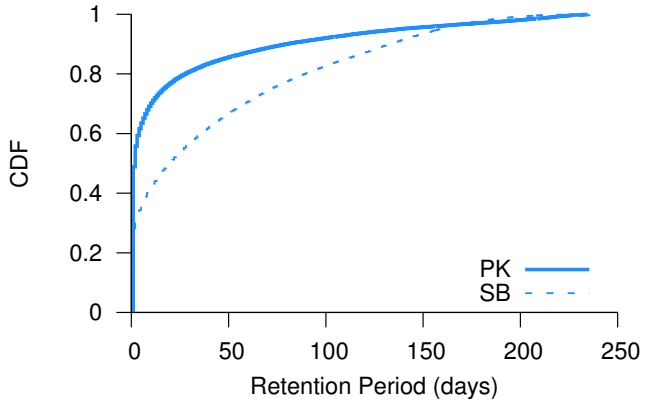

Figure 4.20: CDF of the player retention periods for both applications.

4.5 IMPORTANCE OF USER RETENTION

Attracting more regular users and maintaining user interest is a major focus for serious mobile application developers, because it can lead to more revenues.

How long do users stay involved in the applications, i.e., what is the distribution of retention period of the users?

One measure of user interest in the application is the period for which they use the application. For PK, we compute the number of users joining and leaving the game during Age 2 (March 27, 2009 to November 16, 2009, a total of 235 days). We did a similar analysis for SB users over a period of 235 days (August 2012 - March 2013). If a user first accessed the application on day d1, and if we do not find the user accessing the application after after day d2, we define d2 − d1 as the "retention period" of the user, *i.e.*, the total number of days the user stays in the game.

Figure 4.20 shows the CDF of retention period for both PK (game) and SB (study application). *We observe that around 48% and 27% of PK and SB users, respectively, use the application only for a single day i.e., these users download the application, access it for one or more times during the day and never use the application again.* This indicates that a large proportion of users across both applications had very low retention periods. Many popular sites like AppBrain [22] use

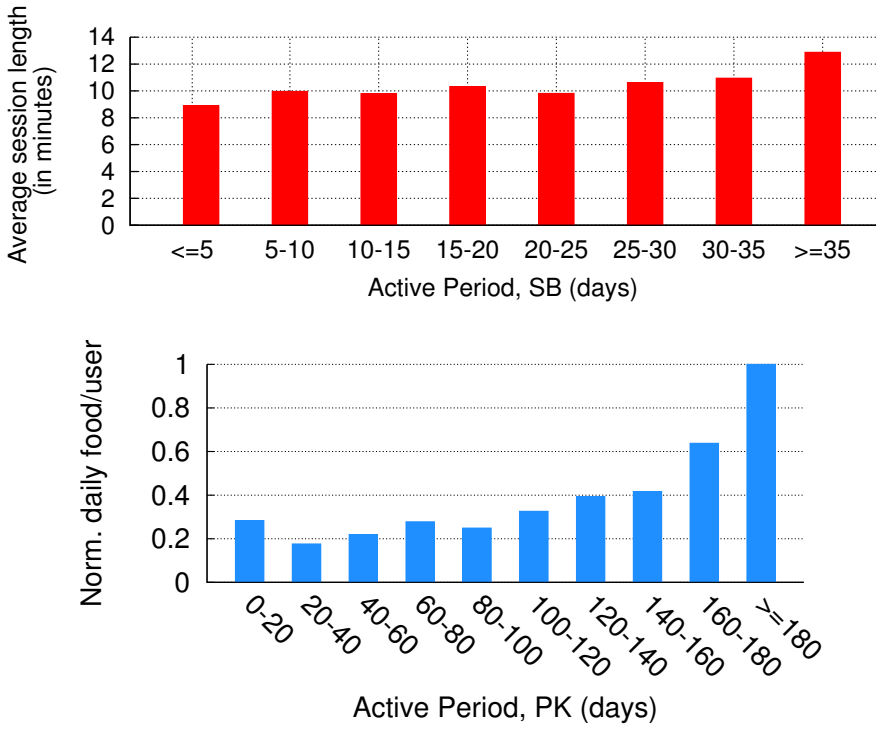

Figure 4.21: (top) Sessions lengths as a function of active periods (SB), (bottom) Active period vs. normalized average Food expenditure per player per day (PK).

downloads to indicate an application's popularity in an app store. However, we observe that downloads alone might not accurately reflect the popularity or user base of an application. For example, only 20% and 42% of PK (game) and SB (study application) users, respectively, were retained for more than one month during this period (235 days).

How did user interest translate into application usage and revenues?

We define the "active period" of a user as the total number of days during which a user accesses an application. For example, if a player plays on day 1 and again on day 5, the active period for the player is two days. For SB, we grouped users based on their active periods over a total period of 3 months. Figure 4.21

(top) shows the median session lengths for users based on their active periods. The graph points to an increasing trend in session lengths with the increase in user active periods. The median *session lengths for regular users with high active periods (>= 35 days) was around 1.5 times (13 minutes vs. 8.5 minutes) compared to intermittent users (<= 5 days)*.

Using Insight, we analyzed all Food expenditure transactions in the PK game during Age 2 (7 months). Figure 4.21 (bottom) shows the relationship between active period and normalized daily average food spent per user. We observe an increasing trend of "food spent per day" as users play the game for more days, *i.e.*, the longer a user stays in the game, the larger is the amount of Food (and therefore, money) spent *per day*. This is especially evident amongst users who played the game for more than 180 days. *These users spend 2.5× more money than users who have played less than 150 days.* For users with active period less than 20 days, the average Food spent is slightly more than some users with longer active periods. This is because new users are given initial Food, and many users with short active periods use up their initial allotment of Food and never buy more.

The analysis shows that it is *crucial to retain old users as they spend more time within the application as well generate higher revenues*. While advertising can help recruit new users, it is necessary to maintain user-interest in the application since older users tend to spend more money. In the case of PK, the developers have increased the frequency of game updates (since Age 3) containing new features and capabilities to maintain user interest. Another way to increase the active periods of non-regular users is to occasionally provide them some free virtual currency. This can provide an incentive to these users to be more active within the game and later generate more revenues for the application.

Summary. Across both PK and SB, a large fraction of the users (27% to 48%) accessed their application for only a single day. This indicates the short retention span of a large proportion of mobile users. Users with larger retention periods generated higher revenues (upto 2.5× for PK). Thus, attracting new users and keeping older users engaged in the application (through feature updates and user incentives) is essential for maintaining application revenues.

We make the following key observations from our data:

Insights from application and device analytics

- The application usage behavior is impacted by its type and userbase. The daily active users for the educational application which has a specific demographic was highly correlated with the day of the week. Such application-specific usage context can be used by developers to reduce server deployment costs: e.g., by dynamically provisioning cloud based servers based on the time of day/week (§4.3).

- The "usability" and type of a platform impacts the application usage. For example, tablet based users were more interactive (upto 2 $\times$) compared to smartphone users across both applications. Amongst Android devices, phones with slide-out keyboards had more user interactions for the PK game than phones with larger screens (§4.3) [2].

- We observed that even on device models with similar CPU, memory and battery capacity, the battery drain caused by the same mobile application had a high variance (upto 3$\times$). Also, high battery drain caused by the application led to lower session lengths across both PK and SB. This points out the need for device specific optimizations for the application to remain energy efficient across a range of devices. For example, controlling the screen brightness on a Kindle Fire device reduced the average battery drain by 40% while using SB. Furthermore, such optimizations can benefit applications since we observed a decrease in session durations with an increase in battery drain on the devices (§4.3).

- A large fraction of users (48% and 27% for PK and SB respectively) used the applications for a single day only. Also, it is important to retain old users as they generate more daily revenues as well as spend more time

[2]The game involved text-based and touch-based interactions.

within the application (§4.5). For e.g., PK's regular users generated 2.5× more daily revenues compared to new users while average session lengths for regular SB users were 1.5× more compared to newer users (§4.5).

Insights combining network and application analytics

- Higher network latencies due to poor network performance reduced user interactivity across both applications. The impact was higher for the PK (upto 40% reduction in interactivity), which requires real-time communication with the servers. Furthermore, poor network performance led to shorter user sessions and loss in application revenues for PK (upto 52% under bad network conditions). Thus, techniques to identify and reduce user-perception of high network latencies (e.g., placing additional servers near regions with high cellular latency) is crucial for such developers (§4.4).
- Poor cellular network performance led to an increase in WiFi access indicating that network quality affected user's choice of networks. The game users (PK) exhibited a higher preference for WiFi networks (69%) compared to the educational app (58%) under poor cellular performance, indicating that choice of the network type is also impacted by the application type (§4.4).

5 RELATED WORK

In this chapter various ongoing and prior research efforts that are related to this thesis. We first discuss research related to monitoring network performance. Then, we discuss work on systems and tools for diagnosing problems in wireless networks. We then discuss approaches for managing WiFi Access Point infrastructure in both enterprise and residential settings. Finally, we focus our attention on work for characterizing and understanding mobile application usage and experience.

5.1 CHARACTERIZING NETWORK PERFORMANCE

We compare WiSe with prior works on characterizing network performance using different platforms – gateways, dedicated packet sniffers/measurement nodes and client devices.

Using residential gateways. The BISmark project [94] aims at understanding the performance of wired access networks in homes through the long-term deployment of gateways in homes serviced by a diverse set of ISPs. The BISmark gateways periodically measure network latency, throughput and home traffic usage patterns to quantify the network performance and usage across different residential ISPs. We use a similar approach but focus on building a framework to study the properties of dense residential wireless networks and deploy multiple APs within the same apartment building. A combination of these two approaches can complement each other by providing a unified view of both wired and wireless access networks.

Using dedicated packet sniffers and measurement nodes. Previous studies [21, 34, 56, 60] have evaluated wireless network deployments in enterprises and homes by collecting passive traces or user traffic statistics. For example, Jigsaw [34] used a large number of passive sniffers in an enterprise WLAN to collect wireless packet traces for debugging wireless problems. This thesis builds upon concepts from this work for synchronizing traces collected

from multiple stand-alone WiFi sniffers. But, instead of using dedicated passive packet sniffers, we perform measurements directly through the wireless APs. This allows us to obtain more comprehensive information about the AP's view of the network in addition to packet level statistics (e.g., airtime utilization, non-WiFi devices etc.).

By performing controlled experiments in three houses using dedicated measurement nodes, authors in [74] observed the properties of WiFi links in home environments such as high asymmetry in links and variability in link performance at different locations within homes. We measure longitudinal WiFi link performance in homes using Access Points and identify the causes of occasional poor performance experienced by links.

Using client devices. Other research efforts [38, 44] focused on end-host techniques for measuring end to end performance in home networks. But, leveraging end host support in homes may not be always feasible, due to the diverse and heterogeneous client devices that use residential WiFi networks. Also, many client devices (e.g., wireless TVs, video streaming devices) do not provide any visibility into the lower layer properties of the network. We use AP-centric techniques, since it allows us to monitor the performance of WiFi links across diverse and heterogeneous client devices in homes using a single instrumented vantage point.

SpeedTest [72] and Netalyzr [50] are popular client based network measurement tools to estimate end to end latency and throughput using active data probes. We avoid using any active measurements to avoid increasing wireless contention experienced by residential WiFi environments.

5.2 TECHNIQUES TO DIAGNOSE PROBLEMS IN WIRELESS NETWORKS

We now discuss prior work on model based techniques to predict overall wireless performance and tools to diagnose the causes of poor wireless experience.

Models to predict wireless throughput. One of the earliest pieces of work on model based throughput prediction for WiFi networks was by Bianchi *et*

al. [26]. The technique uses MAC layer parameter information and proposes an analytical model to predict wireless throughput. Some model based techniques [55] require lightweight active probing to predict link capacity and build conflict graphs. Mirza et al. [66] propose throughput prediction models for opportunistic wireless networks using machine learning techniques and time series analysis of throughput values. Our focus is to passively estimate a wireless link's performance (saturation throughput) by only using coarse-grained, MAC layer observations available at WiFi Access Points (e.g., data rates used, airtime utilization), since any additional traffic from the WiSe APs may impact existing traffic.

Detecting interference in WiFi networks. To avoid measurement overhead, passive techniques have been proposed [61, 86, 89] for detecting the presence of WiFi to WiFi link interference in WLANs. We build upon prior work from PIE [89] that uses microsecond level timing information to passively detect the presence of wireless interference for a link by comparing the link's loss rates in presence and absence of overlapping packets from a nearby WiFi transmitter. But, PIE is designed for enterprise WLANs and assumes the presence of a low-latency and tightly managed control plane. This assumption is not true for residential WiFi networks and thus, we had tailor the technique for residential WiFi networks.

In our prior work, Airshark [84], we built a tool to detect the presence of non-WiFi interferer devices (e.g., cordless phones, microwave ovens etc.) using commodity WiFi cards. The tool uses subcarrier level RSSI data available from WiFi cards as inputs to machine learning models (trained offline) to detect the activity of nearby non-WiFi interferer devices. Other approaches require specialized hardware to detect non-WiFi devices. Hong et. al [45] use a modified channel sounder and propose cyclostationary signal analysis to accurately detect non-WiFi interference. RFDump [57] uses GNURadio and employs timing/phase analysis along with protocol specific demodulators to detect devices. These hardware-specific approaches cannot be directly employed by commodity WiFi Access Points in home. We ported Airshark [84] to our home WiFi Access Point platform (Chapter 2) to detect the presence of nearby non-

WiFi devices in homes. We also our leverage techniques from our prior work on WiFiNet [85] to correlate neighboring WiFi and non-WiFi device activity and measure interference impact.

5.3 MANAGING WIFI ACCESS POINT INFRASTRUCTURE

We now present related work in the areas of management and co-ordination of WiFi Access Points.

SDN style management of home APs. Most enterprise WLANs today use some form centralized management using a central controller connected to each enterprise WiFi Access Point. The majority Access Point vendors (e.g., Aruba [24], Cisco [36] etc.) use an on-premises controller device to manage and configure various parameters on Access Points – WiFi channels, transmit power, SSID, traffic scheduling [88], firewalls and ACLs. Meraki [64] pioneered the concept of cloud based management of Access Points in enterprise WLANs based on an proprietary solution that configures a homogeneous Access Points remotely though the cloud. Proprietary solutions can work well in enterprise WLANs since they are usually under a single administrative control. In dense residential WiFi networks, such as an apartment building, several independent APs coexist in the vicinity of each other.

Dyson [69] proposed a centralized framework to manage APs and clients in enterprise WLANs using a set of APIs at both APs and clients. We propose an open API for for home APs to enable co-operation between heterogeneous neighboring APs in residential settings. Our focus is to enable centralized co-ordination and management of independent and co-located home APs using an open API implemented by these APs.

OpenFlow [63] is a popular SDN framework designed for managing routing policies and other configurations in wired networks. In [102], the authors propose an OpenFlow based centralized framework to allow multiple service providers to share the same physical infrastructure in residential broadband networks (e.g., for bandwidth sharing). Our proposed COAP framework (Chapter 3) is complementary to such efforts in dense residential settings by

providing capabilities to manage the shared wireless resources at homes (e.g., based on the traffic context).

Prior work on using OpenFlow for 802.11 WiFi networks [95, 101] have focused on building an experimental platform and managing mobility of devices. Our goal is to use the SDN approach to actively manage the wireless parameters of 802.11 networks and build management frameworks, especially for residential settings. It is also possible to use other AP management standards (e.g., CAPWAP [51], RDK [14], TR-069 [28]) to implement COAP related APIs.

Using multi-AP support in non-enterprise 802.11 deployments. Prior research has proposed mechanisms to leverage support from multiple APs for sharing the data-plane through client-based backhaul aggregation [43, 54], use AP virtualization to improve video performance through bandwidth sharing [91] and mitigate hidden terminals using distributed algorithms [61]. With COAP, we motivate the use of a cloud-based framework to enable cooperation between nearby home APs and mitigate wireless problems in residential WiFi deployments. This is done by aggregating airtime utilization information from nearby APs for channel configuration, coordinating airtime access across APs to alleviate both channel congestion and hidden terminal scenarios and learning from prior activity to predict per-AP traffic usage patterns and future non-WiFi interference.

5.4 UNDERSTANDING MOBILE APPLICATION USAGE

We now discuss some related work in the areas of characterizing smartphone and mobile application usage, tools for mobile application analytics and understanding usage of specific mobile applications (e.g., gaming, social networks).

Mobile Analytics. Mobile application analytics provided by app stores like Android Market include limited information about application usage e.g., number of downloads, geographical spread of users, devices used etc. Recently, few commercial analytics solutions [5, 7, 9, 10] for iOS and Android platforms have started providing improved capabilities for tracking application

Metrics	Insight	Flurry	AppClix	Android
Session	✓	✓	✓	✓
Device	✓	✓	✓	✓
Location	✓	✓	✓	✓
In-app activity	✓	✓	✓	
User retention	✓	✓	✓	
Network	✓			
Resource	✓			

Table 5.1: Comparing the feature categories of a few popular commercial mobile analytics solutions with Insight.

usage. These solutions are mainly focused on providing the business analytics (e.g., statistics about users, locations, user retention etc.) to the developers. Table 5.1 compares the broad feature categories provided by Insight with some popular analytics solutions. In addition to these analytics, Insight also measures the application footprint (e.g., CPU, memory, battery, screen brightness etc.) on the mobile devices and leverages the application user base to infer the impact of network on application usage and revenues. Further, the data from commercial analytics solutions is only available to the developers of the applications. Insight allowed us to partner with the developers to perform this large scale study across multiple years and share the results with the community. AppInsight [83] instruments the application binaries to identify the critical paths in application execution. Such approaches are complimentary to Insight's goals of understanding the impact of the aforementioned factors on the users' experience. For example, Insight allows developers to add application specific measurements that can be used to understand aspects related to in-application user behavior.

Smartphone and Mobile Application usage. Prior research [39, 41, 81, 87] has looked at the characteristics of smartphone usage, such as device and application usage patterns by instrumenting or collecting traces from smartphone devices. The authors in [41] perform a detailed characterization of user session lengths, network and app usage patterns on 255 users while [81]

focuses its study on 24 users belonging to a specific socio-economic group. [87] uses smartphone traces to build models for predicting the applications used based on the current context. Other studies have deployed standalone apps to study specific aspects such as contextual uses of different smartphone apps (AppSensor [27]) and network performance on smartphones (3GTest [49]). In [100], the authors utilize a snapshot of the network traces from a cellular ISP to study the similarities in the usage of different types of smartphone applications. Through Insight, we use applications and client devices as a vantage point across more than a million users and analyze it along multiple dimensions (e.g., device type, network, user retention). This unique setup captures the impact of factors, such as network performance, on the end user's experience while using an application as well as revenue generation for developers.

Gaming Studies. Games constitute one of the major types of Internet applications today. There have been many studies on online games but to the best of our knowledge, our work was one of the first to characterize the workload, usage and player behavior of a game specifically meant for mobile handheld devices such as smartphones, PDAs etc.

Chambers et al [32, 33] analyzed some popular online game workloads by concentrating on FPS genre of games (Counter-Strike) meant for a different set of users (mainly desktop and laptop users). In [70], the focus is on the analysis of third party applications for OSNs (Online Social networks), one of which is a game. This study concentrates on the underlying social networking aspects of the third party applications. In [71] the authors further studied how Facebook forward/process the requests/response from third-party OSN applications, and its impact on the overall delay performance perceived by end-users. Our study is focused on the a popular MMORPG game available for smartphones and handheld platforms (Apple's iOS [23] and the Google's Android Platform [46]).

Prior research [42, 78, 97] has studied popular MMORPGs such as World of Warcraft and EVE Online to analyze and predict the trends in player populations, distributions and game usage. [42] does a long term study of the game EVE Online, but it only focuses on the issues of general MMORPG game usage

predictability and player population predictability. The studies [78, 97] based on the World of Warcraft game are limited to a single realm and do not provide a snapshot of the entire game and about the different players spread across the globe. In [79], Pittman et al. continue their work on a larger dataset using two MMORPGs to create a model for analysis and simulation of the virtual world and player populations. In our study of a large mobile MMORPG (Parallel Kingdom [77]) using the Insight toolkit (Chapter 4), we analyze properties such as player interactivity and factors impacting game revenues.

6 CONCLUSIONS AND FUTURE WORK

In this thesis, we have shown how to design and implement systems that leverage different vantage points in the wireless ecosystem of Access Points, client devices and mobile applications to better understand and improve user experience. Specifically, we developed systems that leverage residential WiFi Access Points to diagnose and mitigate the different causes of poor WiFi performance. We show how the proposed system can be implemented on commodity WiFi Access Points today to mitigate channel congestion, WiFi and non-WiFi interference as well as coverage issues. We also develop a mobile analytics toolkit that collects information about network behavior, application usage and footprints, platform statistics and user actions to understand the impact of different factors on user experience and their consequences (e.g., application revenues, session lengths etc.).

We now highlight the main contributions of this thesis in the next section, and then present problems for future research endeavors.

6.1 CONTRIBUTIONS

The main contributions of this thesis are listed below:

An infrastructure to understand residential wireless experience. We described a unique measurement and management infrastructure, WiSe (Chapter 2), to perform inline measurements of wireless properties in homes that uses APs as vantage points. We operated this infrastructure for more than 9 months in 30 homes with diverse characteristics — dense apartments and sparser suburban neighborhoods, some residents directly owned the APs while others use a shared infrastructure deployed in the apartment. We propose a passive metric, called Witt, that captures the wireless experience of WiFi links using simple passive measurements such as airtime utilization, packet losses and rates and contention from local links. It uses a simple model to estimate the TCP throughput that can be supported by the wireless link. We use this metric to analyze the performance of wireless links in different environments

and identify the cases with poor performance using data collected from the WiSe APs. We also built a diagnostic toolkit for the WiSe APs to identify the causes during periods of poor performance.

In our measurements, WiFi experience was poor for 2.1% of the time. During these periods, the causes of poor performance were mainly dependent on the nature of the WiFi deployment. In an apartment building where Access Points were individually owned and operated, channel congestion was the main cause (62%) of poor WiFi performance due to the high density of APs. Weak signal strengths due to coverage issues was the main cause of poor WiFi performance (52%) in another apartment building where Access Points were centrally deployed and managed by an ISP. We also observed intermittent short periods (1 - 7 minutes) of WiFi and non-WiFi interference resulting in more than 50% degradation in throughput. We also observed that a majority of observed home WiFi Access Points (56%) are statically configured, indicating that they do not perform any adaption in response to poor WiFi performance.

A SDN framework to enable co-ordination and co-operation between home WiFi gateways. Motivated by the measurement results using the WiSe infrastructure, we designed the COAP framework. COAP proposes a vendor-neutral open API and a cloud based controller to enable cooperation between heterogeneous and collocated home WiFi APs in dense residential deployments (e.g., apartment buildings). We extended the open-source OpenFlow Software Defined Networking (SDN) framework for commodity home WiFi APs. We developed multiple applications using the COAP framework to motivate the benefits of using this centralized framework — better channel assignments using airtime utilization information from co located home APs and managing airtime access of neighboring APs to reduce channel contention for important flows (e.g., HTTP based video) as well as mitigate hidden terminal interference. We also show that learning about previous wireless activity using the COAP framework can enable the controller to better predict future non-WiFi interference and wireless traffic properties. This can allow the controller to pro-actively configure home APs for interference mitigation as well as perform traffic aware adaptations at APs.

Using a deployment of 12 COAP enabled WiFi APs in a apartment building, we observed upto 47% reduction in channel congestion experienced by home WiFi APs. We show that by co-ordinating packet transmissions of interfering WiFi APs through coarse grained time slots (10 - 20 ms), it is possible to significantly reduce MAC layer retransmissions (upto 50% in our experiments) and increase spectral efficiency. Furthermore, we were able to predict future non-WiFi device interference from microwave ovens in a apartment building using only 5 days of training data. This opens up another approach to improve residential WiFi experience by predicting future interference.

An analytics toolkit to understand mobile application experience. Through the development and deployment of our embedded analytics toolkit, Insight, within two popular applications, we presented a large scale and long term study of these applications across more than a million users. In addition to providing aggregate application analytics, Insight allows developers to understand the impact of various factors such as device capabilities and settings and network performance on the application usage and user experience. By using this vantage points within the two applications, we study and contrast these different application types (gaming vs. study application) across different dimensions: session, device impact, resource overhead, network impact, application revenues and user retention. Through this study, we show that capturing and analyzing the inter-dependencies between these factors is essential for a better understanding of smartphone application usage in the wild. We also use these two applications embedded with Insight to contrast the usage characteristics and the impact of network performance based on the nature and requirements of the application.

The insights available from such analytics can help developers identify the bottlenecks that degrade user experience and quantify their impact. Poor network performance reduced user interactivity for both mobile applications analyzed using Insight. The reduction was higher for the mobile gaming application (upto 40%) due to greater network requirements. Furthermore, we observed upto 52% reduction in application revenues under poor network

conditions. Also, we observed that the device type and platform led to 2 × variability in user interactivity for both mobile applications.

Using Insight, we have collected a significant volume data which captures network performance when connecting across diverse locations, through different types of networks, and across different times. We intend to use this data to help users better understand their networks. For this purpose, we have released an application Network Test [11] that performs a simple set of measurements to estimate network latency from the mobile device. It leverages all our network measurement data to perform a relative comparison — the network latency information of different users of a region are compared against a user's own latency to provide a relative grade. A partial data set obtained with our measurements is also currently visualized at `http://networktest.org`. We hope that the users will also be able to benefit from our work.

6.2 FUTURE WORK

This dissertation proposed important building blocks to improve wireless experience by creating measurement and management capabilities for different vantage points – WiFi Access Points, client devices and mobile applications. We now discuss potential future research directions in this space.

Building smarter wireless gateways. With COAP (Chapter 3), we proposed the idea that home WiFi Access Points or gateways can be centrally managed to improve wireless experience through co-operation and co-ordination. These gateways also have general-purpose compute capabilities in-addition to rich data about wireless activity (WiFi and non-WiFi) which has been not been tapped into yet. This presents the possibility to explore new applications that can leverage the wireless gateways as a vantage point. The upcoming generation of WiFi gateways will have additional built-in radios such as LTE, Bluetooth and ZigBee to support Internet of Things (IoT) devices (e.g., temperature sensors, energy monitors). These trends present opportunities to use wireless gateways as a hub for creating novel applications. Following are a couple of example example applications – (a) context-aware applications for home energy

management and (b) location-aware applications that leverage information about user and device activity.

If wireless gateways can estimate users' real-time locations within homes by tracking their WiFi activity and/or wearable devices, they can assist in optimizing energy consumption in homes. For example, they can make decisions like turning off the lights and controlling heating activity based on user location. Health monitoring is another interesting application. Information about users' movements and network activity available at the gateways can complement other data sources (e.g., wearables and smartphone sensors).

Improving multimedia experience in wireless networks. In this dissertation, we developed the Witt metric (Chapter 2) to predict estimated TCP throughput using passively observed wireless statistics from commodity home WiFi Access Points. New techniques need to be developed that can leverage these statistics to predict application specific performance. For example, the consumption of multimedia services such as over-the-top video (e.g., Netflix) and IPTV services (e.g., AT&T U-verse) is increasing rapidly. An open research problem is to develop a technique to measure the impact of home network performance (wired and wireless) on the application related parameters that impact user-experience. In the case of HTTP-based video streaming applications, the main challenge is to estimate different Quality of Experience (QoE) metrics (e.g., playout-buffer size, buffering periods, start-up times) using only the traffic properties captured at the AP without any instrumentation at the clients (due to the heterogeneous platforms and devices involved). Using the Access Point or wireless gateways can also help isolate the causes of performance degradation due to problems occurring on the wired ISP path versus the wireless hop. This mechanism can be helpful for ISPs to passively estimate and improve the QoE experienced by its users across different services (e.g., VOIP, video, gaming) and client devices.

Security and trust framework for COAP. With COAP, we proposed vendor-neutral APIs for third-party cloud-based controllers (e.g., ISPs). Security and trust are two important challenges that need to be tackled within this model.

Since home APs are individually owned and deployed by users, the authenticity of wireless statistics collected from these APs needs to ensured. Malicious WiFi APs may report false data to the controller to obtain more than their fair share of wireless resources. Similar problems can also result due to collusion between multiple home APs. Abstractions for trusted home APs are required to ensure proper functioning of COAP based management applications. Similar challenges have been faced while designing trusted sensors for other platforms, such as smartphones [59]. Exploring these approaches is a topic for future research.

Leveraging clients for WiFi network diagnostics. Handheld devices such as smartphones and tablets are widely used across both enterprise and residential WiFi networks. These WiFi clients provide useful diagnostic wireless metrics that can complement Access Points' view of the wireless network. Examples of prior research efforts that leverage WiFi clients include techniques for channel planning [67] and measuring bandwidth using active probes [49]. MAC layer wireless metrics from WiFi clients such as packet error status, data rates and timestamps can be used to diagnose other problems, such as hidden terminals. Additional research need to be done to explore approaches that use these wireless metrics from both APs and WiFi clients to diagnose WiFi related issues.

From the management perspective, extending the SDN approach to WiFi clients is another topic for future research. With COAP, we only focused on designing APIs and mitigation techniques using WiFi APs. Using clients opens up another dimension in the overall design space. With the increasing popularity of file-syncing applications such as Dropbox, the need to manage and shape uplink flows will be important. APIs and techniques to manage client traffic in these scenarios will be useful in improving the wireless experience.

A IMPACT OF THIS DISSERTATION

We briefly summarize the publications and the broader impact of this thesis below.

- **WiSe**: We designed and deployed a customized off-the-shelf WiFi router platform in over 30 residential apartments (running over a period of more than 9 months). We also collaborated with Madison-based local ISPs to debug their residential WiFi deployments through this work. In this unique study, we leverage multiple Access Points with two different types of multi-dwelling units to analyze major causes of poor wireless in residential settings. To the best of our knowledge, the WiSe infrastructure incorporated the most comprehensive set of WiFi diagnostics capabilities in a residential setting as part of an academic study. WiSe was published at MobiCom 2013 and the datasets were publicly released for the research community.

  WiSe was built upon our earlier work on Airshark [84] and WiFiNet [85] to understand the impact of WiFi and non-WiFi device interference on WiFi links. As part of our work to port Airshark to the WiSe infrastructure, we developed a full prototype of Airshark. This work was appreciated by both academia and industry and won the second the second prize at the InterDigital Innovation Challenge 2012 [3, 30]. Later, we also leveraged this effort to release Airshark as part of an Android application for wireless diagnostics called WiSense [18, 19]. WiSense was featured by Google on the Google Play Store [47] and has been used by some companies for RF diagnostics. The kernel patches required to enable spectral scan capabilities for Airshark on the Android platform were made public to the broader community in 2014 [16].

- **COAP**: COAP proposed a Software Defined Networking (SDN) framework to enable centralized management of home WiFi APs in dense residential deployments (e.g., multi-tenant apartment buildings). We proposed open APIs requiring software only upgrades that can be imple-

mented by Access Point vendors to enable cooperation and coordination between home APs through a cloud controller. This work won the best presentation award at the MobiArch 2014 workshop (co-located with MobiCom 2014) and the full paper was published at INFOCOM 2015.

- **Insight**: Insight allows mobile application developers to collect a broad range of device, performance, networks and user activity related metrics. This enables them to understand the impact of different factors on application usage and revenues. In the first-of-its kind of academic study, Insight served as a distributed lens from inside applications with greater than one million users over more than three years. Our study of a popular MMORPG using Insight won the best paper award at the MobiGames 2012 workshop (co-located with SIGCOMM 2012) and was fast-tracked to the ACM SIGCOMM Computer Communication Review. The full paper on Insight was a best paper award nominee at ACM CoNEXT 2013 (one of the top 4 papers at the conference). We publicly released the source code for Insight on GitHub [8].

B OPENFLOW PROTOCOL MESSAGES FOR COAP

In this chapter, we provide details about the extensions to the OpenFlow specification to add COAP related features (Chapter 3).

B.1 MESSAGE TYPES

1. To collect COAP based wireless statistics (Table 3.1), new types are added to *enum ofp_stats_types*. Figure B.1 shows these new types. In §B.2, we further discuss the details of the structures used to collect these statistics from COAP APs.

```
enum ofp_stats_types {
  OFPST_DESC,
  OFPST_FLOW,
  OFPST_AGGREGATE,
  OFPST_TABLE,
  OFPST_PORT,
  OFPST_QUEUE,

  .
  .
  .

  /* Added to support COAP messages. */
  OFPST_APINFO,       /* AP information */
  OFPST_UTIL,         /* Airtime Utilization */
  OFPST_STATION,      /* Station statistics */
  OFPST_SYNC_BEACON,  /* Overheard beacon information */
  OFPST_NONWIFI,      /* Observed non-WiFi devices */
  OFPST_BEACON,       /* Beacon statistics */
  OFPST_PASSIVE,      /* Overheard WiFi activity */
  OFPST_TRAFFICINFO,  /* Traffic type information */
  // End

  OFPST_VENDOR = 0xffff
};
```

Figure B.1: Adding COAP related types to *ofp_stats_types* for collecting various wireless statistics from the COAP APs.

2. To transmit wireless configuration updates from the controller to the COAP APs, a new message type (*OFPST_SET_WIRELESS_CONFIG*) is added to *ofp_type* (Figure B.2). For example, to set an AP's channel, the

COAP controller uses the this type to send a channel update message to the AP.

```
enum ofp_type {
  /* Immutable messages. */
  OFPT_HELLO,               /* Symmetric message */
  OFPT_ERROR,               /* Symmetric message */
  OFPT_ECHO_REQUEST,        /* Symmetric message */
  OFPT_ECHO_REPLY,          /* Symmetric message */
  OFPT_VENDOR,              /* Symmetric message */

  .
  .
  .

  /* For COAP: Added to manage wireless configuration at the APs. */
  OFPT_SET_WIRELESS_CONFIG  /* Controller/switch message */
};
```

Figure B.2: Adding the *OFPT_SET_WIRELESS_CONFIG* message type to receive wireless configuration updates from the controller.

```
/* AP information statistics. */
struct ofp_apinfo_stats {
  char apmacid[MACID_LENGTH];     /* AP MAC ID */
  uint32_t timestamp;             /* Unix timestamp */

  uint8_t nic_count;              /* NIC number */
  uint8_t is_2GHz_supported;      /* 2 GHz support */
  uint8_t is_5GHz_supported;      /* 5 GHz support */

  uint8_t is_2GHz_ht20_support;   /* HT20 on 2GHz? */
  uint8_t is_2GHz_ht40_support;   /* HT40 on 2GHz? */
  uint8_t is_5GHz_ht20_support;   /* HT20 on 5GHz? */
  uint8_t is_5GHz_ht40_support;   /* HT40 on 5GHz? */

  uint8_t pad;                    /* Align to 32 bits */
};
OFP_ASSERT(sizeof(struct ofp_apinfo_stats) == 32);
```

Figure B.3: OpenFlow structure used by a COAP AP to report airtime utilization information to the controller.

The section describes the OpenFlow structures to collect COAP related statistics described in Table 3.1.

1. The *ofp_apinfo_stats* structure (figure B.3) provides information about each wireless NIC present on the Access Point.
2. The *ofp_util_stats* structure (figure B.4) is used to collect parameters related to airtime utilization on the AP's current channel.

```
/* Airtime utilization statistics. */
struct ofp_util_stats {
  uint16_t type;          /* util or util hop */
  short util_val;         /* Utilization value. */
  uint32_t active;        /* Active period (in seconds) */
  uint32_t busy;          /* Busy period (in seconds) */
  uint32_t receive;       /* Receive period (in seconds) */
  uint32_t xmit;          /* Transmit period (in seconds) */
  uint32_t channel;       /* WiFi Channel */
  uint32_t timestamp;     /* Unix timestamp */
  int noise_floor;        /* Noise floor (in -dBm) */
};
OFP_ASSERT(sizeof(struct ofp_util_stats) == 32);
```

Figure B.4: OpenFlow structure used by a COAP AP to report airtime utilization information to the controller.

3. The *ofp_beacon_stats* structure (figure B.5) is used to obtain information about neighboring APs located in the vicinity of the COAPAP. The specific AP performs a scan on WiFi supported WiFi channel to detect neighboring APs' beacons.

```
/* Neighboring AP information */
struct ofp_beacon_stats {
  char apmac[MACID_LENGTH]; /* AP MAC ID */
  uint32_t timestamp;       /* Unix timestamp */
  int rssi;                 /* Signal strength */
  uint16_t channel;         /* Channel */
  uint8_t pad[2];           /* Align to 32-bits. */
};
OFP_ASSERT(sizeof(struct ofp_beacon_stats) == 32);
```

Figure B.5: OpenFlow structure used by a COAP AP to report neighboring AP information (observed through beacons) to the controller.

4. The *ofp_station_stats* structure (figure B.6) reports a COAP AP's download packet transmission statistics to a local WiFi client. The client is identified by a unique ID (*client_macid*) which can either be the client MAC address or a hashed version of the MAC address for privacy purposes. The *retry_string* is a formatted string containing the packets transmitted and retried by the AP at each individual PHY rate.

```
/* Per-client packet transmission summary */
struct ofp_station_stats {
  char client_macid[MACID_LENGTH];    /* Client MAC ID */
  char retry_string[RATESTR_LENGTH];  /* PHY rate stats */

  uint32_t packet_count;              /* Packets sent */
  uint32_t packet_retries;            /* Packets retried */

  int rssi;                           /* RSSI */
  // uint32_t channel;                /* WiFi channel */
  uint32_t timestamp;                 /* Unix timestamp */
};
OFP_ASSERT(sizeof(struct ofp_station_stats) == 196);
```

Figure B.6: OpenFlow structure used by a COAP AP to report per-station download packet transmission statistics to the controller.

5. The *ofp_syncbeacon_stats* structure (figure B.7) reports beacons from other APs overheard by a COAP AP. The *timestamp* field records a 32 bit timestamp for when the beacon was observed. This data is collected from nearby COAP APs by the COAP controller to obtain their relative clock offsets. The clock offsets are used to synchronize the COAP APs' clock for the *SetAirtimeAccess* API (Table 3.1).

```
/* Beacon information - For time synchronization */
struct ofp_syncbeacon_stats {
  char apmac[MACID_LENGTH]; /* Access Point MAC ID */
  uint32_t timestamp;       /* Timestamp value */
  uint16_t seq;             /* Sequence number.*/
  uint16_t channel;         /* Channel */
};
OFP_ASSERT(sizeof(struct ofp_syncbeacon_stats) == 28);
```

Figure B.7: OpenFlow structure used by a COAP AP to report timestamped overheard beacons to the controller. The controller used this information for inter-AP time synchronization.

6. The *ofp_nonwifi_stats* structure (figure B.8) provides information about nearby non-WiFi activity (e.g., cordless phones, microwave ovens) observed by COAP APs. The information can be helpful to build applications that detect and mitigate non-WiFi interference using commodity WiFi APs. In our implementation, we use Airshark [84] to provide this information.

```
/* Non-WiFi device activity */
struct ofp_nonwifi_stats {
  uint32_t timestamp;    /* Unix timestamp for the record */
  uint32_t start_ts;     /* Unix timestamp when
                            activity is first detected */
  uint32_t end_ts;       /* Unix timestamp when
                            activity is last detected */

  uint32_t duration_sec; /* Activity duration in seconds */
  int rssi;              /* Observed signal strength */

  uint16_t start_bin;    /* Starting WiFi subcarrier (0-55) */
  uint16_t center_bin;   /* Peak RSSI subcarrier */
  uint16_t end_bin;      /* Ending WiFi subcarrier */

  uint16_t device_type;  /* Device type (e.g., FHSS phone) */

  uint32_t channel;      /* WiFi channel with activity */
};
OFP_ASSERT(sizeof(struct ofp_nonwifi_stats) == 32);
```

Figure B.8: OpenFlow structure used by a COAP AP to report non-WiFi device activity to the controller.

7. The *ofp_client_stats* structure (figure B.9) provides client related metadata such the mac_id (mac address or hashed mac address), host_name and optional (device_type). This information can used by the COAP controller to determine the client "context". Such information can be potentially used for predicting future client activity based on prior history (e.g., watching long running video streams) and perform adaptation at neighboring APs to reduce channel contention.
8. The *ofp_passive_stats* structure (figure B.10) is used to report external WiFi activity to the controller. Each instance provides aggregate packet transmission statistics for a single external WiFi link (represented by the *sender, receiver* fields). This is useful for knowing the WiFi activity of nearby WiFi APs, which are not a part of the COAP framework.

```
/* Meta data about the associated clients */
struct ofp_client_stats {
  char mac_id[MACID_LENGTH];    /* Client MAC ID */
  char host_name[NAME_LENGTH];  /* Client's hostname */
  uint32_t timestamp;           /* Unix timestamp */
  long apip;                    /* Local IP Address */
  uint32_t dev_type;            /* Device type */
};
OFP_ASSERT(sizeof(struct ofp_client_stats) == 72);
```

Figure B.9: OpenFlow structure used by a COAP AP to report client metadata to the controller.

```
/* Passively collected neighboring WiFi link statistics */
struct ofp_passive_stats {
  char sender[MACID_LENGTH];         /* Sender's MAC ID */
  char receiver[MACID_LENGTH];       /* Receiver MAC ID */
  char retry_string[RATESTR_LENGTH]; /* PHY rate stats */

  uint16_t average_packet_length; /* Average packet length */
  uint16_t average_rate_times10;  /* Average PHY rate (x10) */

  int rssi;                       /* Signal strength */
  uint32_t packet_count;          /* Packets transmitted */
  uint32_t packet_retries;        /* Packets retried */
  uint32_t channel;               /* WiFi channel */
  uint32_t timestamp;             /* Unix timestamp */
};
OFP_ASSERT(sizeof(struct ofp_passive_stats) == 224);
```

Figure B.10: OpenFlow structure used by a COAP AP to report observed external WiFi link statistics to the controller.

9. The *ofp_trafficinfo_stats* structure (figure B.11) provides information about the traffic characteristics to the controller as a formatted string. The controller can leverage this information to predict future traffic characteristics and perform adaptations at nearby COAP APs to load balance the traffic across different channels.

```
/* Report statistics about the traffic type */
struct ofp_trafficinfo_stats {
    char trafficinfo_string[TRAFFICINFO_STRING_LENGTH];
};
OFP_ASSERT(sizeof(struct ofp_trafficinfo_stats) == 1024);
```

Figure B.11: OpenFlow structure used by a COAP AP to report traffic type statistics to the controller.

10. The *ofp_diagnostic_stats* structure (figure B.12) provides a formatted string containing packet level summaries for diagnostic purposes; such as detecting the presence of hidden terminal interference. The per-packet information only consists of wireless transmission related parameters (e.g., PHY rates, MAC timestamp, packet retries, length etc.). During periods of poor WiFi performance, the COAP controller can collect this information to detect the causes of high interference scenarios to apply the required mitigation strategies.

```
/* Report statistics about the traffic type */
struct ofp_diagnostic_stats {
    char diagnosticstats_string[DIAGNOSTICSTATS_STRING_LENGTH];
};
OFP_ASSERT(sizeof(struct ofp_diagnostic_stats) == 30720);
```

Figure B.12: OpenFlow structure used by a COAP AP to report packet level summaries to the controller.

[1] 2009 Best App Ever Awards. http://www.bestappever.com/awards/2009/winner/mmgm.

[2] 2011 Mashable Awards Finalists. http://mashable.com/2011/11/21/2011-mashable-awards-finalists/.

[3] Airshark Demo. https://www.youtube.com/watch?v=riyp4My0LJ8.

[4] Amazon EC2 pricing. http://aws.amazon.com/ec2/pricing/.

[5] AppClix. http://www.mobilytics.net/.

[6] Click Modular Router. http://www.read.cs.ucla.edu/click/click.

[7] Flurry. http://www.flurry.com/.

[8] InsightClient GitHub repository. https://github.com/patroashish/InsightClient.

[9] KISSmetrics. http://www.kissmetrics.com/.

[10] Localytics. http://www.localytics.com/.

[11] NetworkTest. http://networktest.org/.

[12] OpenWrt. https://openwrt.org/.

[13] Project Floodlight. http://www.projectfloodlight.org/floodlight/.

[14] Reference Design Kit. http://rdkcentral.com/about-rdk/.

[15] SamKnows. http://www.samknows.com/broadband/index.php.

[16] Spectral Scan patches for Linux ath9k_htc driver. http://pages.cs.wisc.edu/~patro/htc_spectral/.

[17] Weka 3: Data Mining Software. http://www.cs.waikato.ac.nz/ml/weka/.

[18] WiSense. http://research.cs.wisc.edu/wings/projects/wisense/.

[19] WiSense Demo. https://www.youtube.com/watch?v=8gRVVAy-Uqs.

[20] Agarwal, Sharad, Ratul Mahajan, Alice Zheng, and Victor Bahl. 2010. Diagnosing Mobile Applications in the Wild. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 22:1–22:6. Hotnets-IX, New York, NY, USA: ACM.

[21] Akella, Aditya, Glenn Judd, Srinivasan Seshan, and Peter Steenkiste. 2005. Self-management in Chaotic Wireless Deployments. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, 185–199. MobiCom '05, New York, NY, USA: ACM.

[22] AppBrain. Top Android Apps and Games on Google Play. http://www.appbrain.com/.

[23] Apple. iPhone Developer Center. http://developer.apple.com/iphone/.

[24] Aruba. Mobility Controllers. http://www.arubanetworks.com/products/networking/controllers/.

[25] Balasubramanian, Niranjan, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, 280–293. IMC '09, New York, NY, USA: ACM.

[26] Bianchi, G. 1998. IEEE 802.11-saturation throughput analysis. *Communications Letters, IEEE* 2(12):318–320.

[27] Böhmer, Matthias, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. 2011. Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, 47–56. MobileHCI '11, New York, NY, USA: ACM.

[28] Broadband Forum. TR-069 Amendment 5. http://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf.

[29] Broustis, Ioannis, Konstantina Papagiannaki, Srikanth V. Krishnamurthy, Michalis Faloutsos, and Vivek Mhatre. 2007. MDG: Measurement-driven Guidelines for 802.11 WLAN Design. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, 254–265. MobiCom '07, New York, NY, USA: ACM.

[30] Calit2. InterDigital Innovation Challenge 2012 Winners. http://calit2.net/newsroom/release.php?id=2065.

[31] Canadi, Igor, Paul Barford, and Joel Sommers. 2012. Revisiting Broadband Performance. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, 273–286. IMC '12, New York, NY, USA: ACM.

[32] Chambers, Chris, Wu-chang Feng, Sambit Sahu, and Debanjan Saha. 2005. Measurement-based Characterization of a Collection of On-line Games. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 1–1. IMC '05, Berkeley, CA, USA: USENIX Association.

[33] Chambers, Chris, Wu-Chang Feng, Sambit Sahu, Debanjan Saha, and David Brandt. 2010. Characterizing online games. *IEEE/ACM Trans. Netw.* 18(3):899–910.

[34] Cheng, Yu-Chung, 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*, 39–50. SIGCOMM '06, New York, NY, USA: ACM.

[35] Cisco. Global Mobile Data Traffic Forecast Update 2014-2019 White Paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[36] Cisco. Wireless LAN Controller. http://www.cisco.com/c/en/us/products/wireless/wireless-lan-controller/index.html.

[37] D-Link. Cloud Router. http://www.dlink-cloud.com/solutions.aspx.

[38] DiCioccio, Lucas, Renata Teixeira, and Catherine Rosenberg. 2010. Impact of Home Networks on End-to-end Performance: Controlled Experiments. In *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, 7–12. HomeNets '10, New York, NY, USA: ACM.

[39] Do, Trinh Minh Tri, Jan Blom, and Daniel Gatica-Perez. 2011. Smartphone Usage in the Wild: A Large-scale Analysis of Applications and Context. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, 353–360. ICMI '11, New York, NY, USA: ACM.

[40] Engadget. Netflix reveals Android app tests that keep it running on 'around 1000' devices daily. http://www.engadget.com/2012/03/15/netflix-android-app-testing-process/.

[41] Falaki, Hossein, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. 2010. Diversity in Smartphone Usage. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 179–194. MobiSys '10, New York, NY, USA: ACM.

[42] Feng, Wu-chang, David Brandt, and Debanjan Saha. 2007. A Long-term Study of a Popular MMORPG. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*, 19–24. NetGames '07, New York, NY, USA: ACM.

[43] Giustiniano, Domenico, Eduard Goma, Alberto Lopez Toledo, Ian Dangerfield, Julian Morillo, and Pablo Rodriguez. 2010. Fair WLAN Backhaul Aggregation. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, 269–280. MobiCom '10, New York, NY, USA: ACM.

[44] Gkantsidis, Christos, Thomas Karagiannis, Peter Key, Bozidar Radunovic, Elias Raftopoulos, and D. Manjunath. 2009. Traffic Management and Resource Allocation in Small Wired/Wireless Networks. In *Proceedings*

of the 5th International Conference on Emerging Networking Experiments and Technologies, 265–276. CoNEXT '09, New York, NY, USA: ACM.

[45] Gollakota, Shyamnath, Fadel Adib, Dina Katabi, and Srinivasan Seshan. 2011. Clearing the RF Smog: Making 802.11N Robust to Cross-technology Interference. In *Proceedings of the ACM SIGCOMM 2011 Conference*, 170–181. SIGCOMM '11, New York, NY, USA: ACM.

[46] Google. Android Developers. http://developer.android.com/index.html.

[47] Google. For The Power User. https://play.google.com/store/apps/collection/promotion_3000109_geek_tools.

[48] HD Guru. Do Wireless HDTV Systems Do The Job? http://hdguru.com/do-wireless-hdtv-systems-do-the-job-lastest-models-reviewed/6178/.

[49] Huang, Junxian, Qiang Xu, Birjodh Tiwana, Z. Morley Mao, Ming Zhang, and Paramvir Bahl. 2010. Anatomizing Application Performance Differences on Smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 165–178. MobiSys '10, New York, NY, USA: ACM.

[50] ICSI. Netalyzr. http://netalyzr.icsi.berkeley.edu/.

[51] IETF. . CAPWAP Protocol Specification. http://tools.ietf.org/search/rfc5415.

[52] IETF. . netconf. http://datatracker.ietf.org/wg/netconf/charter/.

[53] Jardosh, Amit P., Krishna N. Ramachandran, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. 2005. Understanding Congestion in IEEE 802.11b Wireless Networks. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 25–25. IMC '05, Berkeley, CA, USA: USENIX Association.

[54] Kandula, Srikanth, Kate Ching-Ju Lin, Tural Badirkhanli, and Dina Katabi. 2008. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 89–104. NSDI'08, Berkeley, CA, USA: USENIX Association.

[55] Kashyap, Anand, Samrat Ganguly, and Samir R. Das. 2007. A Measurement-based Approach to Modeling Link Capacity in 802.11-based Wireless Networks. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, 242–253. MobiCom '07, New York, NY, USA: ACM.

[56] Kotz, David, and Kobby Essien. 2002. Analysis of a Campus-wide Wireless Network. In *In Proceedings of ACM Mobicom*, 107–118. ACM Press.

[57] Lakshminarayanan, Kaushik, Samir Sapra, Srinivasan Seshan, and Peter Steenkiste. 2009. RFDump: An Architecture for Monitoring the Wireless Ether. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, 253–264. CoNEXT '09, New York, NY, USA: ACM.

[58] Leiva, Luis, Matthias Böhmer, Sven Gehring, and Antonio Krüger. 2012. Back to the App: The Costs of Mobile Application Interruptions. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, 291–294. MobileHCI '12, New York, NY, USA: ACM.

[59] Liu, He, Stefan Saroiu, Alec Wolman, and Himanshu Raj. 2012. Software Abstractions for Trusted Sensors. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 365–378. MobiSys '12, New York, NY, USA: ACM.

[60] Mahajan, Ratul, 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, 75–86. SIGCOMM '06, New York, NY, USA: ACM.

[61] Manweiler, J., P. Franklin, and R.R. Choudhury. 2012. RxIP: Monitoring the Health of Home Wireless Networks. In *INFOCOM, 2012 proceedings IEEE*, 558–566.

[62] Manweiler, Justin, Sharad Agarwal, Ming Zhang, Romit Roy Choudhury, and Paramvir Bahl. 2011. Switchboard: A matchmaking system for multiplayer mobile games. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, 71–84. MobiSys '11, New York, NY, USA: ACM.

[63] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* 38(2):69–74.

[64] Meraki. Enterprise Cloud Management. http://www.meraki.com/products/wireless/enterprise-cloud-management.

[65] Microsoft. Windows Phone. http://www.windowsphone.com/en-us/store.

[66] Mirza, Mariyam, Kevin Springborn, Suman Banerjee, Paul Barford, Michael Blodgett, and Xiaojin Zhu. 2009. On the Accuracy of TCP Throughput Prediction for Opportunistic Wireless Networks. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 1–9. SECON'09, Piscataway, NJ, USA: IEEE Press.

[67] Mishra, A., V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. 2006. A Client-Driven Approach for Channel Management in Wireless LANs. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 1–12.

[68] Moore, Andrew W., and Konstantina Papagiannaki. 2005. Toward the Accurate Identification of Network Applications. In *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, 41–54. PAM'05, Berlin, Heidelberg: Springer-Verlag.

[69] Murty, Rohan, Jitendra Padhye, Alec Wolman, and Matt Welsh. 2010. Dyson: An Architecture for Extensible Wireless LANs. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, 15–15. USENIXATC'10, Berkeley, CA, USA: USENIX Association.

[70] Nazir, Atif, Saqib Raza, and Chen-Nee Chuah. 2008. Unveiling Facebook: A Measurement Study of Social Network Based Applications. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, 43–56. IMC '08, New York, NY, USA: ACM.

[71] Nazir, Atif, Saqib Raza, Dhruv Gupta, Chen-Nee Chuah, and Balachander Krishnamurthy. 2009. Network Level Footprints of Facebook Applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, 63–75. IMC '09, New York, NY, USA: ACM.

[72] OOKLA. Speedtest. http://www.speedtest.net/.

[73] OSMF. Strobe Media Playback. http://osmf.org/strobe_mediaplayback.html.

[74] Papagiannaki, K., M. Yarvis, and W.S. Conner. 2006. Experimental Characterization of Home Wireless Networks and Design Implications. In *INFOCOM 2006. 25th IEEE international conference on computer communications. proceedings*, 1–13.

[75] PC Engines. alix2d2. http://www.pcengines.ch/alix2d2.htm.

[76] Pentaho. REPTree. http://wiki.pentaho.com/display/DATAMINING/REPTree.

[77] PerBlue. Parallel Kingdom. http://www.parallelkingdom.com/.

[78] Pittman, Daniel, and Chris GauthierDickey. 2007. A Measurement Study of Virtual Populations in Massively Multiplayer Online Games. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*, 25–30. NetGames '07, New York, NY, USA: ACM.

[79] Pittman, Daniel, and Chris GauthierDickey. 2010. Characterizing Virtual Populations in Massively Multiplayer Online Role-Playing Games. In *Proceedings of the 16th International Conference on Advances in Multimedia Modeling*, 87–97. MMM'10, Berlin, Heidelberg: Springer-Verlag.

[80] Pocket PC FAQ. Cellular Data Primer. http://www.pocketpcfaq.com/faqs/cellularprimer.htm/.

[81] Rahmati, Ahmad, Chad Tossell, Clayton Shepard, Philip Kortum, and Lin Zhong. 2012. Exploring iPhone Usage: The Influence of Socioeconomic Differences on Smartphone Adoption, Usage and Usability. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, 11–20. MobileHCI '12, New York, NY, USA: ACM.

[82] Rao, Ashwin, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous. 2011. Network Characteristics of Video Streaming Traffic. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, 25:1–25:12. CoNEXT '11, New York, NY, USA: ACM.

[83] Ravindranath, Lenin, Jitendra Padhye, Sharad Agarwal, Ratul Mahajan, Ian Obermiller, and Shahin Shayandeh. 2012. AppInsight: Mobile App Performance Monitoring in the Wild. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, 107–120. OSDI'12, Berkeley, CA, USA: USENIX Association.

[84] Rayanchu, Shravan, Ashish Patro, and Suman Banerjee. 2011. Airshark: Detecting non-WiFi RF Devices Using Commodity WiFi Hardware. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, 137–154. IMC '11, New York, NY, USA: ACM.

[85] Rayanchu, Shravan, Ashish Patro, and Suman Banerjee. 2012. Catching Whales and Minnows Using WiFiNet: Deconstructing non-WiFi Interference Using WiFi Hardware. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 5–5. NSDI'12, Berkeley, CA, USA: USENIX Association.

[86] Reis, Charles, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. 2006. Measurement-based Models of Delivery and Interference in Static Wireless Networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 51–62. SIGCOMM '06, New York, NY, USA: ACM.

[87] Shin, Choonsung, Jin-Hyuk Hong, and Anind K. Dey. 2012. Understanding and Prediction of Mobile Application Usage for Smart Phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 173–182. UbiComp '12, New York, NY, USA: ACM.

[88] Shrivastava, Vivek, Nabeel Ahmed, Shravan Rayanchu, Suman Banerjee, Srinivasan Keshav, Konstantina Papagiannaki, and Arunesh Mishra. 2009. 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*, 297–308. MobiCom '09, New York, NY, USA: ACM.

[89] Shrivastava, Vivek, 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*, 337–350. NSDI'11, Berkeley, CA, USA: USENIX Association.

[90] Shrivastava, Vivek, Shravan Rayanchu, Jongwoon Yoonj, and Suman Banerjee. 2008. 802.11n Under the Microscope. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, 105–110. IMC '08, New York, NY, USA: ACM.

[91] Sivaraman, Vijay, Tim Moors, Hassan Habibi Gharakheili, Dennis Ong, John Matthews, and Craig Russell. 2013. Virtualizing the Access Network via Open APIs. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, 31–42. CoNEXT '13, New York, NY, USA: ACM.

[92] Sommers, Joel, and Paul Barford. 2012. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, 301–314. IMC '12, New York, NY, USA: ACM.

[93] StudyBlue. StudyBlue Flashcards & Quizzes. http://www.studyblue.com/.

[94] Sundaresan, Srikanth, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. 2011. Broadband Internet Performance: A View from the Gateway. In *Proceedings of the ACM SIGCOMM 2011 Conference*, 134–145. SIGCOMM '11, New York, NY, USA: ACM.

[95] Suresh, Lalith, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. 2012. Towards programmable enterprise WLANs with Odin. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 115–120. HotSDN '12, New York, NY, USA: ACM.

[96] Tan, Godfrey, and John Guttag. 2004. Time-based Fairness Improves Performance in Multi-rate WLANs. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, 23–23. ATEC '04, Berkeley, CA, USA: USENIX Association.

[97] Tarng, Pin-Yun, Kuan-Ta Chen, and Polly Huang. 2008. An Analysis of WoW Players' Game Hours. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, 47–52. NetGames '08, New York, NY, USA: ACM.

[98] VoIP Test. OnSIP. http://www.onsip.com/tools/voip-test.

[99] Wee, Tan Kiat, and Rajesh Krishna Balan. 2012. Adaptive Display Power Management for OLED Displays. In *Proceedings of the First ACM International Workshop on Mobile Gaming*, 25–30. MobileGames '12, New York, NY, USA: ACM.

[100] Xu, Qiang, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying Diverse Usage Behaviors of Smartphone Apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, 329–344. IMC '11, New York, NY, USA: ACM.

[101] Yap, Kok-Kiong, Masayoshi Kobayashi, David Underhill, Srinivasan Seetharaman, Peyman Kazemian, and Nick McKeown. 2009. The Stanford OpenRoads Deployment. In *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*, 59–66. WINTECH '09, New York, NY, USA: ACM.

[102] Yiakoumis, Yiannis, Kok-Kiong Yap, Sachin Katti, Guru Parulkar, and Nick McKeown. 2011. Slicing Home Networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks*, 1–6. HomeNets '11, New York, NY, USA: ACM.

[103] Zhuang, Zhenyun, Kyu-Han Kim, and Jatinder Pal Singh. 2010. Improving Energy Efficiency of Location Sensing on Smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 315–330. MobiSys '10, New York, NY, USA: ACM.