Understanding the Role of Active Scans for their Better Utilization in Large-scale WiFi Networks

By

Dheryta Jaisinghani

Under the supervision of Dr. Vinayak Naik and Dr. Sanjit Kaul

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

April, 2019

Understanding the Role of Active Scans for their Better Utilization in Large-scale WiFi Networks

By

Dheryta Jaisinghani

A Thesis
submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Computer Science and Engineering
Indraprastha Institute of Information Technology Delhi
New Delhi– 110020

April, 2019

Certificate

This is to certify that the thesis titled *Understanding the Role of Active Scans for their Better Utilization in Large-scale WiFi Networks* being submitted by *Dheryta Jaisinghani* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

April, 2019
Dr. Vinayak Naik
Dr. Sanjit Kaul

Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Working Code Trumps All Hype
- Dr. Mahadev Satyanarayanan

Abstract

A WiFi client needs to discover access points (APs) in its vicinity before it can establish a WiFi connection and initiate a data transfer. Discovery is also essential for a client to stay connected to an AP deemed best at any given time. Discovery mechanisms in WiFi can be categorized into that of passive and active scanning. Passive scanning involves a client listening to beacon frames across WiFi channels to know which access points are in its vicinity. Using this method a client doesn't create any additional load on the network. However, the latency in discovering new access points, especially in low-density AP deployments, can be significant and impacts the performance of latency-sensitive applications like VoIP. The alternative of active scanning involves a client proactively broadcast probe requests, which are responded to with a probe response by access points that receive it. The proactive sending of probes reduces latency in finding access points and is often used by clients.

While simple and effective, we discover that this seemingly innocuous procedure of probing can cause severe degradation of throughput in an enterprise WiFi network, such as that of Cisco and Aruba. This motivates the premise of this thesis that management procedures in WiFi, which include discovery and association, need a thorough relook. They must be adapted to WiFi networks of today that have high densities of access points and clients unforeseen when the mechanisms were envisaged. Not only has the density of clients increased but also the heterogeneity of their service requirements. While some, for example streaming video, may desire high throughputs for extended periods of time, others, for example, temperature sensors, may only wake up intermittently to send small bursts of data. Last but not the least, mechanisms like probing are no longer used just for AP discovery but also support WiFi services like localization. Changes to mechanisms may require interventions that alleviate the impact on such services.

We begin by investigating the protocol of active scanning from the perspec-

tive of present-day large-scale WiFi networks. With several real-world network traces, we analyze the extent of the impact of active scanning. We demonstrate empirically that WiFi clients, typically in 2.4 GHz frequency band, often trigger active scans without a clear need for the same. As a consequence, they inject excessive low bit-rate traffic in the network that exponentially brings down the goodput of the network. We develop a metric that measures the growth of this traffic and an inference mechanism that detects the cause of this growth. We also suggest measures to mitigate these causes. While dealing with such unnecessary scans is desirable, we observe that a reduced frequency of active scans can have unintended consequences on services leveraging WiFi. Specifically, we study WiFi-based indoor localization that presumes that clients always scan. However, WiFi networks primarily operating in 5 GHz often experience a low rate of scanning and this severely deteriorates the performance of localization services. We demonstrate that an efficient floor detection mechanism can significantly reduce such localization errors. Lastly, we argue that for transferring small amounts of data such as infrequent sensor readings from IoT nodes, a WiFi client must not need to go through the entire process of connection establishment and maintenance. Here, we propose to leverage the perpetual WiFi active scans to enable such data transfer. The proposed approach of data transfer is not only beneficial in WiFi networks where establishing and maintaining a WiFi association is hard due to a variety of reasons, but also saves battery and reduces network traffic significantly.

In our work, we will restrict our proposals to being device agnostic. Specifically, we don't expect changes to be made at the device end. This allows for the possibility of their faster adoption in enterprise networks. Finally, we validate all our proposed solutions on real devices in live WiFi networks and demonstrate how these solutions enhance the performance of present-day WiFi networks.

Dedication

To,
My parents who brought me in this world,
Late. Mr. C. P. Jaisinghani and Mrs. Geeta Jaisinghani
The one whom I brought in this world,
Baby Ashvika Karwal
The one who has been with me all the time, my better half,
Mr. Gaurav Karwal
My in-laws without whom this would not have been possible,
Mr. Devendra Karwal and *Dr. Indra Karwal*
My loving sister,
Mrs. Chandrita Jaisinghani
And her husband,
Mr. (Soon to be Dr.) Saurabh Karwal.

Acknowledgements

Karmanye Vadhikaraste, Ma phaleshou kada chana, Ma Karma Phala Hetur Bhurmatey Sangostva Akarmani - Bhagwad Gita -
Perform your work with perseverance, without worrying about the outcome.

I met several wonderful people in the past six years who have contributed significantly, either technically or emotionally or both, towards this thesis. Today, I am fortunate enough to extend my heartfelt thanks to all those people.

My challenging yet exciting and enriching journey of research started when I contacted my advisor, Prof. Vinayak Naik, in September 2012 post my M.Tech while I was looking out for potential Ph.D. positions. I did not imagine that six years later, I could not thank my stars enough that I made the right decision of joining under his guidance. I am much indebted to him for his amazing support and constant motivation throughout my Ph.D. tenure. Somehow, he always knew the right choice at the right time for me, and it always worked, magically. I still remember his words – *I have never seen people failing, who are consistently working towards that end goal*. I made his words my motto that day. As an advisor, he always set the right learning curve for me – be it speaking, writing, presentation, or, the most important, research skills. Not only did he guide me for my research work, but he also helped me realize the importance of giving back to our society with tutorials and workshops. With his support, I conducted many such sessions to help students learn the topics which are not readily available in books or the Internet. One of such example is my talk on *Opening Nuts and Bolts of Linux WiFi Subsystem*, which has maximum views, $1800+$, from all technical topics on IIIT-Delhi's official YouTube Channel and $4,000+$ downloads on SlideShare. I cherish the feeling of content when I can bridge a learning gap for struggling students. I am very fortunate that an advisor who values such contributions advised me. Even otherwise, Prof. Vinayak has been very supportive of me given my family challenges. My daughter was born

during the second year of my Ph.D., and the already tough path of research got even tougher for me. However, Prof. Vinayak was with me all through and helped me cope with my pressures.

I am likewise extremely thankful for my co-advisor, Prof. Sanjit Kaul. I have never met an advisor like him before, who is always ready to sit with the student for hours irrespective of time and day, to solve that one problem. He helped me polish my technical writing skills to a great extent. I have had a fantastic experience learning from his immense knowledge about wireless networks. He had been an excellent guide and mentor throughout my graduate journey. My Ph.D. research would not have been possible without the insightful comments from my monitoring committee, that consisted of Prof. Sambudhho Chakraborty (IIIT-Delhi, India) and Prof. Vivek Bohara (IIIT-Delhi, India). Their expert comments and feedback helped me improve my work to the next level every year. Here, my heartfelt thanks to Prof. Mythili Vutukuru (IIT-Bombay, India) who was the external examiner during my comprehensive examination. She gave me invaluable advice on the part of work that makes 40% of this thesis.

I have been blessed to have worked and guided by some of the eminent researchers in the field. I want to express my sincere thanks to Prof. Sumit Roy (University of Washington, USA). He brought with him an immense knowledge about the field as well as the Ph.D. tenure. He has mentored me at various stages of my journey. Without his guidance, I might have discontinued work on one of the significant contributions of this thesis. All his suggestions were invaluable, and I am indebted to him. I am profoundly grateful to Prof. Rajesh Balan (Singapore Management University, Singapore) for his selfless help. I met Prof. Rajesh when I was interning at LiveLabs during one of the toughest times of my life when my father was struggling with critical health ailments. In that helpless situation, his constant motivational words kept me going. Prof. Rajesh, despite being extremely busy will take out time for his students to keep their spirits high. I still remember the day, when I could not focus on work, he was so kind that he took me out for a walk and gave some invaluable life lessons. Many thanks to him for his mentoring. I want to thank Prof. Archan Misra (Singapore Management University, Singapore) to have helped me at various stages of my career. Prof. Archan is such a generous person and an excellent guide. I remember first writing to him while I was looking for Ph.D. positions. From then in 2011 to now in 2018, despite his hectic schedules he has always been there to mentor me. Heartfelt thanks to Prof. Youngki Lee (Seoul National University, South

Korea) without whose mentorship the work on localization in this thesis would not have been complete.

I have also had some phenomenal graduate and undergraduate student collaborators – Jagrati Gogia, Nishtha Phutela, Akanksha Pandey, Gursimran Singh, and Harish Fulara. They have had been the part of prototype development, several failed and successful experiments, and data analysis. As a core systems thesis, without their support, I doubt if I would have been able to do this work. I thank them wholeheartedly for being with me through this process. I am profoundly thankful to a fantastic friend, mentor, and collaborator, Prof. Mukulika Maity (IIT-Bombay, IIIT-Delhi). We discussed various problems and struggles of graduate life, personal life, have had brainstorming sessions on solving issues of developing nations, doing experiments, and targetting good conferences. I wish and hope we always stay like this and work together for many more years in the future.

My labmates and friends from IIIT-Delhi had been critical pillars throughout the ups and downs of my Ph.D. They were my stress busters and my constant source of energy to keep me going. I lovingly thank my closest friend, Alvika Gautam, for everything that she did for me in this journey, Haroon Rashid, for the endless discussions on our future career plans, and Anupriya Tuli for her rejuvenating and inspiring conversations. I thank all members of Mobile and Ubiquitous Computing research group – Anil Jain, Deepika Yadav, Garvita Bajaj, Kuldeep Yadav, Madhur Hasija, Madhvi Gupta, Manoj Gulati, Milan Jain, Nipun Batra, Pandarasamy Arjunan, Parikshit Maini, Sneihil Gopal, Siddhartha Asthana, Sonia Soubam, and Tanya Shreedhar. I also thank my friends from other labs, who have been equally supportive – Parag Aggarwal, Devashish Gosain, and Piyush Kumar Sharma.

I would also express my gratitude towards the always working administration staff of IIIT-Delhi, especially, Ms. Sheetu Ahuja and Ms. Priti Patel. They have always been kind and generous enough to address all my queries and making the official formalities easy for Ph.D. candidates. Their efforts are well acknowledged, and I am very grateful to them for all their help and support.

My stint of research started with the M.Tech thesis which I completed under the competent guidance of Prof. P. G. Poonacha (IIIT-Bangalore, India). I want to thank him for introducing research to me. I would also like to thank Prof. Debabrata Das (IIIT-Bangalore, India) and Dr. Kumar Padmanabh (Etisalat

British Telecom Innovation Center, UAE) for advising and examining my first research thesis. I want to express my heartfelt gratitude to my school teacher Ms. Vandita Munjal (D.A.V Public School, India), who recognized my potential of being a computer science student rather than a medical student. It is because of her efforts, that I took computer science as one of my majors in high school and realized my love for the subject.

Lastly, but most importantly, I don't have enough words to thank my incredibly supportive family. I am blessed to have a family who stood to me at all the highs and lows of my Ph.D. To begin with, my daughter – Baby Ashvika Karwal who is just 4.5 years old. She is the youngest of all, but the most supportive one. Her highly accommodating nature never stops to amuse me. She has been my pillar of strength when she was as young as 4 months old – when I got my first paper published. My daughter does not have one mother she has 4. In my absence, everyone makes her feel comfortable. My husband is one of the most beautiful people, who can fight any challenge to fulfill my wish, and he proved that uncountable number of times during all these years. I cannot thank enough my mother-in-law for all her contributions and sacrifices that she made for me and my Ph.D. She sets the perfect example of how to fight the world to get your dream fulfilled, and she did that for me. My father-in-law has been equally supportive to have taken care of my daughter when I was away attending conferences and doing late-night experiments in the lab. Today, I miss my father's (who passed away early this year) constant words of inspiration. If not anyone, he always believed in me and always told me – *you have to try because you cannot even fail if you don't try.* My mother, who made countless sacrifices, my entire life to make me what I am today. I thank my sister and my brother-in-law who always understood my pressures and mentored me like any elder do, to keep you going. I dedicate this work to my nurturing family, who enabled the perfect ecosystem for me to flourish.

Towards the end, I thank the diligent external examiners of the thesis, Prof. Karthik Dantu (University of Buffalo, USA), Prof. Nirupam Roy (University of Maryland, USA), and Prof. Srihari Nelakuditi (University of South Carolina, USA), whose comprehensive comments helped me improve the quality of the thesis. I appreciate and acknowledge IIIT-Delhi and the ITRA project funded by DEITy Government of India under a grant with Ref. No. ITRA/15(57)/Mobile/Human-Sense/01 for the financial support given to pursue the research presented in this thesis.

Previously Published Material

- Chapter 2 revises previous publications [1, 2]
- Chapter 3 revises previous publications [1, 3, 4, 5]
- Chapter 5 revises previous publications [2, 6]
- Chapter 4 revises a previous publication [7]

Contents

List of Tables

List of Figures

List of Abbreviations

ACK Acknowledgment

AP Access Point

BSS Basic Service Set

BSSID Basic Service Set Identifier

CMAS Causal Model of Active Scanning

CSV Comma Separated Values

IoT Internet-of-Things

PCAP Packet Capture

QoS Quality-of-Service

RRM Radio Resource Management

RSSI Received Signal Strength Indicator

RTLS Real-Time Location Service

RTT Round Trip Time

SSID Service Set Identifier

TBTT Target Beacon Transmission Time

VLAN Virtual Local Area Network

WIPS Wireless Intrusion Prevention System

WLAN Wireless Local Area Network

Chapter 1

Introduction

WiFi standards, `IEEE 802.11` [8], were created to ensure coverage. The traditional sparse deployments of access points (AP) made it essential to design mechanisms by which clients, especially mobile clients with VoIP like applications, could discover APs in their vicinity with minimal delays. This motivated the methods of AP discovery, specifically the active scanning methods. These methods worked well in low-density AP scenarios. The deployments of today, however, have a relatively very high density of APs. For example, WiFi services are already available at most places, be it a shopping mall, an island, railway stations, airports, universities, and corporate offices [9, 10, 11, 12, 13, 14, 15].

Unlike the traditional WiFi networks that merely provided basic network connectivity to the WiFi clients, modern services such as Internet-of-Things [16], Environment Sensing [17], and Indoor Localization [18], leverage a WiFi network deployment for heterogeneous Quality-of-Service (QoS) requirements. However, the mechanisms of AP discovery stay unchanged, where WiFi clients

broadcast low bit-rate probe traffic (probe requests and probe responses) during active scans or wait for beacons from AP during passive scans.

In this thesis, we investigate these mechanisms of discovery that are often taken for granted in the context of WiFi networks of today, that have high densities of APs and can span over large areas such as, university campuses and cities. Active scanning is essential, but in its current form may become detrimental to network performance. On the other hand, arbitrary reduction in active scanning negatively impacts the performance of a network in terms of increased handover delays and errors in services like localization that rely on probe traffic.

We collect and analyze data from two different real-world WiFi networks, operating in India and Singapore. These networks differ in terms of vendors – Cisco and Aruba, and clients – dual-band and single-band. With the data collected, we first demonstrate empirically how the existing mechanism of active scanning leads to unnecessary probe traffic in the network. Having demonstrated the seriousness of the problem, we propose methods to automatically diagnose the causes behind active scans, followed by suggesting ways that mitigate the causes. While reducing scanning is often desirable, we discover that reduced frequency can have unintended consequences. Specifically, we study the performance of WiFi-based indoor localization in an environment where clients scan less frequently. Last but not the least, given that active scans are triggered often enough by every client, and given that in the near future many such clients will be Internet-of-Things (IoT) devices with asynchronous short data requests, we show how scans can, in fact, be used to transmit data; thereby

making association unnecessary and thus reducing the load on the network.

We, now introduce the three problems addressed and the methodology followed in this thesis in the next sections.

1.1 Understanding and Mitigating the Impact of Unnecessary Active Scans in WiFi Networks

We find that stationary WiFi clients trigger active scans without considering a need of it. As a result, they inject low bit-rate probe requests and probe responses in the network. Usually, these probe requests and probe responses do not cause any significant throughput issues as they comprise a very small percentage, $< 2\%$ across all channels, of the overall network management traffic. However, we have found that in the heavily utilized 2.4 GHz WiFi networks, where channels are at least 50% utilized, the amount of probe traffic grows up to 50%, at least, of the total management traffic in just a second. This, in turn, brings down the goodput exponentially. The industry and the research community later acknowledged the problem as well [19, 20].

The reason for this dramatic drop in goodput is because probe traffic, as per the WiFi specifications to ensure the highest delivery probability is sent at full power, at the *lowest bit-rate*, and on most channels. This low bit-rate increases their transmission time and during this time, other clients are unable to send or receive data on the shared wireless spectrum. Thus, if these slow frames start increasing beyond a certain point, the overall efficiency of the network begins to

degrade as the clients and APs are unable to find free spectrum slots to send and receive required data. 5 GHz network is mostly unaffected by this phenomena, as there are multiple non-overlapping channels in this frequency band to isolate every AP from its neighbors.

A possible solution to deal with unnecessary active scans is to modify the scanning behavior of WiFi devices *i.e.*, APs and clients at various software and hardware layers of the protocol stack. Either of the following realizes this – not allowing devices to trigger active scans, implementing modified scanning algorithms in devices[21, 22, 23], or by not allowing APs to respond to probe requests. These solutions, however, have scalability issues as the clients come from different vendors and may use different WiFi driver implementations. Furthermore, the lack of standardization of when active scanning may be triggered makes it impractical to make such changes in existing clients. Besides the client changes, the changes at the AP is also a vendor-specific decision that is taken at the algorithms executing in the WiFi controller. Even if active scanning is somehow disabled, the other option is passive scanning, which is time-consuming and hence not preferred [24, 25, 26, 27]. Therefore, we suggest specific measures to control the probe traffic from three different perspectives – the proper configuration of the APs, efficient network planning, and right configuration of the clients.

We understand that most of the APs that could hear a probe request, including the ones sent on overlapping channels, do respond with probe responses. The number of probe responses generated further increase with the number of

BSSIDs (Basic Service Set Identifiers) configured per AP. These factors result in the generation of multiple probe responses for a single probe request sent. Moreover, the information transmitted in these responses is redundant, *i.e.*, it does not frequently change. As part of measures under AP configuration, we propose that the number of operational BSSIDs should be kept as less as possible. In the absence of legacy clients, disable low rate responses. Finally, whenever feasible, enforce 5 GHz operation for the WiFi network that can be enabled with band steering. These decisions require simple configuration changes at the WiFi controller that can be quickly done by the network-admins.

However, despite suggested by the WiFi network vendors, for instance, Cisco and Aruba, not all enterprise networks follow the guidelines to reduce probe responses. To exemplify, they don't configure a threshold for the number of probe requests that an AP should respond to, they don't disable probe responses to rogue and unauthorized clients. Moreover, the suggestions mentioned above are not reliable solutions to control the probe traffic; primarily because they are capable of reducing probe responses in a WiFi network but not probe requests. Furthermore, reducing the number of operational BSSIDs and enabling 5 GHz operation may pose administrative challenges.

Instead, we suggest an alternative approach of implicitly eliminating the probe traffic by *alleviating the cause* that triggers a client to scan. The approach is based on the fact that if transmission of the probe requests is curbed, probe responses will get eliminated implicitly. Our approach is to – (a) detect if probe traffic is the cause of performance drop, (b) if it is, then infer the cause of growth

of the probe traffic and improve the planning of the WiFi network to mitigate the active scanning causes, and/or (c) control the amount of probe requests generated at the client-end itself. We discuss how to apply the inference mechanism to the present day WiFi networks. Note that, unlike the state-of-the-art client-end solutions, our solution does not need any driver or hardware change at the client. Instead, our solution can be rolled out to clients as a simple application update.

1.2 WiFi Indoor Localization using Existing Infrastructure in the presence of Minimal Active Scans

WiFi indoor localization is a fundamental service that is needed by most present-day contextual applications [18]. Most contextual services today, including the recent IoT services, need an efficient, robust, and device-agnostic mechanisms to localize devices. Localization has a rich history of wonderful solutions [28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]. However, in spite of several breakthroughs, there are very few real-world deployments of WiFi-based indoor localization systems in public spaces. The reasons for this are many-fold, with three of the most common being – (a) the high cost of deployment, (b) arguably, the lack of compelling business use, and (c) the inability of existing solutions to seamlessly work with all devices. In fact, current solutions impose a tradeoff between universality, accuracy, and energy, for example, client-based solutions that combine inertial-based tracking with WiFi scanning

offer significantly better accuracy but require a mobile application which will possibly drain energy faster and which will be downloaded by only a fraction of visitors [42].

We present our experiences with deploying and operating a WiFi-based indoor localization system across the entire campus of a small Asian university. It is worth noting that the environment is very densely occupied, by $\approx 10,000$ students and $1,500$ faculty and staff. The system has been in production for more than four years. It is deployed at multiple venues including two universities (Singapore Management University, University of Massachusetts, Amherst), and four different public spaces (Mall, Convention Center, Airport, and Sentosa Resort) [18, 43]. These venues use the localization system for various real-time analytics such as group detection, occupancy detection, and queue detection while taking care of user privacy.

We highlight challenges and propose easy to integrate solutions to build a universal indoor localization system – *one that can localize, on-the-spot, all WiFi-enabled devices on campus without any modifications, whether client or infrastructure-side.* The scale and the nature of this real environment, presents a unique set of challenges – (a) infrastructure *i.e.* controller and APs do not allow any changes, (b) devices cannot be modified in any way *i.e.* no explicit/implicit participation for data generation, no app download allowed, and no chipset changes allowed, and (c) only available data is the Received Signal Strength Indicator (RSSI) measurements from APs, which are centrally controlled by the controller, using a *Real-Time Location Service* (RTLS) interface [44]. It is worth

noting that within the face of these challenges we have to rule out more sophisticated state-of-the-art schemes, such as fine-grained CSI measurements [33], Angle-of-Arrival [45], Time-of-Flight [37], SignalSLAM [46], or Inertial Sensing [47].

Given the challenges, we adopt an offline fingerprint-based approach to compute each device's location. Fingerprints have been demonstrated to be more accurate than model-based approaches in densely crowded spaces [48] and hence widely preferred. Our localization software processes the RSSI updates using well-known "classical" fingerprint-based technique [28]. Given the wide usage of this approach, our experiences and results apply to a majority of the localization algorithms.

Our primary contribution is to detail the cases where such a conventional approach succeeds and the cases where it fails. We highlight the related challenges for making the approach work in modern WiFi networks, and then develop appropriate solutions to overcome the observed challenges. We collect three weeks of detailed ground truth data (≈ 200 landmarks) in our large-scale deployment, that is representative of our four years of data. With the data collected, we carefully construct a set of experimental studies to show two unique challenges – *Cardinality Mismatch* and *High Client Scan Latency* associated with a server-side localization approach. We explain the challenges follows –

(a) *Cardinality Mismatch:* We define cardinality as the set of APs reporting for a client located at a specific landmark. We first show that the cardinality,

during the online phase, is *often* quite different from the cardinality in the offline phase. Note that this divergence is in the *set* of reporting APs, and not just merely a mismatch in the values of the RSSI vectors. The extent of this mismatch is not only surprising but challenging to debug and mitigate. Intuitively, this upends the very premise of fingerprint-based systems that the cardinality seen at any landmark is the same during the offline and online phases. This phenomenon arises from the dynamic power and client management performed by a centralized controller in all enterprise WiFi networks to achieve outcomes such as – (i) minimize overall interference by shifting the neighboring APs to alternative channels, (ii) enhance throughput by shifting the clients to alternative APs, and (iii) reduce energy consumption by shutting down redundant APs during periods of low load.

(b) *High Client Scan Latency:* Most localization systems use client-side localization techniques where clients actively scan the network when they need a location fix. However, when using server-side localization, the location system has no way to induce scans from client devices. Hence, the system can only "see" clients when clients scan as part of their normal behavior. However, we show that the scanning frequency of clients is low for lower RSSI.

These phenomena do not exist in small-scale deployments often used in the past pilot studies, where each AP is configured independently. In large-scale deployments, where it is fairly common to use controller-managed WiFi networks with a large number of devices, these phenomena invariably persist to a great extent. To exemplify, we noticed 57.30% instances of cardinality mismatch in 2.4

GHz and 30.60% in 5 GHz in our deployment. We saw $90^{th}\%$ile of client scan interval to be 20 minutes. While localizing with fingerprint-based solutions in such environments, these phenomena translate to either *minimal* or even worse *no* matching APs, resulting in substantial delays between client location updates and "teleporting" of clients across the location.

It is important to note that not only the schedule of these algorithms is non-deterministic but also their distribution during offline and online phases. This is attributed to the fundamental fact that the dynamics of WiFi networks such as load and interference, is non-deterministic in most of the cases and that the controller algorithm is a black-box to us. Furthermore, the differences in signal propagation and scanning behavior of 2.4 GHz and 5 GHz contribute to these problems. We believe that we are the first to present the challenges of server-side localization as well as their mitigation. Our proposals are device-agnostic, simple, and easily integrable with any large-scale WiFi deployment to efficiently localize devices.

1.3 Harnessing Active Scans for Data Transfer in WiFi-based IoT Nodes

Given the ubiquitous nature of WiFi, it is an apt choice for IoT communication since it saves the cost of deployment both at the client and at the infrastructure-level [49, 50, 51]. Therefore, when considering the deployments at a large-scale, IoT deployments are progressively integrated with present-day WiFi networks. An example is Amazon's latest WiFi-based IoT network [52]. However, WiFi

networks, especially in dense scenarios, are known to suffer from problems such as frequent disconnections, interference, and contention [53, 54]. It is a challenge for the node to reach an AP reliably. This is true for both outdoor and indoor deployments. The scenario arises when an AP is distant from the node, making the connection intermittent. If a connection is established, maintaining it becomes a challenge. If the distance is not an issue, the sheer number of nodes will make it hard for the AP to cater to all of them. A sporadic connection results in draining the battery of the node.

There are various solution approaches for improving the performance such as interference management [55], load balancing [56], or radio resource management [57]. Even with these, we can not guarantee that an IoT node can associate with any AP; which is because of several issues related to network administration [58]. Some places have strict network policies that do not allow any other devices, but the work-related devices of employees, to be part of the official enterprise WiFi network. Even further, in scenarios where the number of APs are lesser, but the number of clients is enormous, the client does not find an AP with which it can associate.

WiFi, as it exists today, is designed to provide high throughput data transfer for a small number of clients. The WiFi clients are expected to get associated with an AP within short range. Minimizing the energy was not one of the fundamental goals of the WiFi. However, we focus on IoT nodes that have short and asynchronous data requests and are mostly battery powered devices. Using WiFi in its current form for the IoT nodes, which mandates an association with

an AP, requires IoT node to maintain the WiFi connection actively. The process of connection maintenance not only injects heavy network management traffic but, consumes a lot of energy [59, 60]. Approaches for saving energy for IoT nodes use energy harvesting sensors [61, 62] or low-cost WiFi sensors [63]. The primary goal of these solutions is to optimize battery consumption and not to combat the issues of intermittent WiFi connectivity. Nonetheless, both solutions mandate a WiFi association to enable data transfer.

In short, the problem is that even though WiFi is a convenient option, it is far from optimal for the IoT nodes. An option is to use Ad-hoc WiFi networks. However, it requires a mesh network protocol, which is far too complex for a large-scale network of IoT nodes [64] and not suitable for infrastructure-based WiFi networks. We explore whether the IoT nodes can send data *without even associating with an AP.* If such communication is possible, the issues of establishing and maintaining a connection will be eliminated. We leverage active scanning to solve the problem. A probe request always reaches an AP, irrespective of the load on an AP or the distance of client from the AP, which is not the case for a data frame. However, the response from the AP may specify that it is overloaded and does not intend to allow association.

Given that the transmission of probe requests as part of active scanning is perpetual, we ask the question *is it possible to enable communication for WiFi-based IoT nodes without the association?* We propose to piggyback data on the probe requests. With this idea, we eliminate the need of a WiFi association and call our protocol – *ViFi* protocol. *V*iFi (scanning) comes before *W*iFi (associ-

ation). This protocol enables IoT nodes to transmit data in an energy efficient manner in any WiFi network without undergoing either administration related hassles such as MAC address registration or necessity of an association.

Transmitting data over the management frames [65] has been explored previously for various applications [66, 67, 68, 69]. Most of the previous research are application oriented, for example indoor positioning [68] or disaster recovery [69]. We saw in the previous problem, that the extended amounts of unnecessary active scans can severely deteriorate the performance of a WiFi network. Therefore, our focus is to assess the performance of data transmission without association at scale, in comparison and co-existence with the default association-based protocol.

We consider a WiFi network deployment where establishing a WiFi association with an AP is either not possible at all or the WiFi association is poor and intermittent. This network has WiFi-based IoT nodes as clients possibly along with other non-IoT WiFi clients such as laptops and smartphones. We consider uplink data transfer, *i.e.* from the clients to a data repository reachable from the AP. In such a network, we want to enable the communication from the clients to an AP even at low -70 dBm RSSI. Our focus is to make the communication between the clients and the AP efficient.

To the best of our knowledge, unlike most previous works we are the first to present an in-depth analysis of ViFi from the perspective of the MAC layer.

For all three problems, we successfully demonstrate the performance benefits

with the proposed solutions in real-world WiFi networks. In summary, this thesis attempts to showcase that, simple enhancements at the MAC layer are credible enough to solve complicated and otherwise, hard to decode problems of dense WiFi networks.

1.4 Thesis Methodology and Organization

In this thesis, we take an empirical approach towards architecting the mechanisms at the MAC layer to improve the performance of WiFi networks. The aim of this thesis is to develop easy to integrate solutions with existing network infrastructure and the devices. Hence, we work with real-world WiFi networks with actual WiFi devices and study the intrinsics of WiFi drivers at various levels of the protocol stack. Figure 1.1 summarizes the methodology followed.

We collect WiFi traffic passively with sniffers [70, 71] from venues such as university campus, conference hall, and airport to validate the existence of problems. We place sniffers close to the APs listening to the same channel as the AP, to maximize the likelihood of recording traffic as observed by the AP. A location closer to an AP is a good vantage point for monitoring the network.

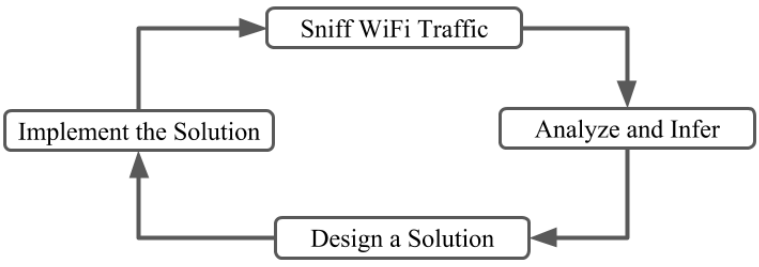


Figure 1.1: Thesis Methodology

Also, this allows our approach to be used by enterprise WiFi AP(s), which often come with an additional WiFi chip for sniffing and/or have the ability to capture all frames decoded by the AP over the channel [72], for example, SS-300-AT-C-60 AP from Mojo Networks. Packet Captures (PCAPs) containing MAC layer traffic as captured by the sniffer are converted to files containing comma separated values (CSVs). The CSVs are then analyzed to design the required solutions. Further, we develop and test solutions on real devices with a series of controlled and uncontrolled experiments.

Rest of this thesis is organized as follows – Chapter 2 explains the protocol of active scanning as described in the `IEEE 802.11` standard. The chapter presents the details of *how* active scanning is triggered at the client-side and the impact and the extent of active scans in WiFi networks. Chapter 3 discusses methods to deal with unnecessary active scans. It introduces a metric to monitor the growth of probe traffic in realtime, presents an inference mechanism to detect the causes of the active scans, and a client-side solution to curb active scanning. Chapter 4 analyzes the accuracy of a fingerprint-based indoor localization system in the absence of active scans. The chapter describes three floor-detection heuristics to improve the performance of the localization. Chapter 5 introduces a protocol that enables data transfer in WiFi networks for IoT nodes when a WiFi association cannot be established. The chapter demonstrates the advantages of this protocol over standard WiFi protocol that mandates a WiFi association. Chapter 6 presents the literature survey. Chapter 7 concludes this thesis and discusses the directions for future research.

Chapter 2

Primer on Active Scanning in WiFi Networks

A large-scale WiFi network deployment consists of a few hundreds of WiFi APs that serve several thousands of WiFi clients. Given the scale of such WiFi networks, they are managed by a central controller *i.e.*, a Wireless Local Area Network (WLAN) controller that aids in centralized management of APs and clients. APs relay information about the part of the network they are serving to the controller. An example of a summary of such information is the Cisco MIB(s) [73]. The controller uses this information to optimize the network, for example, via channel and power selection for the AP(s). It manages all APs by taking care of functions such as Radio Resource Management (RRM), QoS, and Roaming [74]. Interference management algorithms as part of RRM at the controller take care of assigning channels dynamically to the APs. WiFi clients access the services of the WiFi network once they associate with an AP. Most WiFi networks, as of today, are dual-band *i.e.*, they operate in both 2.4 GHz

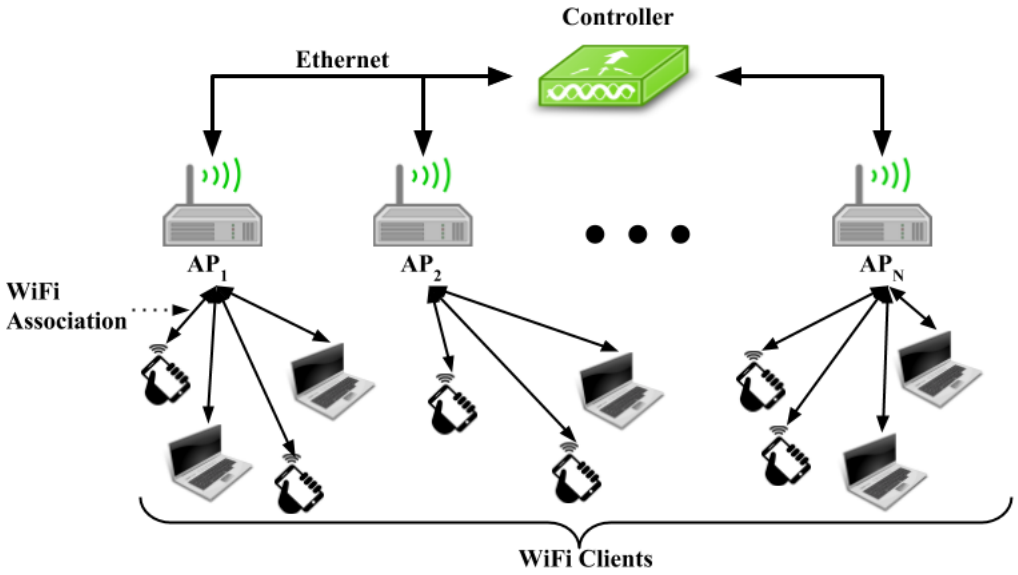


Figure 2.1: WiFi Network Architecture. Large-scale WiFi network deployment has a controller that manages several APs serving a large number of WiFi clients.

and 5 GHz. Figure 2.1 shows an abstract architecture of a typical WiFi network deployment.

Since such controller-managed WiFi networks are deployed at enterprises like universities and offices, they are also known as *Enterprise WiFi networks* and the APs as *Enterprise APs*. Likewise, the APs that are not the part of an *Enterprise WiFi network* are known as *Non-enterprise APs*. Non-enterprise APs are either mobile hotspots or APs brought by users for their personal work, for example, students bring APs for their experiments in labs. They operate in conjunction with enterprise APs however, they are not under the control of the controller. Hence, their operating channels may or may not overlap with those of enterprise AP channels.

The communication between a WiFi client and an AP is realized with the MAC layer frames defined by `IEEE 802.11` protocol. These frames are clas-

sified into three categories of Management, Control, and Data. These frames are used for establishing, managing, and controlling data transfers using WiFi connections. Data frames carry information to be transmitted from source to destination. Control frames help in the efficient transfer of data frames. Management frames help in establishing and maintaining WiFi connections.

Of the three categories of Management, Control, and Data Frames, initiation and transmission of control frames in the network are dependent on the number of data frames, which is not the case for management frames. Beacons and probes are two prominently visible management frames. Beacons help already associated clients to know the current status of BSS, such as current channel, BSS load, and new clients to discover the APs by passive scanning. They are transmitted by AP at scheduled intervals as determined by Target Beacon Transmission Time (TBTT). Since, the number of APs, as well as TBTT, do not change often in the network, the number of these frames in a given interval of time does not change much. Probe frames serve a similar purpose as beacons, with a difference that clients initiate them as part of the active scanning procedure. The number of probe request frames is dependent on factors such as client roaming, packet losses, and vendors. The number of probe responses is dependent on metrics such as the number of APs or the channel overlap. Since these factors are dynamic, the number of probe frames is a dynamic entity; hence the growth of these frames often change with time.

2.1 States of a WiFi Client

A typical WiFi client transitions into multiple states of the WiFi protocol while establishing and maintaining an association with AP. We present an abstract view of these states in Figure 2.2. Initially, the client is in the *sleep* state, where its WiFi interface is turned OFF. When the WiFi interface is turned ON, the client transitions into the *wake up* state. In this state, the interface is powered ON but it is not associated with any AP. Before the client can establish a WiFi connection, it needs to know what all APs are available in its vicinity.

For this purpose, the client scans all the channels. This state is termed as the *scan* state. Post-scanning, the client chooses an AP and get associated to that AP and thereby it transitions into the *associated* state. Now, the client is ready to transmit data. Once the client is in the *associated* state, the WiFi connection needs to be monitored regularly for the maintenance of the state. In this state, the node measures the quality of the association in terms of beacon losses, bit-rate, and RSSI. If the client infers that the quality of the connection is deteriorating, it again transitions into the *scan* state and searches for a better

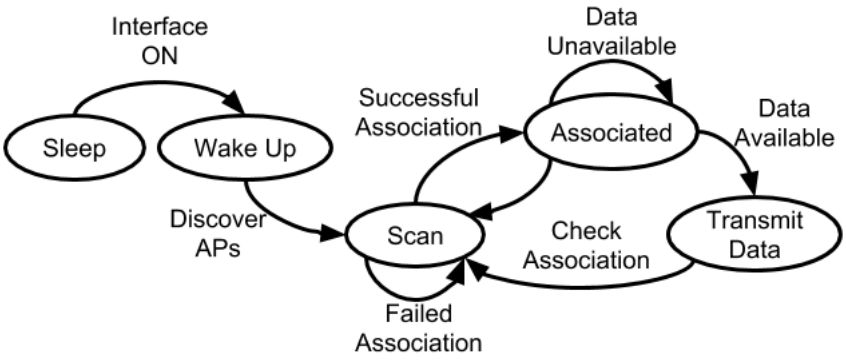


Figure 2.2: Default States of a WiFi Client.

AP to associate with. This will result in disassociation with the current AP and a new association with an AP.

2.2 Active Scanning

`IEEE 802.11` defines two protocols for AP discovery – passive and active scanning. During a passive scan, the client silently listens for beacons, a type of management frame, from the APs for a predetermined duration. Whereas during an active scan, the client actively sends a management frame called *probe request* and waits for another management frame from APs called *probe response*. By default, the WiFi clients follow an active scan as it is known to encounter a much lesser delay in discovering AP than the passive scan [75, 24, 25, 26, 27].

2.2.1 The Causes

The causes of active scanning can be broadly grouped into the three categories of – (a) Discovery, (b) Connection Establishment, and (c) Connection Maintenance. Figure 2.3 lists these categories and the causes under each category. The causes were arrived at by studying a range of devices and their device driver implementations, which are summarized in Table 2.1. Next, we explain these categories.

Discovery: We have observed that a client periodically looks for new AP(s) in its vicinity. This process of *discovery* of AP(s) is enabled by periodic active scanning. The process is enabled both when a client is unassociated and asso-

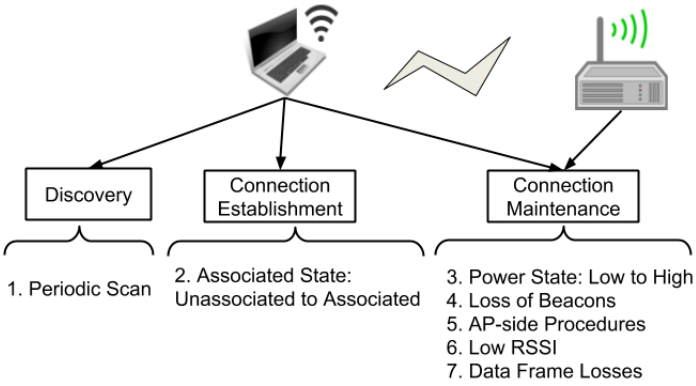


Figure 2.3: Causes of Active Scanning

Device Type	Chipset Vendor	Operating System	Device Driver
Laptops	Atheros, Intel, and Broadcom	Ubuntu 14.04/12.10 and Windows 8.1/7	$ath9k$, $iwlwifi$
Tablets	Qualcomm and Broadcom	Android KitKat and iOS 8	OS provided WiFi drivers
Smartphones	Broadcom, Mediatek, and Qualcomm	Android KitKat/Marshmellow/-Jellybean/Cyanogen and Windows	OS provided WiFi drivers
USB Adapter	Realtek and Atheros	Ubuntu 14.04/12.10 and Windows 7/8.1	rtl8812AU, $ath9k\_htc$

Table 2.1: Specifications of devices studied to find the causes of Active Scanning

ciated to an AP. The period at which discovery is initiated is not standard and may be determined by the application or driver initiating it. To exemplify, in the `wpa_supplicant` the period is observed to grow exponentially. Starting at a small value of 3 seconds, it takes values of $9, 27, \ldots$, with a maximum value of 300 seconds.

Connection Establishment: A client that is unassociated looks to establish a connection with an AP in its vicinity. The connection establishment procedure that helps the client to transition from the unassociated state to being associated with an AP is observed to contain probe request(s) sent by the client to the AP.

We believe this enables a faster response from the AP.

Connection Maintenance: Once a client is associated to an AP, it monitors its connection to the AP. Specifically, it measures the RSSI from the AP, the fraction of data frames that it transmitted unsuccessfully, and the rate at which beacons from the AP are lost. Loss of beacons, low RSSI, and failed data frame transmissions have been observed to start active scanning. These correspond to the causes numbered $4, 6, 7$ in Figure 2.3.

Next, the procedures at the AP (cause 5 in Figure 2.3) can trigger active scanning. Specifically, we have observed that an AP may send deauthentication messages to a client to carry out load balancing and/or to make the client switch to a different frequency band. This leads to a client starting active scanning. Last but not least, a client may often be in a low power state. We have observed that a transition to a high power state (cause 3 in Figure 2.3), for example, due to increased user activity when the screen of an Android phone is lit up, is accompanied by the client starting active scanning.

2.2.2 The Protocol

Figure 2.4 demonstrates the process of an active scan. When active scanning is triggered, the client prepares a probe request frame. The client has a channel list to be scanned for each frequency band it is capable of operating in. For each channel in the list, the client sends a probe request following the basic CSMA/CA procedure. After a probe request is sent, client starts `ProbeTimer`

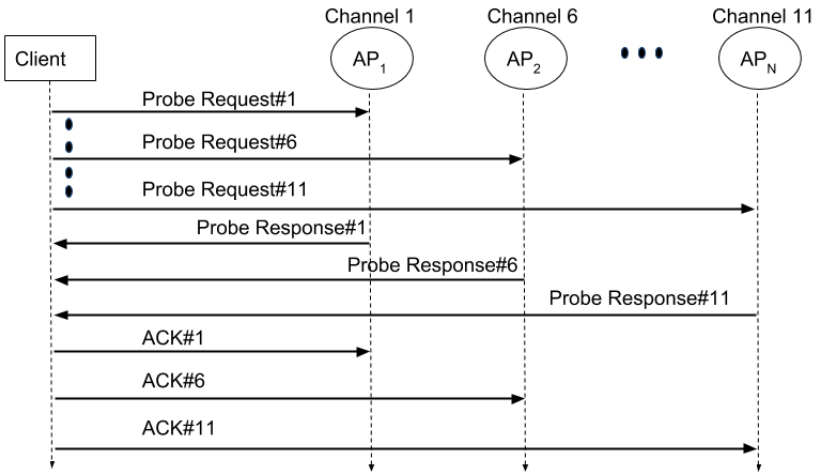


Figure 2.4: Protocol Operation of Active Scanning. The diagram shows an exchange of probe request, probe response, and acknowledgment (ACK) frames during an active scan. A WiFi client sends a probe request on each channel of its frequency band. In the figure, we show three channels – 1, 6, and 11 of the 2.4 GHz frequency band. All nearby APs, three in this case, who hear a probe request, respond with a probe response. The WiFi client acknowledges each probe response with an acknowledgment. In case an AP does not receive an ACK, it resends the probe response.

that waits for `MinChannelTime` to detect signal on this channel. If a signal is detected, the client anticipates the possibility of an operational Basic Service Set (BSS) nearby. So, it waits for `MaxChannelTime` to receive all the probe responses. On receiving all probe responses, it sends an acknowledgment back to the responding BSS's. All the probe responses are processed to fetch the information about the nearby BSS's. `IEEE` standard defines this procedure but the values of these timers are vendor specific.

Active scanning is triggered at the WiFi client, either in the user space or in the kernel space of an Operating System. Figure 2.5 summarizes how these interact with the WiFi chipset. Examples of user space applications include the Ubuntu Network Manager and the Android `wpa_supplicant`. The kernel space includes the WiFi MAC drivers such as `mac80211` or device drivers such as `ath9k` and `iwlwifi`.

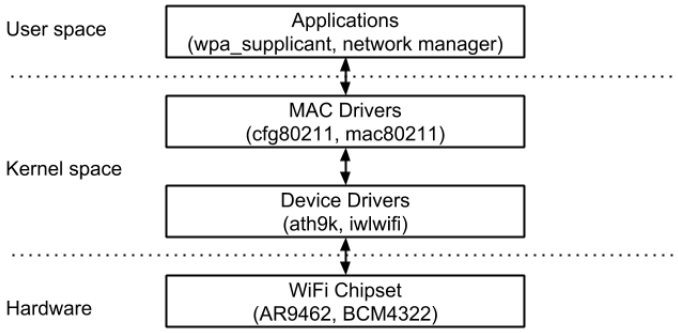


Figure 2.5: Client driver implementation of WiFi functionality. A schematic overview to represent the end-to-end WiFi implementation at the client driver; starting from the userspace to the hardware.

2.2.3 The Challenges

Active scanning seems to be a simple process of sending and receiving multiple probe requests and probe responses, respectively. However, in practice, it is not only a result of multiple causes defined at layers of software stack but also complex interactions among these layers. Userspace applications and kernel space drivers are diversified with the availability of numerous operating systems and WiFi chip vendors.

All these factors together make it challenging to answer questions related to active scanning, at scale. For example, *what causes WiFi clients to trigger active scans?*, *Does active scan impact performance of WiFi networks?*, or *what is the network-wide impact of active scans?*

Next, we present the extent of active scans and their impact on the performance of real-world WiFi networks.

2.3 Understanding the Extent and the Impact of Active Scans

We perform a detailed analysis to understand *why* and *how* does active scanning affect network performance? Most enterprise WiFi networks are dual-band and both bands exhibit different channel characteristics, so we began our analysis with comparing the scanning behavior of devices in 2.4 GHz and 5 GHz (Section 2.3.1). Our empirical analysis revealed that even though India has dual-band enterprise WiFi networks deployed, most devices operate in 2.4 GHz. Thus, we conducted the study in a campus network in Singapore where a comparable number of devices operate in each band. Our analysis revealed that the clients in 5 GHz scan 1.65% times lesser due to lower interference and a higher number of non-overlapping channels.

With this analysis and the fact that most devices in India operate in 2.4 GHz, we pursue further analysis with a specific focus on 2.4 GHz. We present the analysis from two perspectives (Section 2.3.2) – (a) client-side and (b) network-wide. It is important to consider two different perspectives because not only does active scanning introduces latency to an ongoing communication at a client but when it starts, low bit-rate probe traffic is injected into the network.

For the client-side analysis, we log the details of network-related events on the client itself. Such details are generated by the kernel and provide us insights from the perspective of a client involved in active scanning. Client-side logging, however, is impractical for the network-wide analysis as there are several thousands of devices involved. Therefore, we study the WiFi traffic collected with

passive sniffing with the help of monitors. We consider sniffer-based logs from three real-world WiFi networks. These logs are collected at different times and venues so as to prevent our analysis from any network-specific bias and provide us a comprehensive view of active scanning behavior. We, on purpose, diversify our datasets to analyze the scanning behavior devices under different network settings.

2.3.1 Comparing Active Scanning– 2.4 GHz vs 5 GHz

We collected WiFi traffic on one floor in one of the buildings of Singapore Management University (SMU) for 6 hours, from 11 AM to 5 PM. This time slot coincides with the office hours, thereby ensuring the recording of traffic from maximum clients. This floor has 15 APs deployed with each AP broadcasting 4 Service Set Identifier (SSIDs). At least 200 people, including students, faculty, staff, visitors, with a minimum of 1 WiFi device per person are expected throughout the day. We recorded frames with sniffers on channels 1, 6, 11 in 2.4 GHz and 36, 52, 161 in 5 GHz. Note that these channels were in operation at the time of data collection. Our sniffers were TP-Link WN721N USB WiFi adapters for 2.4 GHz and Intel Centrino 6230 PCI cards for 5 GHz. Recall that we place sniffers close to APs to maximize the likelihood of frame capture. We only examine the PHY header and type of frames, specifically probe requests for this analysis. Sniffer recorded a total of 222 and 49 clients per minute in 2.4 GHz and 5 GHz, respectively. Out of these 70 clients and 20 clients in 2.4 GHz and 5 GHz, respectively, were associated. We assume the remaining

clients to be unassociated. However, please note that the number of unassociated clients is an overestimate because of the latest MAC address randomization procedures [76].

We hypothesize that the stationary clients in 5 GHz transmit lesser probe requests than the mobile clients. The reason is reduced interference in 5 GHz, which in turn reduces the amount of probe requests that emanate due to maintaining the WiFi association. In the other case, the lower range of 5 GHz frequency band results in a dense deployment of APs to ensure coverage. Hence, the mobile clients trigger active scanning as soon as the RSSI falls below the roaming threshold, which is -70 dBm typically. Thus, mobile clients experience frequent handovers and probe more in 5 GHz.

We compare the scanning behavior in both frequency bands for the following metrics – (a) #Probe requests transmitted by each WiFi client per minute and (b) Effect of RSSI variations.

Our first metric – #probe requests per client per minute signifies the normalized value of the number of probe requests transmitted by associated and unassociated clients together in each frequency band. Figure 2.6 presents an analysis of this metric. We notice that while #probe requests per client per minute in 2.4 GHz are limited to 1 or 2, in 5 GHz as high as 6 probe requests per client per minute are transmitted. Not only this but the variance is also 5 times higher than 2.4 GHz. With this, it appears that 5 GHz will experience higher probe traffic than 2.4 GHz. However, as we dwell deeper in the analysis, we find that this is

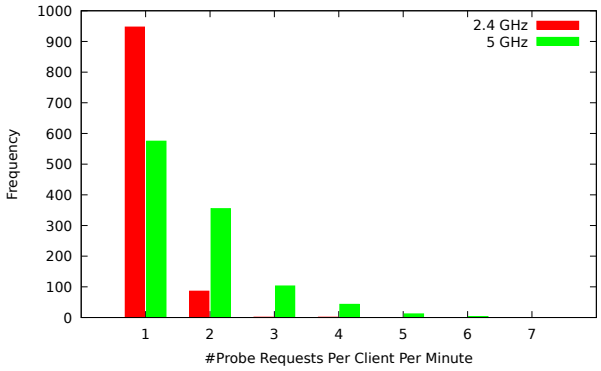


Figure 2.6: Frequency of #probe requests per client per minute in 2.4 GHz and 5 GHz. The frequency of 1 probe request per client per minute is 1.65 times lesser for 5 GHz than 2.4 GHz. Average and maximum are 1 and 2 in 2.4 GHz while 2 and 6 in 5 GHz. Standard deviation and Variance are 0.40 and 0.16 in 2.4 GHz while 0.93 and 0.87 in 5 GHz.

a fallacy.

We proceed further to analyze the effect of RSSI on the #probe requests, #status changes, and the type of SSIDs. All results are normalized to 500 clients per minute in each band. We infer the RSSI values from the PHY header of data frames recorded at the sniffers. Figure 2.7 shows the result of this analysis. We considered 8 classes of RSSI from -20 dBm to -90 dBm with decrementing -10 dBm/class. For every class, we plot the average #probe requests transmitted. While 2.4 GHz sees almost constant #probe requests for all RSSI classes, 5 GHz shows sudden growth in #probe requests as RSSI is close to the roaming threshold (from -40 to -70 dBm). In such scenario, #probe requests in 2.4 GHz grow by 3.15% while in 5 GHz they grow by 209.02%. For all other classes, either there are no probe requests in 5 GHz or $<=$ 2.4 GHz.

To verify this, we study the effect of variation in RSSI on the number of times clients change their association status. We term this metric as #status changes. Figure 2.8 shows the result of this analysis. While in 2.4 GHz, #status changes

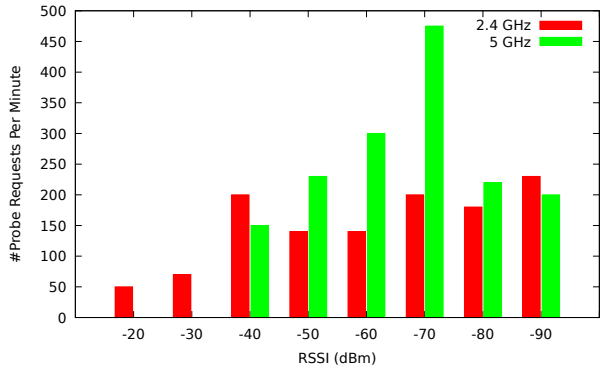


Figure 2.7: Comparing the #probe requests per minute with the variation in RSSI. Notice the growth in #probe requests in 5 GHz as RSSI approaches close to -70 dBm.

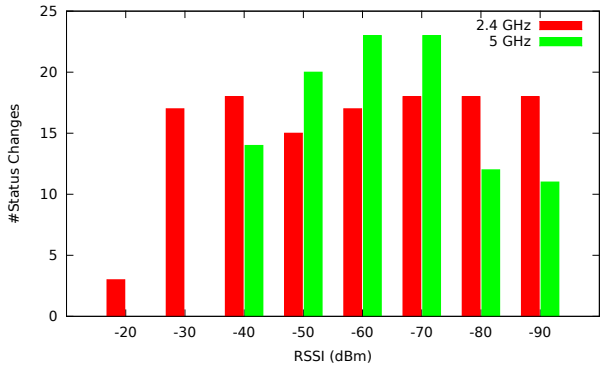


Figure 2.8: Comparing the #status changes (Associated to Unassociated and vice-versa) with the variation in RSSI. Notice the growth in #status changes in 5 GHz as RSSI approaches close to -70 dBm.

are mostly constant across RSSI classes, in 5 GHz we notice 64.28% increase as the RSSI decrease from -40 to -70 dBm.

We validate this result by analyzing the type of SSIDs broadcast in the probe requests when RSSI is close to roaming threshold. We categorize the SSIDs as enterprise and non-enterprise SSIDs. Enterprise SSIDs are the ones, which are part of WiFi network deployment, while non-enterprise SSIDs are not. The occurrence of such probe requests signifies triggering of handover at the client. Our analysis reveals that the number of enterprise SSIDs in the probe requests grow from 46.66% at -40 dBm to 81.23% at -70 dBm in 5 GHz, while in 2.4 GHz they stay at around 45%.

This result provides conclusive evidence for the argument that more #probe requests in 5 GHz are due to mobile WiFi clients. However, stationary clients send lesser probe requests in 5 GHz. Revisiting Figure 2.6, we now understand that 5 GHz experiences 1.65 times lower instances of 1 probe request per client per minute than 2.4 GHz. Clients responsible for this number were near-stationary and associated. However, high variance in #probe requests per client per minute is due to high RSSI variations.

Data analysis reveals that stationary clients transmit lesser probe requests in 5 GHz band than in 2.4 GHz. We further confirmed this conclusion by collecting scanning data on the client itself. We analyzed data from the following phones – iPhone 6, Nexus 5X, Galaxy S7, Galaxy S3, Moto G4, and Sony Xperia, for up to 6 hours.

The fact that most devices in developing countries, like India, still operate in 2.4 GHz it is imperative to study the active scanning in detail. We present the impact of active scanning in the next section.

2.3.2 Impact of Active Scanning in 2.4 GHz

We present the client-side and network-wide analysis on the impact of active scanning. We show the drop in goodput due to increase in probe traffic with the help of a case study.

2.3.2.1 Client-side Perspective

(1) Latency: To understand the client-side impact, we study how latency, *i.e.*, the time it takes for message transfer between a client and an AP is affected when active scanning is triggered. Latency is an important metric [77] to understand this impact because it ultimately affects real-time applications, such as VoIP.

Existing sniffer-based logs do not allow us to understand latency experienced by a client at the application layer. Therefore, we collect latency data at a stationary client in a dense WiFi network deployed in our university campus of IIIT-D, where the client receives beacons from ≈ 10 BSS. The client remains stationary and associated with one AP, with good (≈ 55 dBm) RSSI, throughout the data collection period. Any external applications, for example, Network Manager, that trigger active scanning are disabled. We instrument the device to trigger active scanning once a minute. Note that this experiment is to study the impact of active scanning on latency experienced by a stationary client, irrespective of its cause. We measure latency with the metric Round Trip Time (RTT) [77] reported by `ping` [78] utility at the client.

The client under observation is a 802.11n enabled Atheros AR9462 WiFi chipset and it runs Ubuntu 14.04 operating system. We disabled the power-save feature at the client, to rule out the possibility of it affecting the latency. All APs in IIIT-D WiFi network are $802.11ac$ enabled. Both the client and the APs are capable of operating in both 2.4 GHz and 5 GHz frequency bands. We initiated a `ping` session for 5.5 hours on the client, where it pings the AP to

which it is associated. The duration is enough to showcase the impact on RTT. We record RTT values on the WiFi client itself. Figure 2.9 shows variation in latency experienced with time.

Peaks in latency arise due to active scanning triggered at the client. Active scanning not only impacts latency the moment it is triggered but, continues to increase latency for the next few seconds after that. We demonstrate this phenomenon in the zoomed-in subplot in Figure 2.9. It happens so because the arrival of data frames at the client result in deferring the transmission of remaining probe requests from the ongoing scan. Table 2.2 shows the statistics for latency in the presence and absence of active scans at the client. Average latency increases from mere 3.57 ms in the absence of active scans to 21.72 ms when the client triggers the scans. Even higher divergence is reported in the variance.

(2) Association Patterns: Now we analyze if active scanning helps in choosing a different BSS. We do this by studying the association pattern of stationary

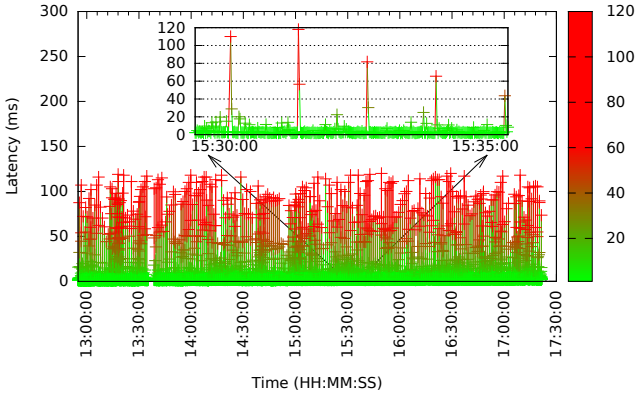


Figure 2.9: Impact of active scanning on latency experienced by the client in realtime. The plot shows data of 5.5 hours long experiment. Red peaks denote active scanning. Due to many data points, we zoom in the plot for a 5 minute window. Active scanning is triggered 5 times in this window, notice that on its occurrence latency increases from ≈ 20 ms to ≈ 100 ms.

Statistic	No Scan	Scan
Mean	03.57 ms	21.72 ms
Variance	53.94 ms	1057.48 ms

Table 2.2: Client latency statistics in the presence and absence of active scans. Mean latency grows by 6x and variance grows by 19.6x when the client triggers active scans.

WiFi clients. Precisely, we study two sets of BSS's – (a) the set that responds to probe requests broadcast by a WiFi client and (b) another set to which it associates. We perform this analysis in the live WiFi network of IIIT-D; hence we do not assume that all APs are uniformly accessible. A typical WiFi client is in the vicinity of 2-3 APs, where each AP broadcasts 5 BSSIDs. Note that each BSS in operation transmits its beacons and probe responses. We randomly preselected 34 clients and analyzed their association patterns for 10 days. While we ensured that the devices should be present on all days of data collection, we did not mandate their association choice.

Figure 2.10 shows the results. All BSS's in operation respond to probe requests without considering the type of client. To exemplify, a rogue client, one who is not authorized to access the WiFi network, and an official client, one who is entitled to access the WiFi network, both receive probe responses. Our data analysis confirmed this observation with a maximum of 11 BSS's discovered per WiFi client. Further, APs broadcast many BSSIDs to implement Virtual Local Area Networks (VLAN). Not all authorized clients are allowed to access all VLANs. However, irrespective of this fact, all BSS's respond to all probe requests. We find that more than half of the clients chose a maximum 4 out of 11 discovered BSS's to associate. Specifically, 38% associated with 3 while 24% with one of 4 BSS's. Furthermore, 20% did not associate with any BSS. Probe

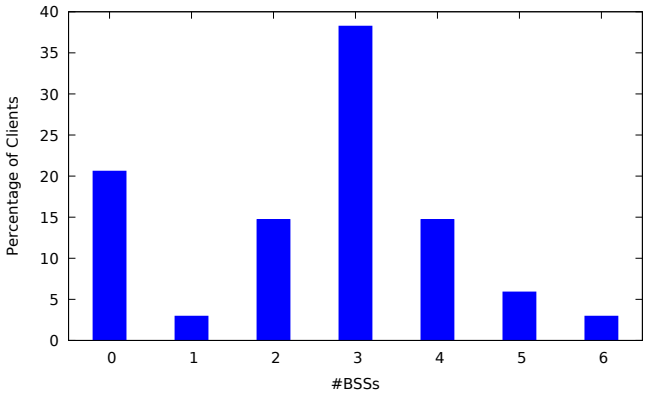


Figure 2.10: Association pattern of 34 pre-selected clients in IIIT-D WiFi network over a period of 10 days. 38% of clients selected only 3/11 discovered BSS's.

responses from BSS's to which clients did not associate were unnecessary.

For stationary clients, active scanning increases the latency. Moreover, even though they discover many BSS's, they rather associate with a much smaller subset. This observation implies that most of the probe requests and probe responses are inessential and their discontinuation should improve the performance of a network.

Next, we analyze the impact of active scanning from the perspective of the whole network.

2.3.2.2 Network-wide Perspective

To study the impact of active scanning on a network-wide scale, we consider sniffer-based logs from three real-world WiFi networks – [SIGCOMM] [79], [IIT-B] [53], and [IIIT-D]. The SIGCOMM dataset contains WiFi traffic from the SIGCOMM 2008 conference [79]. It has WiFi traffic from 4 AP(s) captured by 8 sniffers. AP(s) are operating in the 5 GHz frequency band. We analyze

the WiFi traffic of 2 days captured by one of the sniffers. It contains traffic from about 500 WiFi clients. The IIT-B dataset was collected from a classroom WiFi network in IIT-Bombay. It has one AP that is being used by 43 students for about 2.5 hours [53]. The IIIT-D dataset was created by us by collecting 63 hours of WiFi traffic over a period of 10 working days from 10 AM to 6 PM in our institute's, *i.e.* IIIT-Delhi, WiFi deployment provided by Cisco. The traffic was captured by a sniffer that was placed in a lobby close to an AP serving the lobby, as part of the WiFi network at the institute. These datasets consist of logs recorded using sniffers in real uncontrolled WiFi networks settings.

Table 2.3 summarizes details of these datasets. We begin with providing the details of probe traffic in these logs followed by measuring the amount of redundant probe traffic.

(1) Extent of probe traffic in Real-world WiFi Deployments: We extract the details about probe requests and probe responses from the logs. Particularly, we study the amount of management and probe traffic, frame sizes, PHY bit-rates, and inter-frame arrival times.

Amount of management and probe traffic Table 2.3 lists the #frames recorded

#	[SIGCOMM]	[IIT-B]	[IIIT-D]
Band	5 GHz	2.4 GHz	2.4 GHz
Days	2	1	9
Hours	14	2.5	63
Total Frames	5,922,772	40,080,792	149,879,321
Management Frames per minute	468	1469	5541
Probe Frames per minute	70	761	973

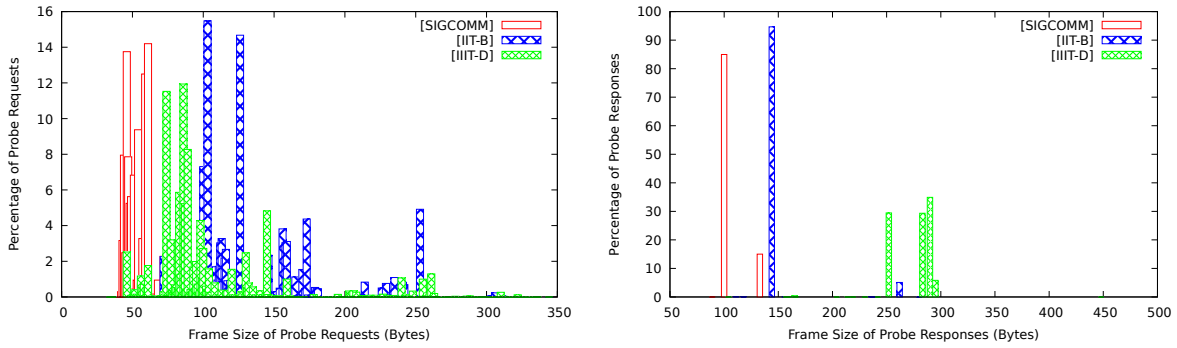
Table 2.3: Details of datasets

per minute. We report the numbers for total traffic in general, management and probe traffic in particular. [SIGCOMM] has the lowest #probe frames per minute followed by [IIT-B] and [IIIT-D]. We explain this observation with the following reasons – (a) clients are less likely to probe in 5 GHz band, which is the case for [SIGCOMM], (b) [SIGCOMM] has the lower #frames and losses recorded than [IIT-B] [53], and (c) [IIIT-D] has the highest #clients and APs recorded, which means a higher number of probe requests and probe responses.

As per the IEEE standard, frames like beacons, association requests, and responses, authentication requests and responses, and probe traffic together constitute management traffic. We note that out of the total management traffic recorded, the proportion of probe traffic for [SIGCOMM], [IIT-B], and [IIIT-D] is 14.95%, 51.80%, and 17.56%, respectively. These numbers signify that [SIGCOMM] and [IIIT-D] see more beacons and association related events than [IIT-B], which sees more scanning related events.

Frame Sizes We observe that probe requests have variable size, but the size of probe responses is near constant for a given WiFi network. The reason of which is that probe requests and probe responses are transmitted by clients and APs, respectively. For a given WiFi network, specifically, the ones deployed in enterprises, APs are from the same vendor while the clients are from different device vendors. Figure 2.11 shows the percentage of probe requests and probe responses for each unique frame size.

The median size of probe requests for each of the datasets is– [SIGCOMM]:



(a) (b)

Figure 2.11: Frame size of probe requests and probe responses. We report the percentage of total probe requests or probe responses for each unique frame size. The size of probe requests vary for different WiFi clients, which is not the case for probe responses. Further, [SIGCOMM] has smaller probe requests than the other 2 datasets, while [IIIT-D] has the highest frame size for probe responses.

50 bytes, [IIT-B]: 110 bytes, and [IIIT-D]: 70 bytes. More than 80% of probe responses for [SIGCOMM] and [IIT-B] are of 100 bytes and 150 bytes, respectively. However, [IIIT-D] contains probe responses of variable sizes, ranging from 250-300 bytes. The reason for this variability is twofold. First, [IIIT-D] has the highest #BSS's than the other two datasets. #BSS's in [IIIT-D] are 5, in [SIGCOMM] are 2, and in [IIT-B] is 1. Second, a mix of enterprise and non-enterprise APs operate together here. Revisiting the definition of enterprise and non-enterprise, non-enterprise APs as the ones that are not part of official WiFi network deployment. In the case of [IIIT-D], enterprise APs are deployed as part of Cisco WiFi network, while few users deploy non-enterprise APs for their personal usage.

Bit-rates We find that $\approx$ 100% of the probe requests are sent at 1 Mbps if the frequency band of operation is 2.4 GHz, as seen in [IIT-B] and [IIIT-D]. However, the rate is 6 Mbps in 5 GHz, as seen in [SIGCOMM]. bit-rate of probe responses is found to vary among 1, 6, or 24 Mbps. For each dataset

the bit-rate of probe responses is as follows - [SIGCOMM]: 6 Mbps, [IIT-B]: 1 Mbps, and [IIIT-D]: 99.5% at 24 Mbps and rest at 1 Mbps. The bit-rate of probe responses depends on the rate configured in the BSS and type of WiFi clients in the network. Mostly, in 5 GHz lowest bit-rate is 6 Mbps. Therefore, we find the bit-rate of probe requests and probe responses in [SIGCOMM] to be 6 Mbps. The presence of legacy WiFi clients mandate the BSS to send probe responses at the lowest bit-rate of 1 Mbps, and for non-legacy WiFi clients, BSS can be configured to send probe responses at a higher bit-rate. [IIIT-D] has both non-enterprise APs as well as legacy WiFi clients. Thus, 0.5% of probe responses are sent at 1 Mbps.

Inter-Frame Arrival Times We define Inter-Frame Arrival Time as the time between the arrival of 2 consecutive MAC layer frames. Figure 2.12 shows CDF of inter-frame arrival times for probe requests and probe responses. For [SIGCOMM], the median inter-frame arrival time for probe requests is 0.01 ms, whereas for the other 2 datasets it is 10 ms. These numbers show that the WiFi clients were more aggressive in [SIGCOMM] than those in [IIT-B] and [IIIT-D]. Recall that the average probe requests recorded per minute are 70, 761, and 973 for [SIGCOMM], [IIT-B], and [IIIT-D], respectively.

Therefore, even though inter-frame arrival time of probe requests in [SIGCOMM] is the lowest, least #probe requests per minute will not affect the performance of WiFi network, as the clients trigger active scanning in 5 GHz only while they are mobile. Further, the median inter-frame arrival time of probe responses for [IIIT-D] is less than 0.01 ms, [IIT-B] is less than 10 ms, and [SIG-

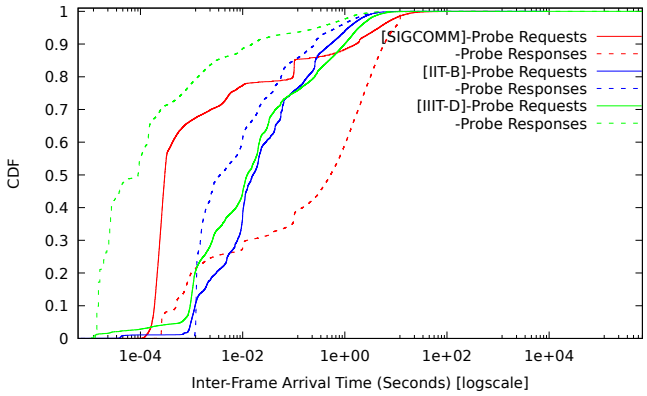


Figure 2.12: Inter-frame arrival time of probe requests and probe responses. [IIT-B] and [IIIT-D] see similar inter-frame arrival time for probe requests. [IIIT-D] sees lowest inter-frame arrival time of probe responses, followed by [IIT-B].

COMM] is less than 1 sec. A higher #operational BSS's result in reduced inter-frame arrival time, which is the case of [IIIT-D].

(2) Redundant probe traffic: Now, we introduce the metric *Redundant probe traffic*. We use the redundant probe traffic metric to compare details of discovered BSS's per scan. Specifically, these details include SSID, BSSID, Channel, and #Associated Clients announced in probe responses for consecutive episodes of active scans. We define an episode of active scan as a group of probe requests separated by less than a second. Given an episode, redundant probe traffic contains all those probe responses in the current episode that fetch same information about APs as was fetched in the predecessor episode. Figure 2.13 summarizes the details of redundant probe traffic for our datasets.

We find an average of 99% of probe responses for stationary WiFi clients fetch redundant information of nearby BSS's. Precisely, $12926/13069$, $72161/72321$, and $426477/429426$ probe responses are found to carry redundant information in [SIGCOMM], [IIT-B], and [IIIT-D], respectively.

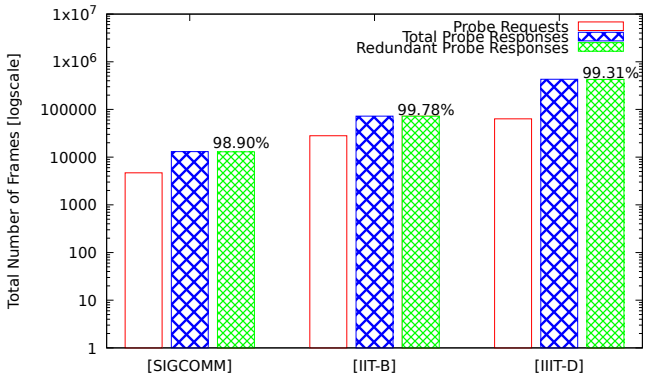


Figure 2.13: Redundant Probe Traffic. Total #probe requests, probe responses, redundant probe responses for [SIGCOMM], [IIT-B], and [IIIT-D]. For all 3 datasets, most of the probe responses carry same information about nearby APs and are hence, shown as redundant probe responses.

2.3.2.3 Case Study: Effect of Probe Traffic on $Goodput$ in 2.4 GHz

We demonstrate, with a case study, how probe traffic grows in a highly utilized network and ultimately how does that affect the MAC layer $Goodput$. For this purpose, we consider WiFi traffic for an hour on one of the busy days in IIIT-D campus.

For a given time slot T_s, we define $Goodput$ as:

$$Goodput = \frac{\sum_{i=1}^{N} D_s}{T_s} \text{bits/second} \tag{2.1}$$

where, N is the number of acknowledged data frames and D_s is the size of acknowledged data frames in bits.

We report the Channel Utilization value defined by `802.11` standard which is measured by the QoS BSS configured in the APs. As shown in Figure 2.14, the channels 1 and 6 are always heavily utilized, while channel 11 is intermittently heavily utilized. While a channel utilization higher than 50% denotes

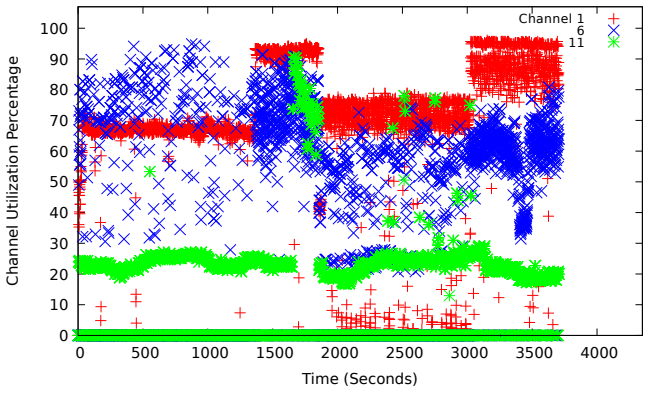
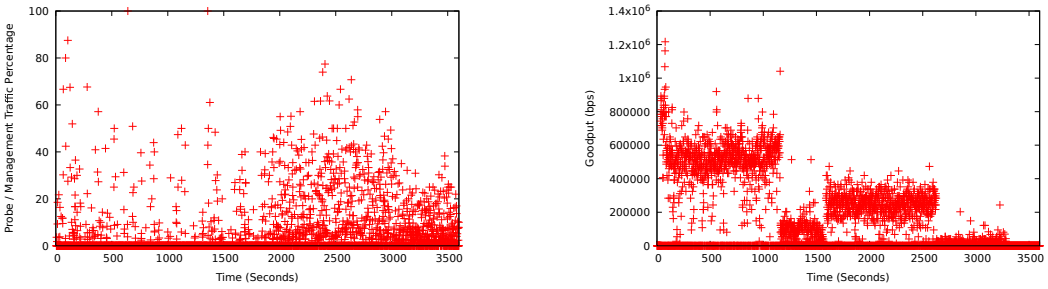


Figure 2.14: Channel utilization as reported by the APs for 2.4 GHz. Notice that channels 1 and 6 are heavily utilized.

heavy utilization of the channel, a channel utilization of 0 signifies either there is no data to be transmitted or clients are not able to transmit due to contention. We rule out the possibility of no data to be transmitted because we were aware that active WiFi clients were present at the time of data collection.

Next, we proceed to understand the amount of probe traffic in the network. We measure the percentage of probe traffic with respect to total management traffic recorded every second at the sniffer. Figure 2.15a shows the results of this measurement. We notice that the probe traffic is sparse before 1600 seconds and after that, it grows drastically. Effect of this growth is visible on the MAC layer goodput, which we demonstrate in Figure 2.15b. Not only does the goodput drop by more than 50% but, even channel utilization increases once the probe traffic grows. An overall increase in channel utilization indicates that other WiFi clients will have to contend more to get access to the channel, thus their performance will be degraded.

We summarize our findings as follows – (a) 2.4 GHz experiences higher



(a) Probe traffic to overall management traffic

(b) MAC layer Goodput (bits/second)

Figure 2.15: Case study to demonstrate the effect of probe traffic on $Goodput$. Notice the growth in probe traffic at around 1600 seconds and the corresponding drop[$\approx 50\%$] in the $Goodput$.

active scans than 5 GHz, (b) probe traffic can be as high as 60% of total management traffic, (c) 88% of probe requests search for non-existing BSS's, (d) 99% of probe responses for stationary WiFi clients fetch redundant information about nearby BSS's, and (d) not only does active scanning increases latency between a client and its AP, but it does not even help a near-stationary client to find a different AP. In a heavily utilized network operating on 2.4 GHz, on the network-wide scale, frequent active scans inject redundant probe traffic that is low-rate, wastes airtime, and hinders the transmission of data traffic; thereby negatively impacting the goodput as we demonstrated with the case study.

In the next chapter, we present mechanisms to deal with probe traffic.

Chapter 3

Mitigating the Impact of Unnecessary Active Scans

3.1 Introduction

In the previous chapter, we validated that the stationary WiFi clients trigger active scans without considering a need of it, thereby hampering not only their own performance but even the performance of an entire network. In this chapter, we present our proposed mechanisms to deal with such scans. Begin with a metric that measures the growth of probe traffic in realtime in Section 3.2, we proceed towards developing a device-agnostic mechanism to infer the causes behind active scans in Section 3.3, and lastly, we propose a simple client-update that prevents stationary clients to trigger active scans when not needed, in Section 3.4.

3.2 A Metric to Detect the Growth of Probe Traffic in Realtime

Probe traffic may not be a reason behind the network's degraded performance. Interference [80, 81, 82], hidden terminals [83, 84], exposed terminals [85], rate adaptation [86] are amongst few other causes. A large number of WiFi devices exaggerate these causes. Therefore, we suggest detecting in realtime if probe traffic is reducing network performance in terms of lowered goodput before taking action against probe traffic.

We introduce the metric P/D to monitor the realtime increase in probe traffic and empirically evaluate its functioning. Our metric can be used to find out whether it is necessary to disable probing. We also present an empirically derived mathematical relation to demonstrate the exponential drop in MAC goodput due to excessive probe traffic.

3.2.1 The Metric P/D

3.2.1.1 Effect of Probe Frames on Data Frames

Figure 3.1 shows a relationship between number of P (Probe) and D (Data) frames. Instead of client specific P and D frames, we consider network-wide frames. Our network setting always had data to be sent. The data presented is averaged over 36 hours of captured traffic in an uncontrolled WiFi network.

For every 10 P frames/second injected in the network, each point on the plot is an average of the number of D frames transmitted in 1 second versus the

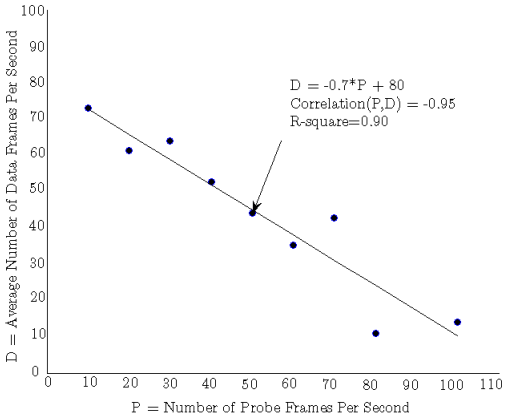


Figure 3.1: Effect of increased probe (P) traffic on data (D) traffic. The average number of D frames versus P frames transmitted in a second. D decreases by a slope of -0.7 with an increase in P, also P and D are inversely proportional with negative correlation coefficient -0.95.

number of P frames in that time slot. D frames decrease with a slope of 0.7 with an increase in P frames, as shown by Equation 3.1,

$$D = -0.7 * P + 80 \tag{3.1}$$

P and D frames are negatively correlated, with correlation coefficient equal to -0.95. Therefore, an increase in P implies a decrease in D. Since our network always had data to be sent, when P increases, channel contention also increases resulting in – (a) decrease in number of D frames, (b) loss of ACKs , (c) increase in data frame retries, and (d) Triggering of rate adaptation. All these, further result in a reduction of network $Goodput$.

3.2.1.2 Understanding the Metric P/D

To measure the relation of P and D in a time slot, T_s, we take the ratio of P and D, where considered data frames are fresh data frames *i.e.*, they are not retried

frames. Equation 3.2 shows a calculation for this metric,

$$P/D = \frac{N_p}{N_d} \tag{3.2}$$

where, N_p is the number of P frames including probe requests and probe responses and N_d is the number of fresh data frames in a T_s.

Assuming there is always data to be sent when $P/D < 1$ [Case 1], most of the D frames are able to get channel access and network $Goodput$ is not affected due to P frames. When $P/D = 1$ [Case 2], both P frames and D frames are able to win the channel contention in equal numbers. In this case, a lesser number of D frames are able to win the contention, thus network $Goodput$ starts reducing. Severe decrease in network $Goodput$ can be seen when the instances of $P/D > 1$ [Case 3] increase. This growth essentially means P frames are increased to the extent that D frames are unable to get the channel access. The rate of increase of $P/D > 1$ allows us to detect in realtime if $Goodput$ is reducing due to the increase in probe traffic. Table 3.1 summarizes these cases. We validated the equation 3.2 in two network settings (Academic building of IIIT-Delhi and Conference venue at IIT-Bombay).

We take a conservative approach for early detection of growth in probe traffic

Case	Metric	D frames	P frames	Probe Traffic ↑
1	$P/D < 1$	✓	✗	No
2	$P/D = 1$	✓	✓	No
3	$P/D > 1$	✗	✓	Yes

Table 3.1: Detecting the growth in probe traffic with metric P/D [D =Data frames, P =Probe frames, ✓ $\implies$ Able to access channel, ✗ $\implies$ Unable to access channel. Assumption: Clients always have data to transmit.] Case 3 $\implies$ probe traffic is the cause of goodput drop.

by considering only fresh data frames for the calculation of P/D. The rationale behind choosing only fresh data frames for the metric P/D is that in the ideal scenarios when there is minimal network contention, data frames are not retried. However, there is an increase in data frame retries with an increase in probe traffic due to the corresponding network contention. Considering retried data frames would not capture the actual network condition where the client is putting extra effort for frame retransmissions. Therefore, to effectively capture the impact of an increase in probe traffic on data traffic we do not consider retried data frames but fresh data frames only.

Empirically, we find relation between $Goodput$ and metric P/D, as given in Equation 3.3,

$$Goodput = a * e^{-b*P/D} \tag{3.3}$$

where, a, b are the constants specific to network under study. This equation is based on curve fitting, with the statistical coefficients for the goodness of fit as $SSE = 0.003338$, $R\text{-}square = 0.9966$, $Adjusted\ R\text{-}square = 0.9965$, and $RMSE = 0.008428$.

3.2.1.3 Using the Metric P/D

A counter is maintained for each of the 3 cases – (a) $P/D < 1$, (b) $P/D = 1$, and (c) $P/D > 1$. For each time slot T_s, the counter is incremented in a cumulative manner as per the case. We use $T_s = 1$ second in our analysis. The cumulative increase in the case of $P/D > 1$ for every T_s is plotted against time

for a duration of 1 minute, before the counters are reset. Data suggests this duration is long enough to diagnose the growth in probe traffic. Our analysis shows that with 90% accuracy when the slope of this plot is equal to or greater than 0.10, goodput starts reducing. Thus, the metric allows us to measure the growth of probe frames in realtime.

3.2.2 Evaluating the Metric P/D

Large-scale network deployments have a varying and large number of clients and APs that exaggerate the factors responsible for $Goodput$ drop of a WiFi network. Therefore, to show that an increase in the probe traffic results in reducing the network $Goodput$ we first perform a controlled experiment with a fixed number of clients and APs and controlled network traffic. Later, we demonstrate the drop in network $Goodput$ due to an excessive increase in probe traffic in an uncontrolled WiFi network. All the devices work in `802.11g` mode. We measure the growth in probe traffic in realtime with the metric P/D in both controlled and uncontrolled network setups.

3.2.2.1 Experiment Methodology

We inject UDP traffic over a WiFi link in the network and capture wireless traffic to measure $Goodput$ in presence and absence of excessive probe traffic. UDP traffic is injected to prevent the chances of transport layer re-transmissions. Nonetheless, TCP re-transmissions will further elevate the shown negative effect of probe traffic. `Iperf` client and server setup [87] is used for traffic

injection and bandwidth monitoring of the network. `Iperf` client is executing on a laptop with the WiFi link, sending data to an `Iperf` server executing on an Ethernet client.

3.2.2.2 Experiment Setup

(1) Controlled Network: Figure 3.2 shows this setup with 1 WiFi client, 1 AP, 1 ethernet client and 5 probing clients. Table 3.2 lists the specifications of these devices. The WiFi client is associated with the AP and the ethernet device connected to the ethernet port of the AP. `Iperf` client is executed on WiFi client and `Iperf` server is executed on ethernet client.

We initiate UDP data transfer from `Iperf` client to `Iperf` server for an hour. While this data transfer is going on, we initiate per second active scanning on 5 laptops. These laptops are not associated with the AP, hence do not transmit any data frames. We used 5 laptops for scanning because lesser than 5 had merely any effect on bandwidth. These laptops probe the network for 30

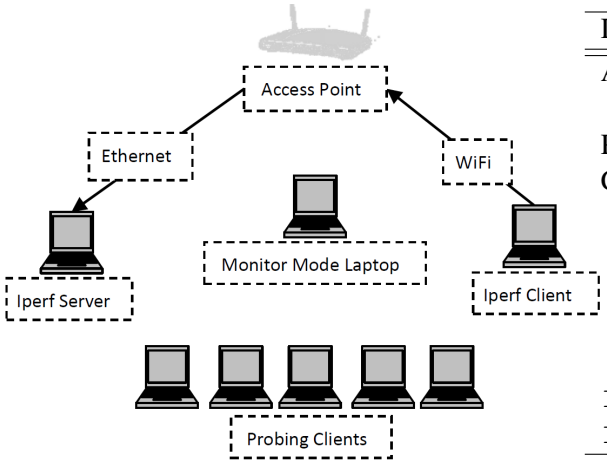


Figure 3.2: Experiment setup for controlled network environment to evaluate metric P/D.

Device Type	Model	Qty
AP	Wireless N-150, WGR614v10	1
Probing Clients	Intel Centrino Wireless-N 1030	1
	TP-Link TL WN721N	2
	Intel Corp. PRO/Wireless 5100	1
	Broadcom Corp. BCM4312	1
	Broadcom Corp. BCM4313	1
`Iperf` Client	Atheros Comm. AR9485	1
`Iperf` Server	Broadcom Corp. BCM5784	1

Table 3.2: Specifications of devices used in controlled network to evaluate metric P/D.

minutes before we turn off scanning. Along with our AP, their requests are also responded by neighborhood APs in the residential area. There were 2-4 APs responding to probe requests. Other APs are also far enough to cause any drop in our network's $Goodput$. In order to eliminate other causes of $Goodput$ drop, we disable rate adaptation on the `Iperf` client. Default bit-rate is 54 Mbps.

(2) Uncontrolled Network Controlled network setup had a fixed number of clients and APs, controlled data traffic, and rate adaptation disabled. The number of devices was lesser as compared to an uncontrolled WiFi network. The uncontrolled WiFi network that we considered is an enterprise Cisco WiFi network deployed in the university campus of IIIT-D. We considered a floor of one of the buildings that had 20 to 40 WiFi clients, up to 15 APs including non-enterprise APs, overlapping channels, rate adaptation enabled, and WiFi traffic not in our control. Given the scale of this deployment, we use a Wireless Intrusion Prevention System (WIPS) device as the sniffer. In this section, we demonstrate the effect of increased probe traffic in such a live network.

Figure 3.3 shows the floor map of the enterprise WiFi network, where we collected network traffic of the uncontrolled environment. Three floors of the building are shown here. We collected traffic on 3^{rd} floor B-wing section of the map shown. The WiFi network is deployed with a WiFi network controller, 1 WIPS device per floor, and 3 APs per floor. Every AP here is broadcasting up to 7 SSIDs. Along with enterprise APs, we notice the presence of non-enterprise APs in the network.

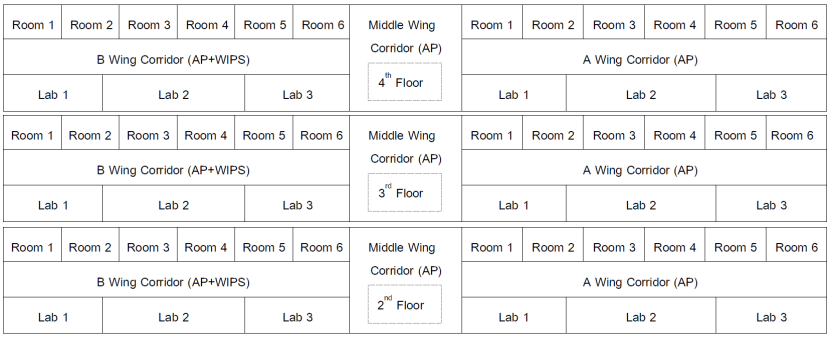


Figure 3.3: Building floor map with placement of APs and WIPS.

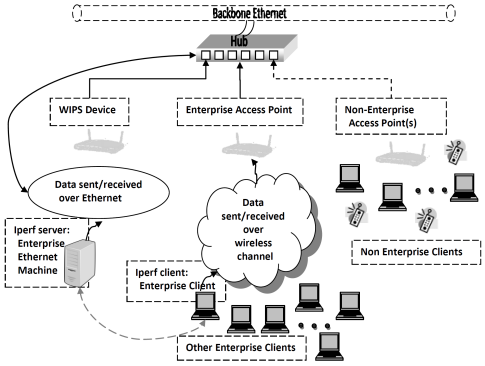


Figure 3.4: Experiment setup for uncontrolled network environment to evaluate metric P/D.

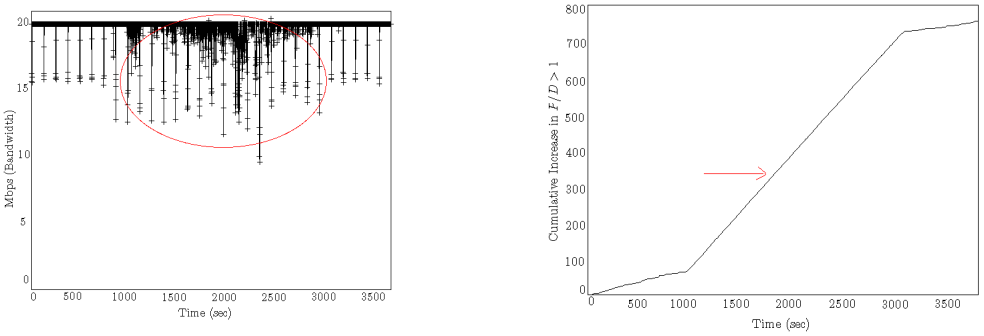
Device Type	Model	802.11
AP	Cisco Aironet 1140 Series	a/b/g/n
`Iperf` Client	TP-Link TL WN721N	a/b/g/n
`Iperf` Server	Ethernet	NA
WIPS device	Airtight Sensor SS-300-AT-C50/60	a/b/g/n

Table 3.3: Specifications of devices used in uncontrolled network setup to evaluate metric P/D.

Setup and specifications of these devices is shown in Figure 3.4 and Table 3.3, respectively. We analyzed an hour of traffic collected per day for 36 days over a period of 5 months. We tried to collect the data in peak hours of operation; usually, 10 AM to 3 PM in working days.

3.2.2.3 Results

(1) Controlled Network: Figure 3.5a shows network bandwidth as monitored by `Iperf` and Figure 3.5b shows the realtime detection of growth in probe traffic. This experiment demonstrates – (a) an increase in probe traffic reduces network *Goodput* and (b) metric P/D detects the growth of probe traffic in realtime. For the first 1000 seconds, active scanning is disabled on laptops; it is enabled



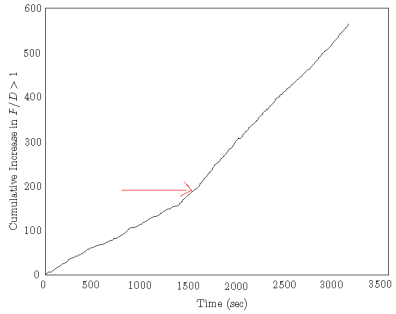
(a) Bandwidth monitoring of WiFi link in absence and presence of excessive probe traffic.

(b) Realtime detection of growth in probe traffic using metric P/D.

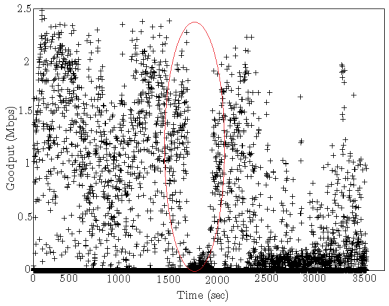
Figure 3.5: Demonstrating the effect of increased probe traffic in controlled setup. Figure 3.5a shows the drop in bandwidth from 1000 to 3000 seconds, when network experiences increase in probe traffic. Metric P/D in Figure 3.5b demonstrates the increase in slope from 1000 to 3000 seconds, when clients begin active scanning and decrease from 2700 seconds onwards, when active scanning is turned off at the clients.

for 1000 - 3000 seconds, and disabled again for last 1000 seconds. The rate of growth in P/D as seen from the plot is faster for 1000 - 3000 seconds. For this period, drop in bandwidth to as low as 10 Mbps from 20 Mbps is observed. ACK losses increase from 0.04% to 0.3% and frame retries increase from 0.8% to 4%. These factors ultimately result in increasing the variance of the reported bandwidth from ≈ 0.66 to ≈ 1.59; thereby, reducing the *Goodput* by upto 6%.

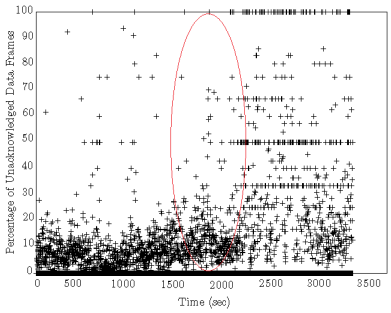
(2) Uncontrolled Network: We present here results of one of the 36 experiment days. However, all days who saw growth in probe traffic experience similar results. Figure 3.6a shows the change in slope of plot $P/D > 1$ with an increase in probe traffic. The network saw 162% growth in probe traffic. Growth in P frames results in decreased network *Goodput* as shown in Figure 3.6b. It drops from 2.5 Mbps to as low as 0 Mbps due to increase in unacknowledged data frames, which eventually result in frame retries and drop in frame data rates. Figure 3.6c and 3.6d show the growth in unacknowledged and retried data frames, respectively, which eventually drop data rates from 54 Mbps to as



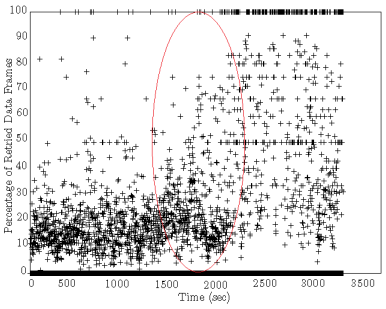
(a) Realtime detection of probe traffic growth.



(b) Drop in *Goodput*.



(c) Increase in ACK losses.



(d) Increase in frame retries

Figure 3.6: Demonstrating the effect of increased probe traffic in uncontrolled setup. Figure 3.6a shows the increase in slope of metric from 1500 seconds onwards, when probe traffic started to increase. Correspondingly, Figure 3.6b demonstrates the realtime drop in network MAC *Goodput* to as low as 0 Mbps from 1500 seconds with an increase in probe traffic, Figure 3.6c show unacknowledged data frames increase from approximately 10% to almost 100%, and Figure 3.6d show retried data frames increase from approximately 20% to 100%..

low as 1-2 Mbps. Table 3.4 shows the change in mean values of *Goodput*, ACK losses, and frame retries pre and post probe traffic growth.

We also analyze the APs, who respond to the probe requests. Table 3.5 summarizes the details of all responding APs, with averages calculated over all the experiment days. As seen from WIPS device, enterprise APs on the floors above and below across all 3 wings as well as non-enterprise APs also

Metric	Pre Probe Traffic Growth	Post Probe Traffic Growth
Goodput	0.82 Mbps	0.34 Mbps
ACK Losses	6.16%	16.00%
Frame Retries	12.80%	28.10%

Table 3.4: Measuring the mean impact of probe traffic growth in uncontrolled setup. Post probe traffic grows, goodput reduces by more than 50%, ACK losses increase by 2.6x, and frame retries increase by 2.19x.

Parameter	Value
Probe Responses per probe request	1-7
Responding enterprise APs	2-6
Responding non-enterprise APs	1-5
Locations of responding enterprise APs	3^{rd} Floor: B,M,A-Wing, 2^{nd} Floor: B,M-Wing, 4^{th} Floor: M,A-Wing
Number of enterprise APs whose channels overlap with other enterprise APs	0-5
Number of non-enterprise APs whose channels overlap with other enterprise APs	0-2

Table 3.5: Details of probe responses received during uncontrolled experiment.

respond to probe requests. The channels of enterprise APs also overlap with other enterprise APs as well as non-enterprise APs. Even the RRM algorithms on the controller, do not always assign non-overlapping channels to all the APs. Overlapping channels complicate the situation by introducing RF interference and increased probe responses due to overhearing [21].

The negative impact of active scans in 2.4 GHz motivates us to investigate an automated mechanism to infer the causes of active scanning. We present the mechanism in the next section.

3.3 Inferring the Causes

3.3.1 The Cause Inference Procedure

A sniffer often records frames sent over overlapping channels. For example, if its channel of operation is 6, in practice, it may record frames sent over channels $2-10$, which overlap with 6 [88]. As a result, a packet capture by a sniffer may end up with probe request(s) that were sent on overlapping channels. On the other hand, a sniffer may miss capturing frames due to overflows at its kernel's

queues say because it is busy logging frames that were captured, and when the frames arrive with low RSSI.

Given the above stated practical considerations, we do not look at particular instances of probe request(s) and/or probe response(s). Instead, for each client whose frames have been captured by the sniffer, we partition the captured active scanning frames corresponding to the client into non-overlapping *episodes* of active scanning. Specifically, if a probe request from a client is separated from its previous probe request in the capture by more than a second, we treat this probe request as the start of a new episode of active scanning by the client.

We derive metrics and simple rules, which are explained next, to infer the cause of a given episode of active scanning. The metrics are calculated using a *window* of frames that precedes the active scanning episode. This *window* extends to the end of the episode of active scanning, of the same client, that preceded the current episode. An example of active scanning episodes separated by a *window* is shown in Figure 3.7. We calculate the metrics for the client associated with the active scanning episode.

Recall that *Connection Maintenance* is performed by a client only if it is associated. Many possible causes of active scanning can thus be culled by merely

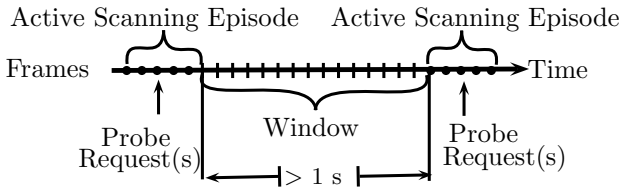


Figure 3.7: Episodes and windows for cause inference. Splitting a sniffer-based capture into active scanning episodes and windows for each client found in the capture.

knowing the association status of a client. In the absence of any messages in the sniffer-based capture that indicate that a client is associated (for example, data frames), we assume that the client is unassociated. In case some messages indicate a successful association, for all time following these messages in the sniffer capture, we assume that the client is associated. We change the status of the client to unassociated only if we find explicit deauthentication messages from the client or its AP.

The cause is deemed to be that of *Connection Establishment*, under which a client goes from an unassociated state to an associated state, if the episode of active scanning is preceded by at least one of the Association, Authentication, and Re-association requests. Note that one or more of these messages are a part of the procedure used by a client to associate to an AP [8]. The MAC layer header of captured frames is used to identify these requests.

Recall that *Connection Maintenance* on the part of the client involved monitoring its link to the AP. This involves monitoring the RSSI, frame drop rates, and beacon loss rates.

We calculate the average and the standard deviation of the RSSI for frames sent by the client in the *window* immediately before the probe request(s). The RSSI of a frame is obtained from the radiotap/prism header [89, 90] that is appended to every frame captured by the sniffer. Note that, since the sniffer is assumed to be placed close to the AP, the statistics of the RSSI as calculated by a sniffer for frames sent by a client are a good approximation of these values at

the AP. Hence, they are a good indicator of the signal quality of the link of a client to the AP. We declare the cause for active scanning to be *Low RSSI* if the average RSSI is less than -72 dBm and the standard deviation is greater than 12 dB.

We infer frame drops in the following three ways. First, we measure the number of frames sent by a client that are MAC layer retries. Information on whether a frame is a retry or not is obtained from the MAC layer header of the frame. This information is trustworthy as it is set by the MAC layer at the client sending the frame. Second, we estimate the number of data frames in the capture for which no ACK(s) were captured. We treat every such data frame as one for which the ACK was lost on the link between the AP and the client. Specifically, we assume that ACK(s) lost because the practical limitations of the sniffer explained earlier are negligible. Third, we monitor the PHY layer bit-rates, a function of the modulation and coding scheme used, of the frames. This is motivated by the fact that WiFi chipsets often downgrade the PHY rate in response to frame losses. As with the RSSI, the PHY rate of a frame can be reliably obtained from the radiotap/prism header. We declare the cause to be *Data Frame Losses* if frame retries increase by more than 50% or ACK losses increase by more than 50% or PHY data rates reduce by more than 50% during the *window* of frames preceding the active scanning episode.

Finally, under connection maintenance, we monitor loss of beacons. When the client detects the loss of beacons, it may send awake messages to the AP. It may also send directed probe request(s) with SSID of its AP. We declare the

cause to be the *Loss of Beacons* if these messages are captured before the active scanning episode. In addition, we declare it to be the cause if the time to next beacon exceeds the beacon interval (103 msec) for seven consecutive beacons during the window before the active scanning episode.

Connection Maintenance, in addition to monitoring, includes AP-side procedures and management of the power state of a client. If before an active scanning episode, a deauthentication message was sent to the client, then the cause of the active scanning episode is declared to be *AP-side procedures*. We do not infer the power state of a client. However, we declare that the client moved from a low power state to a high power state when the number of frames from the client increases from less than two per second to a higher value. The cause for the following active scanning episode is said to be *Power State: Low to High*.

Recall that *Connection Maintenance* happens only when the client is associated. Amongst the causes of it, we look to map one of them to the active scanning episode, starting with the cause *Low RSSI*, followed by *Power State: Low to High*, *AP-side Procedures*, *Loss of Beacons*, and *Data Frame Losses*. The first cause that gets associated is deemed to be the cause for the active scanning. Note that we first sample causes spanning the different facets of *Connection Maintenance*. The causes *Loss of Beacons* and *Data Frame Losses* are likely to correlate with *Low RSSI* and so are checked only in case the active scanning episode is not mapped to any of the other causes.

Finally, if an active scanning episode cannot be mapped to any of the above causes, we declare its cause to be *Periodic Scan*, which comes under the category of Discovery in Figure 2.3. We end this section by noting that the thresholds used to map causes to an active scanning episode were arrived at via experimentation.

We implemented the inference mechanism with a Deterministic Finite Automata whose details can be found in appendix A.

3.3.2 Evaluating the Cause Inference

To evaluate the efficacy of the proposed method to infer causes of active scanning we need to be able to compare the inferred cause of active scanning with its true cause. While WiFi datasets, created using sniffers are publicly available, none of them record the true cause of active scanning. We created a small experimental setup with the ability to introduce the different causes of active scanning in a controlled manner.

3.3.2.1 Experiment Methodology

We consider a WiFi setup with APs and clients. APs have the same SSID having an overlapping coverage area to ensure seamless handovers. This is a common setup of an enterprise WiFi network. We perform experiments in controlled and uncontrolled network setup. The experiments in controlled network setup allow us to evaluate the efficacy of the inference mechanism since the ground truth

about cause injection is known.

In the controlled network, we associate one client at a time and for each client introduce each of the causes. At the end of the experiment, we have a packet capture for each client, on which we use our proposed method to infer causes. The clients are instrumented to trigger active scanning once every 15 minute if they are associated and every minute when they are unassociated. This simulates behavior in line with the general observation that associated clients start *Periodic Scans* less frequently than the unassociated clients. The cause of *Connection Establishment* is introduced when a client goes from being unassociated to being associated to an AP. When a client is associated with an AP, the rest of the causes are introduced in turn as explained next. The cause *Power State: Low to High* is introduced by turning off the screen of the client followed by turning it on. *Loss of Beacons* is introduced by briefly turning off the AP to which the client is connected. *AP-side Procedures* is introduced by briefly changing the channel of the AP to which the client is associated and then restoring the channel. *Low RSSI* is introduced by moving the client far enough from the AP. *Data Frame Losses* is introduced by introducing another node that is hidden to the client and is also transmitting to the `iperf` server. Note that the triggers for causes *Loss of Beacons* and *Data Frame Losses* were chosen such that they don't trigger the reduction in RSSI.

We extend the application of the inference mechanism in uncontrolled network setups to understand the extent of causes in real networks.

3.3.2.2 Experiment Setup

The setup consists of two AP(s) – a Netgear JNR1010 and a TP-Link AC1900 – are placed in a lab in each others' vicinity. A sniffer that is a TP-Link device is placed near the AP(s), and it captures frames on the channel on which the AP(s) are configured to operate. We use four laptops with Atheros and Broadcom WiFi cards as clients. The device drivers were `ath9k` and `b43` for Atheros and Broadcom, respectively. The clients wirelessly send data to an `iperf` server [87] that is on a wired LAN shared with the AP(s).

3.3.2.3 Results

We quantify the efficacy of our proposed inference mechanism using the metrics of Precision, Recall, and $\mathrm{F}-\mathrm{Score}$ [91]. Let N be the total numbers of active scanning episodes per cause. Let T_p be the number of true positives, which is the number of episodes of active scanning for which the true cause was inferred. Let F_n be the number of false negatives, which is the number of episodes of active scanning for which the true cause was not inferred. Let F_p be the number of false positives, which is the number of episodes of active scanning for which the cause under consideration was wrongly inferred to be the cause of active scanning. We use the results of the controlled experiment to calculate T_p, F_n, and F_p, for each introduced cause. Precision in inferring a cause can be calculated as $P = T_p/(T_p + F_p)$. It is the fraction of all active scanning episodes inferred to have the cause under consideration, a total of $(T_p + F_n)$ episodes,

truly triggered by the cause. Recall can be calculated as $R = T_p/(T_p + F_n)$.

It is the fraction of active scanning episodes triggered by the cause under consideration that were mapped to the cause by the proposed inference mechanism. The larger the precision and the recall the better the inference. In addition to precision and recall, we use the $\mathrm{F-Score}$ to measure the accuracy of inference. It is given by $\mathrm{F-Score} = (2 * P * R)/(P + R)$. The larger the score, the better is the accuracy. Precision, Recall, and $\mathrm{F-Score}$, all have a maximum value of 1.

Table 3.6 summarizes the metrics defined above calculated over the four captures created using the controlled experiments. We collected several hundreds of episodes for all but, *Data Frame Losses*, for training machine learning models. We talk in detail about them in Appendix B. Every cause is identified correctly at least once as seen in the column for the number of true positives T_p in the table. However, for certain causes, we see false positives F_p and false negatives F_n. The values for recall R vary from 0.33 to 1.00 and the values for precision P vary from 0.22 to 1.00. While 3 out of 7 causes are detected with minimum $\mathrm{F-Score}$ of 0.80, 5 causes are detected with $\mathrm{F-Score} \geq 0.57$.

3.3.2.4 Cause Inference in Real WiFi Deployments

We use the proposed inference mechanism to infer causes of active scanning in our three datasets – [SIGCOMM], [IIT-B], and [IIIT-D].

Table 3.7 summarizes the causes of active scanning, for each of the datasets,

Cause	N	T_p	F_n	F_p	P	R	F − Score
Periodic Scan (Unassociated)	4	4	0	0	1.00	1.00	1.00
Periodic Scan (Associated)	2	2	0	3	0.40	1.00	0.57
Connection Establishment	5	3	2	1	0.75	0.60	0.67
Power State: Low to High	6	2	4	2	0.50	0.33	0.40
Loss of Beacons	4	2	2	7	0.22	0.50	0.31
AP-side Procedures	6	5	1	0	1.00	0.83	0.91
Low RSSI	3	2	1	0	1.00	0.67	0.80
Data Frame Losses	4	4	0	4	0.50	1.00	0.67

Table 3.6: Accuracy of the proposed inference mechanism. We separate the cause Periodic Scan into when the client is associated and not associated.

Cause	[SIGCOMM]	[IIT-B]	[IIIT-D]
Periodic Scan (Unassociated)	04.07	23.42	24.92
Periodic Scan (Associated)	53.33	47.65	65.00
Connection Establishment	00.00	00.00	00.01
Power State: Low to High	31.97	10.23	09.16
Loss of Beacons	00.00	00.00	00.03
AP-side Procedures	00.01	00.47	00.50
Low RSSI	00.00	18.08	00.00
Data Frame Losses	10.62	00.15	00.38

Table 3.7: Extent of Active Scanning in Real-world Datasets. Percentage of occurrence of the different causes of Active Scanning in sniffer-based captures of three real-world WiFi networks. We do not control the cause of active scans present in the real-world datasets.

as inferred by the proposed mechanism. Note that different networks have different dynamics. Even though each cause is equally likely to occur in a network, the distribution of causes in each network is different. A few insights that may be derived by a network administrator from the observed causes are as follows.

1. Active scanning due to *Periodic Scans* by unassociated devices is high for the datasets [IIT-B] and [IIIT-D]. This hints at the possibility of a large number of rogue clients in the network. These clients are not registered with the enterprise network but are actively looking for AP(s) for an association.

2. About 32% of active scanning is caused due to clients transitioning from a

low power state to a high power state *i.e.* due to cause *Power State: Low to High* in the [SIGCOMM] dataset. This may be explained by users of the network at the conference using their devices intermittently for short bursts of time.

3. In the [IIT-B] dataset, about 18% of the active scanning are due to *Low RSSI*. This indicates that some students in the classroom were close to the boundary of the AP(s) coverage. An administrator may resolve this by simply adding another AP in an orthogonal channel.

4. *Data Frame Losses* contribute to about 11% of the active scanning in the [SIGCOMM] dataset. This is likely because of the many users sharing the network competing for access to the shared wireless channel.

Next, we present a client-side mechanism to curb the generation of probe requests.

3.4 Approach to Reduce Active Scans

We observe that the WiFi clients unintelligently opt for active scanning by default; thereby, discarding passive scanning. We propose a modified scanning strategy that is motivated by the idea of interleaved active and passive scans proposed in [92].

In that work, the authors proposed to judiciously choose between active and passive scans based on the QoS requirements of the data to be transmitted. Their

goal was to reduce scanning delay to support the stringent time requirements of QoS data. We find following limitations of the presented work – (a) authors assume that the clients have apriori information about nearby APs; that may not be true always, for example, a new client in the network. A possibility is that the client triggers a few initial active scans to know nearby APs, however, that defeats the purpose of reducing active scans as we do not know how often such scans would be required, (b) a per-AP scanning schedule is prepared based on the list of APs (that are the known APs); that limits client to discover new APs, (c) the proposed scanning considers the data needs of the client but not the network condition; that may result in client triggering active scans due to *Connection Maintenance* cause, (d) authors do not discuss about how the proposed algorithm can be integrated with real WiFi clients, and (e) although there is still a possibility of reduced probe requests from the client, however the authors evaluated their algorithm in simulations, so its feasibility and the related performance enhancement in real WiFi networks is unknown.

We explain our approach in the next subsection.

3.4.1 The Modified Scanning Strategy

We propose to consider the mobility status of the client before taking an informed decision of active or passive scanning. The hypothesis behind our proposed strategy is that active scanning is required only when, the client is mobile and in all other cases passive scanning should suffice.

As per our proposal, of the three principal causes that trigger active scanning, only one cause - *Connection Establishment* should trigger active scans; this includes the case of handover as well. Further, as we showed in Chapter 2.3 most probe responses fetch redundant information about APs; thus active scans due to *Discovery* can be safely disabled for associated clients. For unassociated clients, we propose passive rather than active scans for *Discovery*, the reason being unassociated clients do not have any connection-specific strict time constraints. Lastly, during *Connection Maintenance*, a WiFi client needs to make sure that its AP is available. Periodic broadcast of beacons allows easy tracking of AP availability. Thus, passive scanning is sufficient for *Connection Maintenance*.

Figure 3.8 presents the implementation flow of the proposed approach. A client decides dynamically on the run-time which scan should be triggered. Passive scans are triggered either periodically when the WiFi client is unassociated or when the causes under *Connection Maintenance* occur. Active scans are triggered either during *Connection Establishment* or while a handover is happening.

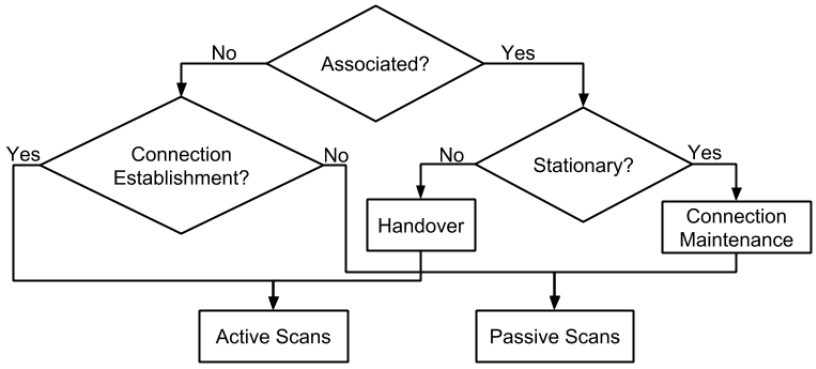


Figure 3.8: Modified scanning strategy to decide dynamically decide passive or active scanning on the runtime.

We mentioned earlier that passive scanning is slower than active. Therefore, an argument can be that the proposed scanning scheme might add significant latency to an ongoing connection. However, this will not be the case because firstly periodic scanning which is the most common cause (Average 72%), either passive or active, is altogether disabled and secondly, this scheme triggers passive scans during *Connection Maintenance* operation of an associated client while the client is immobile or stationary. Thus, passive scanning is triggered only in necessary conditions like *AP-Side Procedures* and *Data Frame Losses*. Unless such conditions are triggered, which is unlikely in a well established WiFi network, passive scanning will not trigger; thereby preventing unnecessary latency at the client.

3.4.2 Evaluating the Modified Scanning Strategy

3.4.2.1 Code Implementation

We instrument Ubuntu 16.04 machines to implement the above-discussed scanning strategy. All kinds of network managers were disabled on the machines. We modify `wpa_supplicant` [93] code to decide at runtime whether to trigger active or passive scans. This is done with the `bg_scan`'s `simple` module [94]. By default, we enable passive scans and disable active scans. Active scanning is enabled only when the RSSI value as seen by the client from AP dropped below -70 dBm, which is an empirically derived estimate. We do not control the probe requests generated by `mac80211` code or lower level device driver code since that defeats our purpose of developing a vendor-neutral solu-

tion.

3.4.2.2 Experiment Methodology

We evaluate the performance of the modified scanning strategy in a WiFi network that simulates the settings of a real WiFi network. We deploy APs with overlapping coverage, broadcasting the same SSID, and operating on the same channel for the same reason of enabling a smooth handover for clients. A sniffer, passively listening on the same channel as of APs and placed close to the APs, recorded the traffic. The network had a mix of clients with modified and unmodified scanning strategy. We begin with all the devices in the unassociated state. After an hour, we associated them with one of the APs. Every hour, we randomly introduced the causes that trigger active scanning. We hypothesize that modified clients will induce lesser probe requests than unmodified clients without affecting their connection.

3.4.2.3 Experiment Setup

In addition to enterprise-grade APs of IIIT-D WiFi network, we install two more APs that we could control. We collect the sniffer logs with TP-Link WN721N sniffer. The network has 20 WiFi clients, 10 with a modified scanning code and 10 with unmodified scanning code. They had WiFi chipsets from Atheros, Broadcom, MediaTek, Ralink, and Realtek. Duration of the experiment was 9-hour. Each client is instrumented to log the timestamps of various connection-related events such as association, authentication, scanning started, and scan-

ning ended. Note that this setup simulates a real WiFi network. We do not control any other network or client parameters.

3.4.2.4 Results

We evaluate the performance of the clients for #probe requests transmitted, the duration for which their connection persisted [54], and the time it takes to re-establish a broken connection.

#Probe Requests Figure 3.9 shows #probe requests as recorded by the sniffer. During the first hour, when all the clients were unassociated, unmodified clients transmitted 1184 probe requests while there were no probe requests from modified clients. Since WiFi network setup does not change throughout the experiment, all these frames induced redundant probe traffic in the network. For the following hours, unmodified clients always transmitted higher probe requests than modified clients. Modified clients sent a maximum of 50% and a minimum of 22.61% lesser probe requests than the unmodified clients.

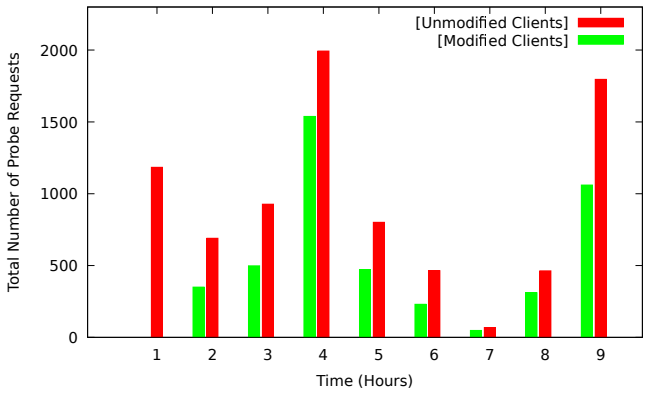


Figure 3.9: Reduction in the #probe requests with modified scanning strategy. Notice that in the initial hour, modified clients do not send any probe requests, while there are more than 1000 probe requests from unmodified clients. Even for the subsequent hours, unmodified clients continue to transmit much higher probe requests than modified clients.

Network Discovery Clients under observation are near-stationary throughout the experiment, except during the induction of the cause *Low RSSI*. Therefore, the APs discovered at the clients remain to be the same. Our modified scanning strategy deliberately enables active scanning on *Connection Establishment*, so there is a possibility of inherent switching delay that occurs due to the transition from passive to active scanning. Thus, we analyze the delay to connection setup by analyzing the time elapsed between disconnection and connection. We refer to this as Time To Connect.

Time To Connect We calculate the time to connect by analyzing the timestamps recorded for various events at the clients. Typically, when a disconnection has triggered a series of events as recorded at the client-end are disconnected, scanning, and connected. In our case, the scanning is either active or passive. The difference of timestamps recorded for disconnected and connected events give us the value of time to connect. Figure 3.10 presents the CDF of time to connect. We observe that for the modified clients time to connect is much lesser

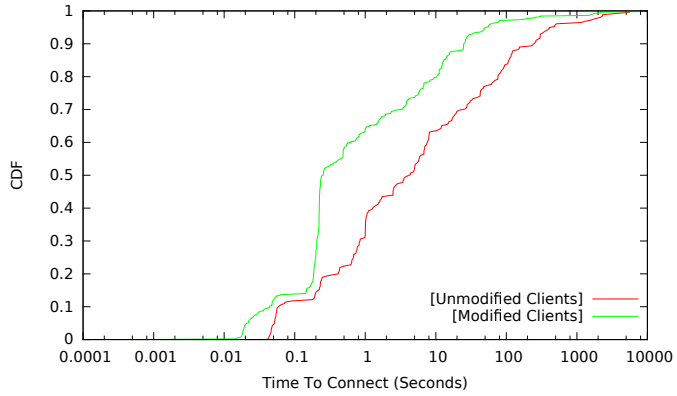


Figure 3.10: Time to connect with modified scanning strategy. Median time to connect for unmodified clients is 4.35 seconds, whereas for modified clients it is 0.25 seconds. Even 75^{th} percentile for unmodified clients is 43 seconds, whereas for modified clients it is 5.7 seconds.

than the unmodified clients. In fact, at the 50^{th} percentile the time to connect is mostly constant at approximately 0.25 seconds for modified clients, whereas for unmodified clients it goes up to 4.35 seconds.

The reason for much lower time to connect is that the modified clients are not involved in unnecessary active scans, and hence, its transmission queues are relatively less loaded. Clients dynamically decide the type of scan. Stationary clients trigger passive scans. As seen from the data in Chapter 2, the network deployment *i.e.*, the APs remain unchanged for stationary clients. The network information gathered from passive scans remains in the history of the client. The client uses this information to connect with the AP. The information is updated for new and different APs only with the active scans when the client is mobile and/or establishes a new connection.

Connection Persistence Once the client is connected, transmitting lesser probe requests does not deter the performance of WiFi connection. We confirm this with our next metric that measured the duration of connection persistence. We measure this duration with the time elapsed between 2 successful connection establishments as recorded at the client. For the entire length of the experiment, modified clients persistently showed better connection persistence, as shown in Figure 3.11. We investigate the probabilities for 1 to 225 minutes of association. The modified clients continually demonstrate higher chances of staying associated than the unmodified clients for all durations considered.

When we induce the causes such as *Data Frame Losses* and *AP-side Proce-*

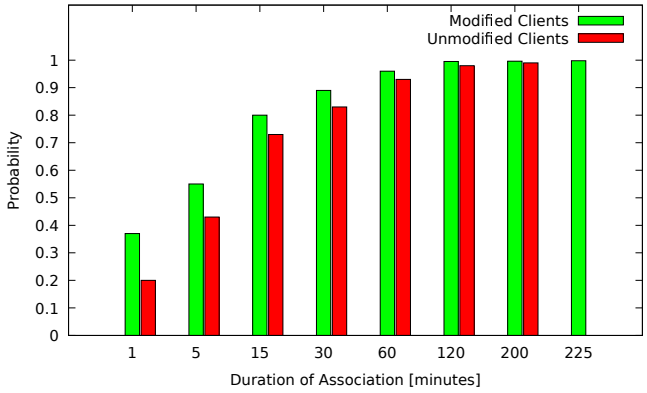


Figure 3.11: Evaluating the connection persistence with modified scanning strategy. Notice that modified clients continue to stay connected with higher probability. Modified clients show the maximum duration of association being 1196 seconds higher than unmodified clients.

dures, driver code of few unmodified froze the complete system, and we had no other option than to restart the machine. This behavior shows that unmodified clients had the intolerance towards a few causes. This behavior is one of the reasons behind a lower association duration of unmodified clients.

With our scanning strategy, we successfully demonstrate that stationary clients need not perform unnecessary active scans. Even without it, the WiFi connection did not suffer, but is maintained in a much better manner. Last but not least, implementation of the proposed strategy can be easily rolled out to devices as a simple application update.

As a limitation of this strategy, stationary clients will experience higher delays to the ongoing connection if losses increase at alarming rates. This is because passive scanning will be triggered due to *Connection Maintenance* cause.

In the next chapter, we study the impact of minimal active scans on the accuracy of fingerprint-based indoor WiFi localization and the possible methods to improve accuracy.

Chapter 4

WiFi Indoor Localization using Existing Infrastructure in the presence of Minimal Active Scans

4.1 Introduction

In the previous chapters, we argued that there is a need for automated and device-agnostic mechanisms to deal with extensive active scans by stationary WiFi clients in highly-congested WiFi networks. In this chapter, we show that an arbitrary reduction in active scans negatively affects the performance of services that leverage WiFi. Specifically, we focus on WiFi-based indoor localization for smartphones on a campus. Leveraging a large-scale deployment of the enterprise WiFi network provided by Aruba in Singapore Management University, we provide the evidence that the clients scan minimally in well-established WiFi networks. Such networks have a low network contention and they primar-

ily operate in 5 GHz. The minimal active scans not only result in a reduced number of APs reporting for a client, but even a mismatch in the source of RSSI. As a result, minimal active scans adversely impact the performance of localization. We describe the setup, the localization technique, and the challenges encountered in Section 4.2. Further, we show the impact of the minimal active scans on the accuracy of WiFi-based indoor localization and propose heuristics to alleviate the errors encountered while localizing the clients in Section 4.3.

4.2 The Server-Side Indoor Localization

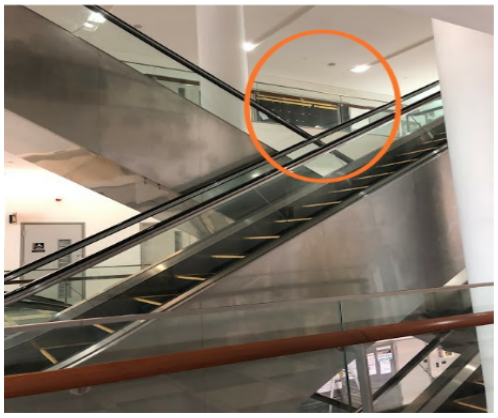
In this section, we present the details about system architecture and the dataset.

4.2.1 The Dataset Details

4.2.1.1 Background and Deployment

This work began in 2013 when we started deploying a WiFi-based localization solution across the entire campus. It has since gone through much major and minor evolutions. However, in this work, we focus our evaluation and results in just one venue – a university, as we have full access to that venue. Our university campus has seven schools in different buildings. Five buildings have six floors, remaining two have five and three floors respectively, with a floor area of $\approx 70,000\ m^2$. Landmarks, characterized by water sprinklers are deployed every three meters, on a given floor denote a particular location. Figure 4.1 shows images of the landmarks. There are 3203 landmarks across thirty-eight

(a) Landmarks as the water sprinklers on the ceiling



(b) Landmarks in the overlapping regions

Figure 4.1: Characterizing the landmarks with water sprinklers for localization. The landmarks are highlighted in an orange circle.

floors of seven schools. WiFi network deployment includes 750+ dual-band APs, centrally controlled by eleven WiFi controllers, with $\approx$ 4000 associated clients per day.

4.2.1.2 System Architecture

Figure 4.2 represents the primary building blocks of the system. The system is bootstrapped with APs configured by the WiFi network controller to send RTLS data feeds every five seconds to the RTLS server. Most commercial WiFi network infrastructures allow such a configuration. Once configured, APs bypass WiFi network controller and report RTLS data feeds directly to our Location Server. Table 4.1 presents all the fields contained in an RTLS data feed per client. The reported RSSI value is not on a per-frame basis, but a summarized value from multiple received frames. The Location Server analyzes these RTLS data feeds for the RSSI reported by different APs to estimate the location of a client. Note that the APs do not report the type of frames. They gather informa-

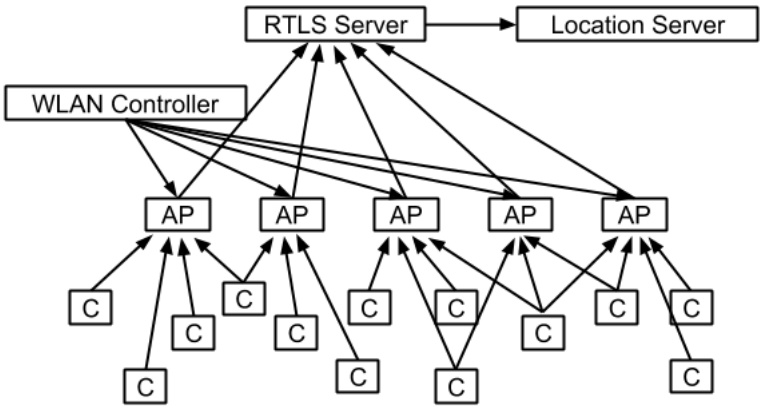


Figure 4.2: Block diagram of Indoor Localization System. Note that lines between AP and C denote the coverage area of AP and not the association. *Legend: AP - Access Point, C - Client, WLAN - Wireless LAN, RTLS - Real-Time Location System*

Field	**Description**
Timestamp	AP Epoch time (milliseconds)
Client MAC	SHA1 of original MAC address
Age	#Seconds since the client was last seen at an AP
Channel	Band (2.4 GHz or 5 GHz) on which client was seen
AP	MAC address of the Access Point
Association Status	Client's association status (associated/unassociated)
Data Rate	MAC layer bit-rate of last transmission by the client
RSSI	Average RSSI for duration when client was seen

Table 4.1: Details of RTLS data feeds

tion from their current channel of operation and scan other channels to collect data. Vendors have microscopic details of what APs measure [95], however as an end-user we do not have access to any more information than what is specified. Nevertheless, even this information at large-scale gives us a view of the entire network from a single vantage point.

4.2.1.3 Recording of the Fingerprints

We define a fingerprint as a vector of RSSI from APs for a given client. We consider two types of fingerprints – offline and online. An offline fingerprint is collected and stored in a database before the process of localization is boot-

strapped, while an online fingerprint is collected in real-time.

Offline Fingerprinting A two-dimensional offline fingerprint map is prepared for each landmark on the per-floor basis. The client devices used for fingerprinting were dual-band Android phones, which were associated with the network, and they actively scanned for APs. For each landmark, the device collected data for five minutes. While the clients scan their vicinity, APs collate RSSI reports for the client and send their measurements as RTLS data feeds to the Location Server. For a given landmark L_i, an offline fingerprint takes the following form:

$$< L_i, B, AP_1 : RSSI_1; ...; AP_n : RSSI_n; > \tag{4.1}$$

We maintain fingerprints for both 2.4 GHz and 5 GHz frequency bands. Band B, in the above equation, takes a value of band being recorded. The vectors are stored in a database on the Location Server.

Online Fingerprinting Localization of a client is done with online fingerprints. An online fingerprint takes the same syntax as offline fingerprints in Equation 4.1, except the landmark, as shown below:

$$< B, AP_1 : RSSI_1; ...; AP_m : RSSI_m; > \tag{4.2}$$

Now, we match this online fingerprint with offline fingerprints of each landmark to calculate the distance in signal space, as discussed in [28]. The landmark with minimum distance in signal space is reported as the probable location of the client.

Now, we present the details of data collection and its processing.

Collection of the Ground Truth We collect the ground truth data for online fingerprints. We want to correlate the data collection with real-world usage scenarios. Therefore, we choose the four most common states of WiFi devices as per their WiFi association status and Data transmission. The states are – (i) Disconnected, (ii) WiFi Associated – ($ii.a$) Never actively used by user, ($ii.b$) Intermittently used, and ($ii.c$) Actively used. These states implicitly modulate the scanning frequency. We use a separate phone for each state; thus, we use 4 Samsung Galaxy S7 phones to record ground truth for each landmark.

We define each of these states as follows:

1. State (i) The client is disconnected. In this state, WiFi is turned on but not associated with any AP and screen remains off throughout the data collection. Therefore, only traffic generated from this client is scanning traffic and no data traffic. We ensure that this client did not follow MAC address randomization, which latest devices follow in the unassociated state [96].

2. State ($ii.a$) The client is associated but inactive. In this state, WiFi is turned on, it is associated but the screen remains off throughout the data collection.

3. State ($ii.b$) The client is associated and intermittently used. In this state, WiFi is on, the client is associated and user intermittently uses the device. This is one of the most common states for mobile devices, and our previous work of sniffer based inference found that scanning is triggered whenever the screen of the device is lit up.

4. State ($ii.c$) The client is associated and actively used. In this state, WiFi is on, the client is associated, and a YouTube video plays throughout the data collection. This state generates most data traffic, *i.e.* non-scanning traffic.

Each client stayed at a landmark for about a minute before it moved to the next landmark. We manually noted down start time and end time for every landmark at the granularity of seconds. We did this exercise for 3203 landmarks of our university, collected 86 hours worth of data, that accounts for $54,096$ files carrying 274 GB of data.

The amount of time to localize a client is 40 seconds. Processing the entire dataset would take ≈ 100 days. Therefore, given the size of the entire dataset, we present our analysis of 200 landmarks, which accounts for 3121 files with 15.3 GB of data. Figure 4.3 shows the floor map of one of the schools whose data we refer for our analysis.

We aim to demonstrate the challenges associated with fingerprint-based localization. These challenges apply to all the solutions that employ fingerprint-based localization, irrespective of the type of device present in the network. The

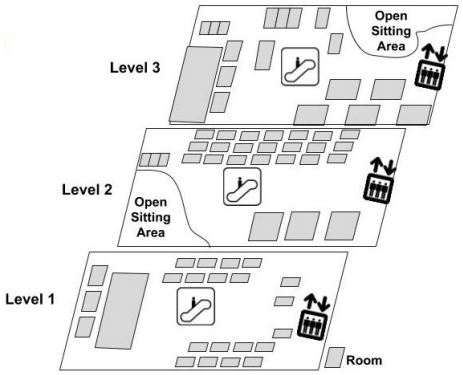


Figure 4.3: Floor map of the school where we collected ground truth data.

variation of RSSI with device heterogeneity is well-known [97] and that will further exacerbate the problems identified by this paper. We collect ground truth with only one device so that we can highlight issues without any complications added by heterogeneous devices.

Pre-processing of the RTLS Data Feeds Our code reads every feed to extract the details of APs reporting a particular client. RTLS data feeds, may obtain stale records for a client. Therefore, we filter the raw RTLS data feed for the latest values, with age less than or equal to 15 seconds, and the RSSI should be greater than or equal to -72 dBm. The threshold for age is a heuristic to take the most recent readings. The threshold for RSSI is decided based on the fact that a client loses association when RSSI is below -72 dBm.

For our analysis, we classify MAC layer frames in two classes (a) *Scanning Frames* – high power and low bit rate probe requests and (b) *Non-Scanning Frames* – all other MAC layer frames. Offline fingerprints are derived from the scanning frames, which are known to provide accurate distance estimates as they are transmitted at full power. In the offline phase, a client is configured to scan continuously. However, in the online phase, we have no control over the scanning behavior of the client, resulting in a mix of scanning and non-scanning frames. Therefore, while localizing with the fingerprints, RSSIs available for matching are from the different categories of frames. RTLS data feeds do not report the type of frame and do not have a one-to-one mapping of MAC layer frames to the feeds. Therefore, we devise a probabilistic approach to identify these frames.

We design with a set of controlled experiments, where we configured the client in one of the two settings at a time – (a) send scanning frames only and (b) send non-scanning frames only. These two settings are mutually exclusive. We collected the traffic from the client with a sniffer as well as the corresponding RTLS data feeds. Then, we compare both the logs – sniffer and RTLS, to confirm the frame types and the corresponding data rates.

Our analysis reveals that when a client is associated and sending non-scanning frames, the AP to which it is associated reports the client as *associated*. The data rates of the RTLS data feeds vary among various $802.11g$ rates, e.g., 1, 2, 5.5, ...,54 Mbps. Even though our network deployment is dual-band and supports the latest 802.11 standards including $802.11ac$, still the rates reported in the RTLS data feeds follow 802.11g. We do not have any visibility in the controller's algorithm to deduce the reason for this mismatch in the reported data rates. However, when a client sends scanning frames, all the APs that could see the client report the client as *unassociated* moreover, the data rates reported is fixed at either 1, 6, or 24 Mbps, as per the configured probe response rate.

We use these facts to differentiate non-scanning and scanning RTLS data feeds. We believe this approach correctly infers scanning frames because – (a) the data rates are fixed to 1, 6, or 24 Mbps, (b) when an associated client scans, other APs report that client as unassociated, and (c) an unassociated client can only send either scanning or association frames.

However, our approach may still incorrectly identify a scanning frame as

non-scanning in the following cases –

(a) When an associated client scans and the AP, to which it is associated, reports. This AP reports the client as associated and its data rate as 1, 6, or 24 Mbps. In this case, these rates may also be because of the non-scanning frames. We identify such feeds as non-scanning.

(b) When an unassociated client sends association or authentication frames. In this case, also, the rates overlap with the scanning data rates and the association status is reported as unassociated. Here, we incorrectly identify non-scanning frames as scanning frames.

However, these cases are rare. For other cases, our approach is deterministically correct.

4.2.2 Challenges Discovered

In this section, we give evidence of the issues, namely Cardinality Mismatch and High Client Scan Latencies. We compare the severity of these issues for both the frequency bands. We identify the causes behind these issues and measure their impact on the issues.

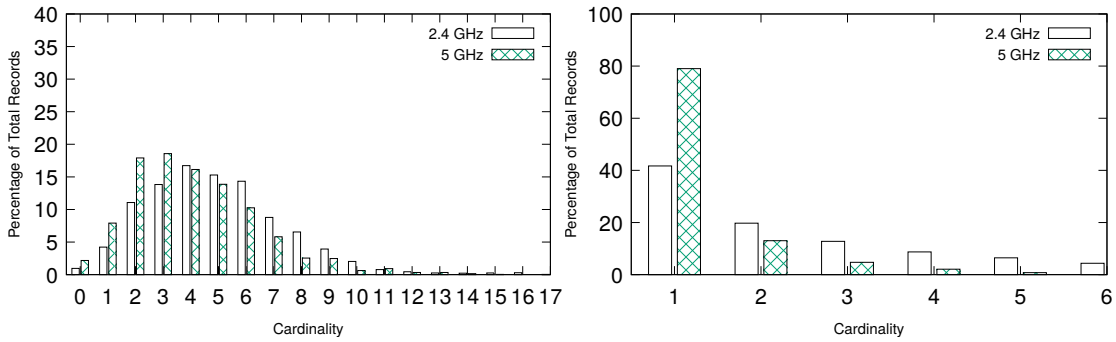
4.2.2.1 Evidence of the Issues

The Cardinality Mismatch arises from the dynamic power and client management performed by a centralized controller as well as the client-side power management. Given the dynamic nature of these management policies, it is not

possible to estimate their implications on the Cardinality Mismatch, so thereby on the localization errors. We take an empirical approach to see whether – (a) we can find out the severity of these implications on the Cardinality Mismatch and the localization error and (b) identify the tunability of the implicating factors.

Figure 4.4 plots the differences in cardinality between the offline and online phases for 2.4 GHz and 5 GHz. Figure 4.4a shows the cardinality observed in our offline fingerprints. Figure 4.4b shows the cardinality observed during the online phase. Note that, as explained in the pre-processing phase (Section 4.2.1.3), the cardinality reported in the online phase is based on a mix of scanning and non-scanning frames as opposed to that of the offline phase where the fingerprints are solely obtained from scanning frames. While the maximum cardinality is 16 during the offline phase, it is merely 6 in the online phase. This shows the spectrum of the Cardinality Mismatch.

In the online phase, 80% of the time only 1 AP reports for a client in 5 GHz



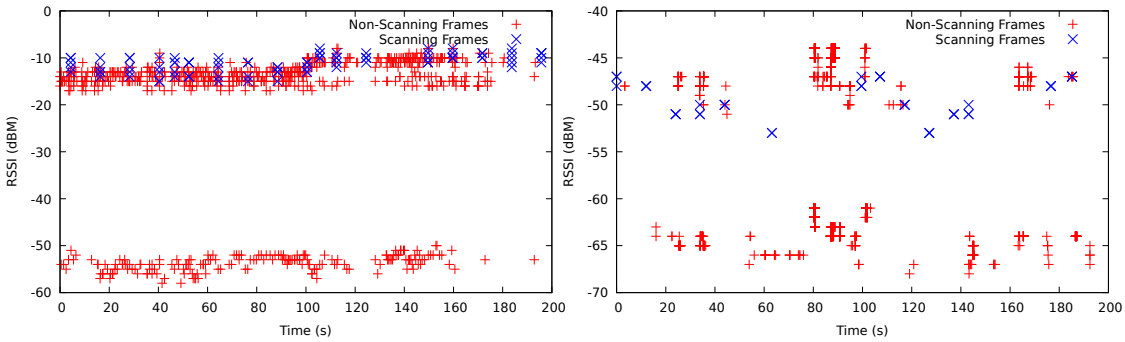
(a) Offline Phase (b) Online Phase

Figure 4.4: Cardinalities observed during the offline and online phases. For both the phases, the cardinalities are lower for 5 GHz. During the online phase, there is a substantial decrease in the cardinality for both 2.4 GHz and 5 GHz bands as compared to the offline phase.

while 40% in 2.4 GHz. Any fingerprint-based algorithm will be adversely affected by such a big difference in the cardinality. For each band, we find out how much is the extent of the Cardinality Mismatch. Overall, across all the cardinalities, 2.4 GHz has 57.30% mismatches and 5 GHz has 30.6% mismatches. The 5 GHz band is more adversely affected by the Cardinality Mismatch issue because it experiences lower cardinality, which increases the chances of a mismatch. Overall, 2.4 GHz always sees higher cardinality than 5 GHz, both during the offline and online phases. This is because signals in 2.4 GHz travel farther than that of 5 GHz. However, it is not the only reason. The other reason is the number of scanning frames transmitted. Unlike the data frames, the scanning frames are broadcasted and hence heard by more number of APs. As the number of scanning frames increases, more APs hear them and revert, thereby increasing the cardinality.

Besides, the RSSI variation for the scanning frames is lesser compared to that of the data frames. To validate, we perform a controlled experiment with a stationary client and collect client's traffic using a sniffer. The client has an ongoing data transmission and periodic scanning is triggered every 15 seconds. From the sniffed packet capture, we extract per-frame RSSI. The experiment is repeated for two scenarios- (a) the client is close to the AP and (b) the client is far from the AP. With these two scenarios, we simulate the client behavior for low and high RSSI from the AP.

Figure 4.5 shows the RSSI measurements in two cases. In the first scenario, when the client is close to an AP, RSSI of the scanning frames varies by up to



(a) Client Close To AP

(b) Client Far From AP

Figure 4.5: Variations in RSSI for scanning and non-scanning frames in two scenarios – (a) client close to the AP and (b) client far from the AP. For both cases the RSSI from scanning frames vary far lesser than the non-scanning frames.

10 dB and for non-scanning frames, it varies by up to 50 dB. Similarly, in the second scenario, when the client is far from AP, RSSI of scanning frames varies by up to five dB, and for non-scanning frames, it varies by up to 30 dB. Both our experiments validate that the RSSI from scanning frames vary far lesser than the non-scanning frames. This means the online RSSI from scanning frames match more closely and is a much more reliable indicator of the client's position. We want to study how clients in their default configuration behave in *real* networks; therefore we do not modify the default behavior of client driver in any way. We repeated the experiment with devices of Samsung, Nexus, Xiaomi, and iPhone.

Next, we study the effect of the band on the frequency of scanning. We collect WiFi traffic with sniffers listening on the channels in operation at that time in both the bands for 6 hours. Data from 200 WiFi clients is recorded. Figure 4.6 shows the plot. For both 2.4 GHz and 5 GHz bands, the frequency increases as RSSI reduces. Overall, the frequency is lesser for 5 GHz, even though most ($\approx 2X$) of the clients in our network associate in 5 GHz. More the frequency of scanning, lesser is the chance of Cardinality Mismatch. Our com-

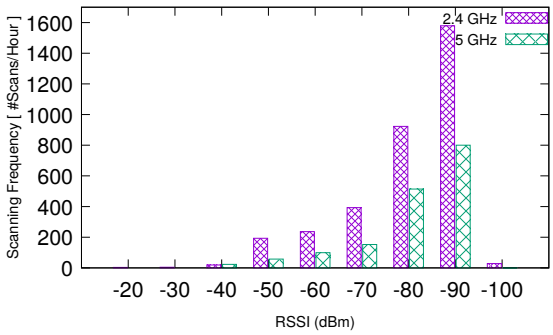


Figure 4.6: Frequency of scanning in both bands increases as RSSI reduces. 2.4 GHz experiences higher scan frequencies.

parative analysis of the two bands revealed that the instance of frame losses and poor connection quality, which cause scanning, are much lower in 5 GHz due to lower interference. The analysis of the scanning behavior of our clients reveals that – (a) 90^{th} %ile values of scanning intervals is in the order of few 1000 seconds, which is a lot for fingerprint-based solutions, (b) 5 GHz is the least preferred band of scanning, and (c) clients rarely scan both the frequency bands. Hence, we rule out the possibility of the reduced range of 5 GHz resulting in lesser scanning frames.

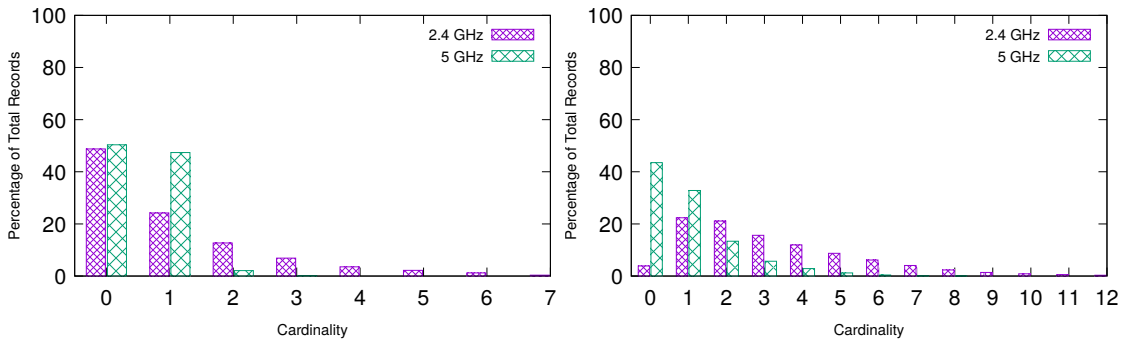
4.2.2.2 Causes Behind the Issues

Next, we study the combined effect of frequency of scanning, *i.e.*, number of scans per hour, and transmission distance on the cardinality. For this, we consider clients configured in one of the four states. Note that each state implicitly controls the amount of scanning. We do not manually control scanning behavior to imitate the real world. In the absence of client scans, APs get only non-scanning frames. For each of the four states of the client, we study how many APs report that client *i.e.* the cardinality. With this analysis, we are able

to compare the cardinality in the presence and absence of scans, for both 2.4 GHz and 5 GHz bands.

We see that as the frequency of scanning increases, more number of APs respond and the cardinality increases. We don't show the results for individual states of the clients, as the trend remains the same. Figure 4.7 shows the aggregated results. Note that, the Figure 4.7 specifically segregates the online cardinality for the presence and the absence of scans, as opposed to the data in Figure 4.4b, that showed the combined online cardinality. The cardinality is consistently higher for 2.4 GHz than that of 5 GHz. This implies that a higher frequency of scanning possibly reduces the Cardinality Mismatch and vice-versa.

However, a downside for 2.4 GHz is that the frames (both scanning and non-scanning) have higher variation in RSSI. This means, even though the extent of the Cardinality Mismatch is lower, the RSSI will differ more in 2.4 GHz. To confirm this, we analyze the RSSI from a stationary client by enabling its association in one band at a time and disabling the other band altogether. We



(a) Client Scans Absent

(b) Client Scans Present

Figure 4.7: Cardinality in the absence and presence of client active scans. More APs report during scanning.

use the RSSI recorded at the RTLS server for a duration of 1 hour.

As shown in Figure 4.8, even with more scanning information available 2.4 GHz is more prone to RSSI fluctuations than 5 GHz. The reasons for this behavior are – (a) the range of 2.4 GHz is almost double than that of 5 GHz and (b) a lesser number of non-overlapping channels makes it susceptible to interference. Therefore, RSSI from 2.4 GHz results in predicting distant and transient locations. We validate this in different locations with devices of four other models.

To summarize, there is a significant extent of Cardinality Mismatch and High Client Scan Latency. There is a difference in the extent of the issues for the two classes of frames and the two bands of operation. While scanning frames have a longer distance of transmission and less variation in RSSI, they are not often sent by the clients. The factors favoring 2.4 GHz are the longer distance of transmission and higher frequency of scanning. However, low variation in RSSI works in favor of 5 GHz. We summarize these observations in Table 4.2.

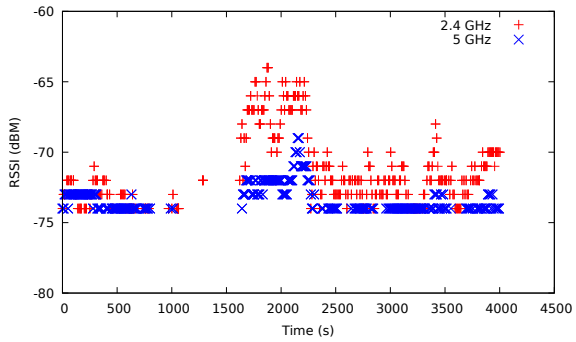


Figure 4.8: Variations in RSSI in 2.4 GHz and 5 GHz for a stationary client as measured at the server. Notice that 5 GHz is relatively stable than 2.4 GHz.

4.3 Reducing the Localization Errors

. Next, we propose the heuristics to reduce the localization errors.

4.3.1 The Proposed Heuristics

As seen in Table 4.2, the causes are conflicting to each other. Neither 2.4 GHz nor 5 GHz has all the causes in its favor. Therefore, getting rid of the two issues is not trivial. For improvement, we take a position to make the best use of whatever RSSI we receive during the online phase. The idea is to restrict the search space of landmarks from the entire building to a particular floor. For this purpose, we propose heuristics to, first select the floor from the APs reported during the online phase and then localize the client within that floor.

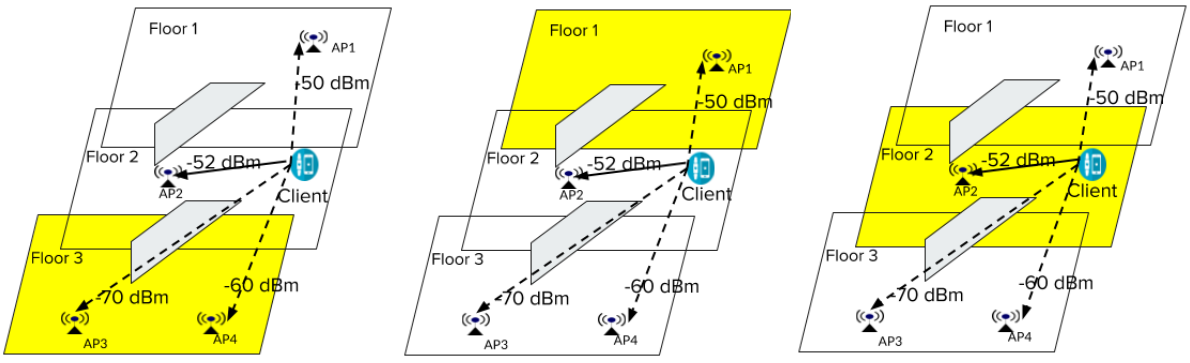
Note that, we already know the location of each AP. We use this information to shortlist the APs from the online fingerprint. Our localization algorithm first selects a floor, shortlists all the landmarks and the APs that are located on the same floor. We use the shortlisted APs to find a match with offline fingerprints of the landmarks on the floor. For selecting the floor, we explore three heuristics

Frames	**Transmission Distance**	**RSSI**	**Variation**	**Frequency of Transmission**
Scanning	*High* – ✓		*Low* – ✓	*Low* – ✗
Non-Scanning	*Low* – ✗		*High* – ✗	*High* – ✓
Band	**Transmission Distance**	**RSSI**	**Variation**	**Frequency of Scanning**
2.4 GHz	*High* – ✓		*High* – ✗	*High* – ✓
5 GHz	*Low* – ✗		*Low* – ✓	*Low* – ✗

Table 4.2: A summary of the causes that impact the accuracy of WiFi-based indoor localization. (✓- Reduces Localization Errors, ✗- Increases Localization Errors). Causes conflict with each other, making server-side localization non-trivial.

– (a) Maximum Number of APs - the floor for which the maximum APs are reporting the client, (b) AP with Maximum RSSI - the floor from which the strongest RSSI is received, and (c) AP of Association - the floor of AP to which the client is presently associated with.

Figure 4.9 shows an example that demonstrates the usage of each of the heuristics. The client is located at floor 2 and associated with AP2. In this example, we assume that when the client scans, all the APs from floor 1, 2, and 3 respond to the client. As shown in Figure 4.9a, 2 APs report the client from floor 3 while, a single AP reports from the remaining two floors. Hence, the heuristic – Maximum Number of APs detects floor 3 as the floor of the client. Next, the RSSI from AP1 on floor 1 is -50 dBm, RSSI from AP2 on floor 2 is -52 dBm, and RSSI from AP3 and AP4 on floor 3 is -70 dBm and -60 dBm, respectively. The case is shown in Figure 4.9b. Since, the maximum or the strongest RSSI is received from floor 1, floor 1 is detected as the floor of the client with the heuristic – AP with Maximum RSSI.



(a) Maximum Number of APs (b) AP with Maximum RSSI (c) AP of Association

Figure 4.9: An example to demonstrate the usage of floor detection heuristics. The client is associated with AP2 on floor 2. Selected floor as per the respective heuristic is highlighted. Maximum Number of APs (2) detects floor 3 as the floor of the client, AP with Maximum RSSI, detects floor 1 as strongest RSSI is received from the floor, and lastly, AP of Association detects floor 2, as the client is associated with AP on floor 2.

Lastly, since the client is shown to be associated with AP2 on floor 2, the heuristic – AP of Association detects floor 2 as the floor of the client. We show this in Figure 4.9c.

We implemented a server-side localization using well known fingerprint-based method [28]. Since we use server-side processing, we do not require any client-side modification. Our proposals do not make assumptions about the hardware or OS of the clients or the controller. Although each adaptation of fingerprint-based technique from the existing body of work may result in different errors, our exercise gives us a baseline that cuts across all the adaptations. The 2.4 GHz and 5 GHz bands differ in distance of transmission, variation in RSSI, and frequency of scanning. We measure the localization errors for both the bands.

We report localization errors for each value of the cardinality in the online phase to understand how the error varies as a function of the cardinality. We measure the errors in terms of (a) Different Floor and (b) Same Floor errors. Different Floor error is the percentage of total records for which a wrong floor is estimated. These are seen at the higher percentiles. For the rest of the records, the Same Floor error is the distance in meters between the actual and the predicted landmark on the floor. The errors at the higher percentiles are essential for security applications, for example, an error by a floor in localizing a suspect can make or break the evidence. We want to minimize both the errors.

4.3.2 Evaluating the Proposed Heuristics

Now, we evaluate the impact of the causes on the localization errors. Table 4.3a shows the results for the baseline localization algorithm. The errors are high for the low cardinalities. We see that the errors for 2.4 GHz are more significant compared to that of 5 GHz. This is despite the fact that more APs hear clients and the frequency of transmission of scanning frames is more for 2.4 GHz. This means that the variation in RSSI, including that induced by the transmission power control, has a significant impact on the cardinality and therefore on the localization errors.

Table 4.3b shows how the localization errors vary for the three heuristics for 85^{th} percentile values. There is clearly an improvement for both Same Floor and Different Floor errors. Floor detection with Maximum Number of APs gives the least improvement. In fact, until cardinality 4 it performs worse than the Baseline. A cause behind this is that the distant APs, especially in 2.4 GHz that has longer transmission distance, respond and thus localization errors increase. Next, is the floor detection with the AP with Maximum RSSI and AP of Association. The AP with Maximum RSSI or the AP of Association are typically closest to the client, except when the controller does load balancing and transmit power control. There is a marginal improvement for 5 GHz. Since the Table 4.3 only showed data for 85^{th} percentile, we plot the CDF of error for cardinality=1 in Figure 4.10 for 2.4 GHz. We see that the error reduces for all the percentiles. We see similar results for other cardinalities.

Band	2.4 **GHz**		5 **GHz**	
C	**SF**	**DF**	**SF**	**DF**
1	>50	20.00	29.50	03.10
2	>50	18.30	26.80	02.00
3	>50	33.15	20.12	00.00
4	>50	18.49	NA	NA
5	19.20	10.25	NA	NA
6	23.00	04.17	NA	NA

(a) Baseline

Heuristic	**Maximum Number of APs**				**AP with Maximum RSSI**				**AP of Association**			
Band	2.4 **GHz**		5 **GHz**		2.4 **GHz**		5 **GHz**		2.4 **GHz**		5 **GHz**	
C	**SF**	**DF**	**SF**	**DF**	**SF**	**DF**	**SF**	**DF**	**SF**	**DF**	**SF**	**DF**
1	>50	48.00	29.60	03.49	49.20	14.70	28.30	02.09	32.30	04.90	27.60	01.28
2	>50	33.33	27.00	05.60	24.20	05.60	27.00	03.46	21.60	00.70	25.80	01.48
3	>50	40.00	24.00	00.00	36.00	14.70	20.12	00.00	22.80	05.50	20.12	00.00
4	33.00	13.40	NA	NA	24.00	11.00	NA	NA	18.90	04.40	NA	NA
5	26.00	14.60	NA	NA	17.49	00.00	NA	NA	17.49	00.00	NA	NA
6	22.00	00.00	NA	NA	16.00	00.00	NA	NA	22.00	00.00	NA	NA

(b) Three floor detection heuristics

Table 4.3: Localization errors. The errors are measured as Same Floor (SF) errors in meters and Different Floor (DF) errors in % of floors for Cardinalities (C) ranging from 1 to 6. SF errors are reported as $85^{th}\%ile$. Cardinalities >3 are Not Applicable (NA) to 5 GHz due to cardinality mismatch. Lowest localization errors obtained using the heuristic AP of Association. On an average across all cardinalities, AP of association reduces same floor errors and different floor errors by 58% and 78% in 2.4 GHz while, 3.8% and 46% in 5 GHz, respectively.

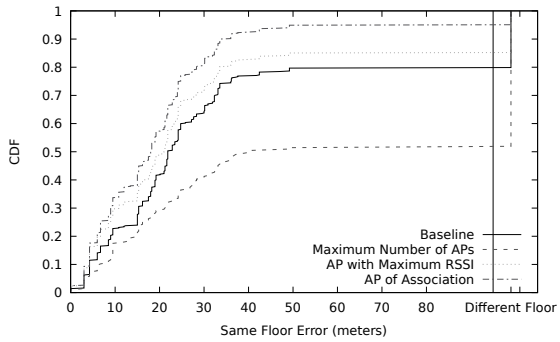


Figure 4.10: Localization errors with three floor detection heuristics – (a) Maximum Number of APs, (b) AP with Maximum RSSI, and (c) AP of Association at $Cardinality$=1 for 2.4 GHz. AP of Association performs the best for both bands. Maximum Number of APs performs worse than the Baseline, while the other two significantly reduced the errors.

We compare our results with Signal SLAM [46] which is deployed in a public space like mall since we also have similar deployments. We have similar observations in other venues too. We find their 90^{th} percentile is about 15 meters. We perform similar. In fact, their AP visibility algorithm has 90^{th} percentile as

24.3 meters. We perform better than this in 5 GHz. Given the complexity of the algorithm Signal SLAM incorporates and the amount of sensing it needs, we believe even with few meters of accuracy, our approach is better; particularly because it is simple and scalable.

In the next chapter, we explore active scans for IoT data transmission.

Chapter 5

Harnessing Active Scans for Data Transfer in WiFi-based IoT Nodes

5.1 Introduction

Previous chapters helped us to conclude that active scans are perpetual in WiFi networks. They can be reduced to a certain extent, but can never be completely eliminated as they are essential to both fundamental connection establishment procedures as well as sophisticated modern services like localization. In this chapter, we explore the case of WiFi networks, where continuing with a WiFi association is hard; for example, in challenged network scenarios. In such scenarios, data transfer is transferred either intermittently or even worse cannot be transferred at all. We propose a protocol named - *ViFi* to enable data transfer in such situations. ViFi protocol leverages active scans and is an apt solution for IoT nodes that are sensors transmitting a few bytes of data intermittently. ViFi does not introduce any new `802.11` frames, needs a minimal change at the AP,

and a simple userspace application at the client-end. We motivate the need for ViFi protocol, its operation, and implementation in Section 5.2. We present the evaluation results in Section 5.3.

5.2 ViFi Protocol

5.2.1 Need of ViFi Protocol

In this section, we motivate the need for the ViFi protocol from the perspective of a WiFi client. The WiFi client tries to transmit its data in a WiFi network, in which establishing and maintaining an association is difficult.

5.2.1.1 When WiFi Fails

A primer on default client states was given in Chapter 2. Out of the states shown in Figure 2.2, *scan*, *associated*, and *transmit data* are known to consume maximum energy [98]. Figure 5.1 shows the process of monitoring the quality of the connection. Revisiting the process of connection maintenance, the client transitions to *scan* state when the quality of connection deteriorates.

Data transmission with the de facto WiFi protocol is hard when either an AP is unavailable – physically due to distance, due to heavy network load, or when the network conditions are such that the clients suffer due to frequent disassociations [53, 54]. We demonstrate these cases in Figure 5.2. As shown in the figure, clients `N1` and `N3` can discover the AP at good and bad RSSI,

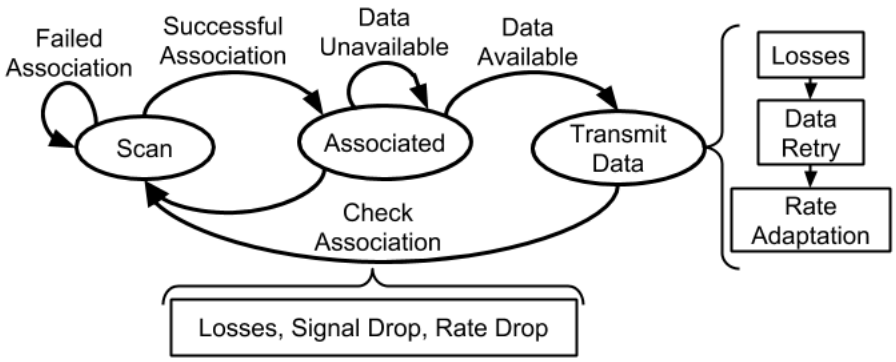


Figure 5.1: Effect of poor network connectivity.

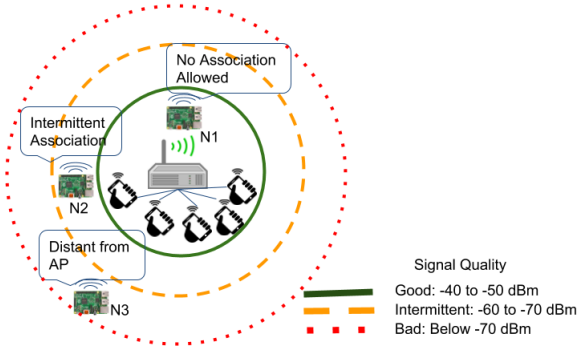


Figure 5.2: Demonstrating the cases where WiFi protocol does not work.

respectively. However, neither of them can associate with the AP. For N1, the AP itself denies the association either due to load or admin policies; while for N3 the RSSI is not enough to establish the WiFi association. Lastly, the client N2 is associated with intermittent connectivity; primarily because the AP is overloaded and the RSSI is midway between good and bad.

We consolidate the cases when WiFi clients are unable to transmit data, the causes, and effects. Table 5.1 summarizes these cases. As shown in the table, either a WiFi association cannot be established [Cases#1 and 3], or if established, it is hard to continue [Cases#2 and 4].

With the default WiFi protocol, its impossible to transmit data in cases 1 and 3; as the premise of this protocol is to have an active association. Reason being

Case	Signal Quality	Associated	Cause	Effect
1	Bad	✗	Sparse AP Deployment	No Association $\Longrightarrow$ No Data Transfer
2	Bad	✓	Reduced RSSI	Connection Maintenance/Handovers $\Longrightarrow$ Intermittent Data Transfer, Low data rate, Energy wastage
3	Good	✗	Overloaded APs or Admin Policies	No Association $\Longrightarrow$ No Data Transfer
4	Good	✓	High Channel Utilization and Losses	Connection Maintenance/Handovers $\Longrightarrow$ Intermittent Data Transfer, Low data rate, Energy wastage

Table 5.1: Need of ViFi Protocol for Data Transfer

– (a) most WiFi network controllers execute algorithms for load balancing that set a threshold on the number of clients that can be associated with an AP. For example, Cisco controllers set an upper limit of 30 clients per AP and (b) the type of clients allowed to use services of enterprise WiFi networks are decided as part of administrative policies. For example, a university may only allow laptops, tablets, or smartphones, but not IoT devices. Under this scenario, the IoT devices will not be allowed association.

However, for the other two cases – cases 2 and 4, data is transferred intermittently. In both these cases, the quality of the connection is poor *i.e.* either the RSSI from the AP is, or the channel is heavily utilized (90% or higher). With the poor connection quality, a client experiences heavy losses, which trigger the link-rate adaptations to reduce bit-rate and handovers, thereby reducing the throughput. This introduces unwanted delays and increases the energy consumption of the client. For an 802.11g network, in poor network conditions, the bit-rate reduces from the maximum 54 Mbps to as low as 1 Mbps. With

a drop in the link quality, the client actively starts looking for a better AP for handovers, thereby increasing the latency. This entire process needs a constant active scanning along with several retry attempts. The energy budget for IoT nodes increases specifically for these two cases.

A fact of vital importance here is that in all the above cases, the client continues to scan – whether or not it is associated. Therefore, we propose to deal with all these cases with our ViFi protocol. Here, we suggest to merge – *wake-up*, *scan* and *transmit data* states. We conceptualize this in Figure 5.3. Primarily, ViFi achieves the data transfer with the probe requests transmitted during an active scan. Probe requests ensure data delivery with a high probability due to the following reasons. The probe requests are always transmitted at the lowest bit-rate – 1 Mbps in 2.4 GHz and 6 Mbps in 5 GHz. Transmission at the lowest bit-rate implicitly means that each bit is transmitted with maximum power.

Therefore, not only these frames have the highest transmission range but the probability of bit-error-rate is also minimum. Moreover, since the probe requests are broadcast in nature, they reach all the APs in the vicinity. Even if one AP receives it successfully, that is sufficient. Lastly, probe requests can be

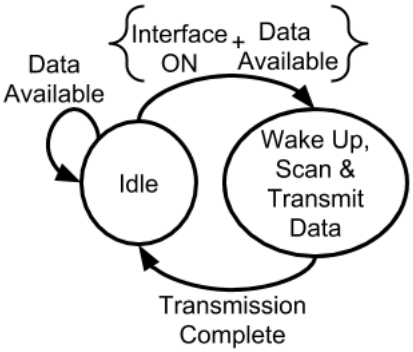


Figure 5.3: Proposed states for an IoT node.

decoded at the AP even at a weak RSSI – with or without an active WiFi association. In the other case, transmissions at such a low RSSI are hard to realize with data frames transmitted with the de facto WiFi protocol. The reasons are – (a) they cannot be transmitted without an active WiFi association and (b) the frame losses increase.

5.2.1.2 Challenges to Implement ViFi

To support such communication, we need to address the following challenges:

(a) Our solution should be agnostic to the software and hardware of the IoT node, i.e., it should work for any processor, driver, and application. It should adhere to the specifications of the standard `IEEE 802.11` protocol.

(b) The solution should not change the WiFi driver to ease the large-scale deployment.

(c) Data should be sent reliably.

(d) The solution should not hamper the WiFi communication, which uses association. For example, it should not hamper the performance of co-existing laptops and smartphones.

We address the challenges as follows:

(a) We implement our protocol as a simple application that can be executed on any device that supports `libnl` [99]. We provide a simple user interface

commands to operate the application. We do not introduce any new frames and do not change the default active scanning procedure. We piggyback data in the probe requests and the probe responses.

(b) Our userspace application does not require any changes to the WiFi driver. Therefore, it is easy to deploy at a large-scale.

(c) Reliability is implicitly achieved as – (i) the probe requests are sent at the lowest bit-rate, and therefore error rate is minimized, and (ii) transmission range of the probe requests is higher, and they can be decoded at lower RSSI, therefore there is a high probability of data reaching an AP. We introduce the sequence numbers for the data and use implicit acknowledgments from the probe responses.

(d) Our protocol ensures that virtue of the following properties does not cross the threshold – (i) the clients carefully choose the frequency of sending probe frames and (ii) the clients do not have any active WiFi connection. Therefore our nodes do not transmit anything but infrequent probe frames.

5.2.2 Operation of ViFi

In this section, we first present our ViFi protocol. Next, we discuss the frame format of a management frame and how we stuff data into such frames. Finally, we discuss the implementation details.

Figure 5.4 demonstrates the operation of the protocol. When data is available at a client, it prepares a probe request frame by stuffing the data in the frame.

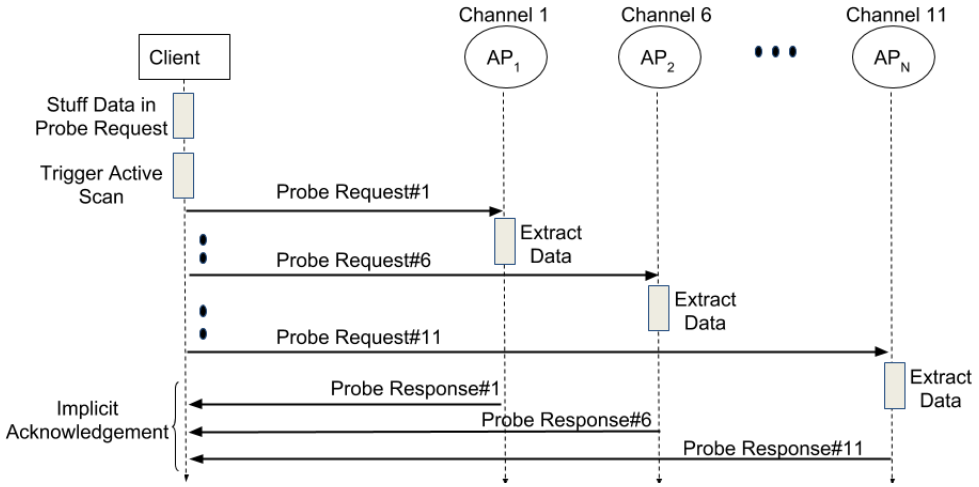


Figure 5.4: Operation of the ViFi protocol.

After that, it triggers a full-band active scan. As per the `IEEE 802.11` standard, when a full-band active scan is triggered a probe request is transmitted on all the channels. In the figure, we show only 3 channels due to space constraints. After that, the client starts a timer and waits for the acknowledgment from the AP(s) in the vicinity. Note that this is not the MAC level acknowledgment, but an implicit one with the probe response from AP.

Next, all the APs in the range receive the probe request on their respective channel of operation. To exemplify, we show 3 APs in the figure operating on channel#1, 6, and 11 and receiving probe requests on these channels. On receipt of the probe request, the AP extracts the data and prepares a probe response for the client. AP then transmits the probe response *i.e.*, the implicit acknowledgment to the client. We do not change the default mechanism, i.e., a default probe response is generated when the AP receives a probe request without our piggybacked data.

5.2.2.1 Frame Format

Figure 5.5 shows structure of a management frame. For each type of management frame, the fields in a Frame Body (2320 Bytes) change. For example, there are 20 fields defined for probe request and 71 fields for probe response. Vendor-specific elements are one of these fields. It is shown as a sub-field of Elements in the Frame Body. Its value is at the liberty of the device vendor. This field is used to achieve interoperability and for non-standard purposes [100]. We make use of this field to stuff our data.

Figure 5.6 shows the proposed structure of data in an element of probe request. We define 4 fields – Sequence Number, More Data, Length, and Data. The sequence number is the byte-wise sequence of data sent. More data field allows easy management of data fragments. If the data to be transmitted cannot be sent in one element, it can be split over multiple elements and potentially multiple probe requests. More Data field is 1 until there is no more data to be transmitted from the client. Length is the length of Data being transmitted. Finally, Data is the actual data being transmitted. Source MAC address is extracted from the MAC header of probe request. For a probe response, only one

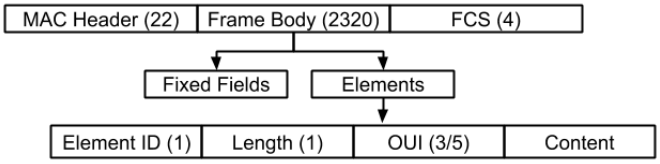


Figure 5.5: IEEE 802.11 management frame format

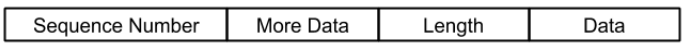


Figure 5.6: Proposed frame format for stuffing elements in the ViFi protocol.

element is added. It contains the last received sequence number.

5.2.2.2 Protocol Implementation

Now, we discuss the implementation details of our protocol for both the client and the AP. Figure 5.7 shows the schematic view of the implementation.

(1) Client-side implementation: At the client-end, an application runs at the client in the user space of the operating system. The client triggers a scan with a utility command `iw` [101]. `iw` uses a library `libnl` to pass the scan request from the application layer to the kernel. We make use of `libnl` library to stuff the data received from the application layer in the vendor-specific elements of probe requests.

When the application generates the data, we get the interface name on which the data is to be transmitted with `if_nametoindex()`. Then, a netlink socket is created and bound with `nl_socket_alloc()` and `genl_connect()`, respectively. `nl80211` driver is then connected to the socket using `genl_ctrl_resolve()`. Once the setup of `nl80211` socket is complete, we create elements using the method `create_vendor_ie()`. `do_probe_stuffing()` stuffs the data in these elements and triggers a scan. Lastly, the status of transmission – success/failure – is tracked with `track_status()`.

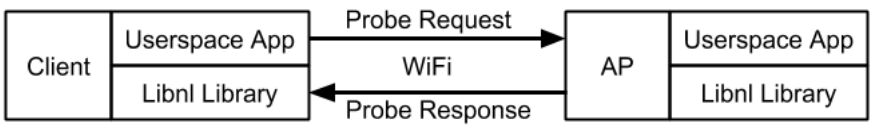


Figure 5.7: Implementation details of the ViFi protocol

These probe requests are transmitted by the hardware driver when the `mac80211` driver schedules the scan. For an end-user, only compiled version of the `libnl` library and our application layer code is needed to execute our protocol.

Table 5.2 shows an abstract representation of new/modified methods along with their source files. We contribute 558 lines of code for the client-side implementation. The executable binary of our client-side code with the statically linked `libnl` library has a size of 29 KB. Thus, it is easy to port on memory constrained devices. We ported our client code to a Raspberry Pi module (Model 3B) with an internal BCM43438 WiFi chipset as well as an external TP-Link WN721N USB WiFi adapter. We have also ported it to a Linux PC with Atheros AR9271 chipset for the WiFi.

(2) AP-side implementation: We use `hostapd` [102] for implementing the AP-side protocol. We modify the default `hostapd` code and run a userspace application on the AP. In the `hostapd`, on receipt of a probe request `handle_probe_request()` method is called. In this method, we check for the pres-

	Method Name	**Explanation**
Client-side	`create_vendor_ie()` [`utils.c`]	Creates customized vendor elements.
	`do_probe_stuffing()` [`probe_stuffing.c`]	Stuffs data generated at the client in the elements created by `create_vendor_ie()`. Then, it triggers the scan to send stuffed probe requests.
	`track_status()` [`main.c`]	Tracks the status of successful and failed probe requests.
AP-side	`handle_probe_request()` [`beacon.c`]	Filter the probe requests received with our customized vendor elements. Extract the sequence number. Prepares probe response to acknowledge the sequence number received.

Table 5.2: Code Abstractions

ence of our customized elements. If they are present, then the AP extracts the data. Next, the AP stuffs the sequence number of successfully received probe request in the probe response and transmits it to the client. It is easy to port the implementation to an OpenWRT AP that also uses `hostapd`.

The userspace application running over the AP sends the extracted data to the repository. During a scan, multiple copies of the same data are broadcast in probe requests on different frequencies, possibly resulting in multiple APs receiving copies of the same data. Our application does deduplication of the data. The MAC address of the sender and the Timestamp of the received data forms the basis for deduplication.

Table 5.2 also shows the abstract representation of new/modified methods for the AP. The size of the default `hostapd` code $\approx$ 4 MB. We contribute 112 lines of code (15 KB) for AP-side implementation. Thus, our code has minimal impact on the size of the `hostapd`. We do not need any kernel-level or physical driver level modifications. We plan to make all the source code publicly available.

5.3 ViFi Evaluation

In this section, we first describe our metrics, the experimental setup for evaluating the performance of ViFi protocol. After that, we present the evaluation results.

5.3.1 Metrics to Measure the Performance of ViFi

We use the following metrics to measure the performance:

(a) #Frames: We study the number and the type of MAC layer frames transmitted by the client(s). More the number of frames a client transmits, more it stays in the powered on states – *scan*, *associated*, and *transmit data* state, and more energy it consumes. For the same amount of data to be transmitted, clients with the ViFi protocol transmit lesser frames than with the WiFi protocol.

(b) Airtime Utilization: Airtime Utilization is the percentage of time for which the channel is busy per unit time. We calculate airtime utilization as proposed in [103], with the size and the bit-rate of the frames transmitted in a time slot of 1 second. As the airtime utilization increases, the channel is occupied for more time and thereby affects the performance of other clients. Clients operating with the ViFi protocol consume lesser airtime as compared to when they operate with the WiFi protocol.

(c) Delivery Rate: We report this metric as the percentage of instances when the data generated at the client was successfully recorded at the AP, in a given duration of time. This metric analyzes the reliability of a protocol. Higher the value of delivery rate for a protocol, higher is the reliability of the protocol. The delivery rate of the data transmissions with the ViFi protocol is consistently higher than the WiFi protocol.

(d) $Goodput$: Revisiting the definition of $Goodput$ in Section 2.3.2.3, we

define $Goodput$ as the number of bits successfully transmitted in a unit time. We measure $Goodput$ at the MAC layer with the amount of data that is carried in the frames. This metric helps us analyze the maximum data-transmission capability of a protocol. $Goodput$ of ViFi protocol is better than that of the WiFi protocol when the traffic volume is high.

(e) End-to-end Delay: The time elapsed between the generation of data at the client end and the reception of the same data at an AP. This metric helps in analyzing the delay perceived by the end-application. ViFi protocol experiences a higher end-to-end delay than the WiFi protocol.

(f) Energy Consumption: We measure the energy consumed, in $watt - hour$, by a client for completing a data transfer. This is a key metric for low-end battery-operated devices that help in evaluating their power consumption. Clients operating with the ViFi protocol consume lesser energy than with the WiFi protocol.

(h) Contention Time: In WiFi, the clients follow CSMA/CA where they contend for the channel *i.e.*; they need to listen before they can transmit. If the channel is busy, they back-off. Contention increases with the number of clients in the WiFi network. The performance of a client is affected when it does not find the channel to be free and wastes time in contention/collision [104]. We measure the above-said impact with the amount of contention a client with the WiFi protocol faces.

WiFi is the default protocol for the non-IoT clients which run different types

of applications such as bulk data transfer like file downloads or interactive traffic like VoIP. The performance of such applications is measured with numerous types of application layer metrics such as throughput, jitter, and latency and so on. Instead of using separate metrics for measuring different application layer performance, we come up with a generic metric – *contention time*. If this metric gets affected, all other application layer metrics will get affected.

Contention time measures the inter-frame arrival time *i.e.* the time between two data frames from the same client. Ideally, in the absence of contention, there are no re-transmission attempts. Therefore, we measure the contention time with the inter-frame arrival time for the first transmission attempt of data frames, excluding the retransmitted data frames. A higher value of this metric denotes higher contention on the channel which ultimately affects the frame transmissions from non-IoT clients. Therefore, a lower value of this metric is favorable for the performance of non-IoT clients.

Contention time of the non-IoT clients increases more with the co-existing IoT clients if the IoT clients use WiFi protocol than ViFi.

5.3.2 Experiment Methodology

Prior to explaining the experiments, we explain the network conditions we consider for the experiments.

We consider two network conditions – Bad and Good. In a bad network condition, frame losses are high, and it is hard for a client to maintain an association

with the AP. We emulate these network conditions by increasing the distance of a client from the AP. Higher is the RSSI; better is the quality of the connection. For a good network condition, we ensure that the RSSI from the AP is $\approx$ -40 to -50 dBm. In a bad network condition, the client experiences frame retries and intermittent network connectivity. Therefore, we place the client at a distant location from the AP such that the RSSI is less than -70 dBm.

We perform two experiments to (a) compare WiFi and ViFi, and (b) study the impact of ViFi on WiFi. Both the experiments are explained as follows–

($E.1$) *Comparison of WiFi and ViFi protocol:* To analyze which of the two protocols – WiFi or ViFi is well suited for IoT applications, we compare the performance of a client operating with each of these protocols. Since the challenge is to transmit data with minimal or no WiFi association, we specifically present results for bad network condition in this experiment. Performance of ViFi stays unchanged even in good network conditions. For both the protocols, clients send the same amount of data at the same time interval, T seconds. In this experiment, 5 clients operate with the ViFi protocol and 5 clients operate with the WiFi protocol, together. We want to analyze their performance when the clients of both protocol categories face the same network conditions. Therefore, we mandate the co-existence of clients operating with both protocols in this experiment.

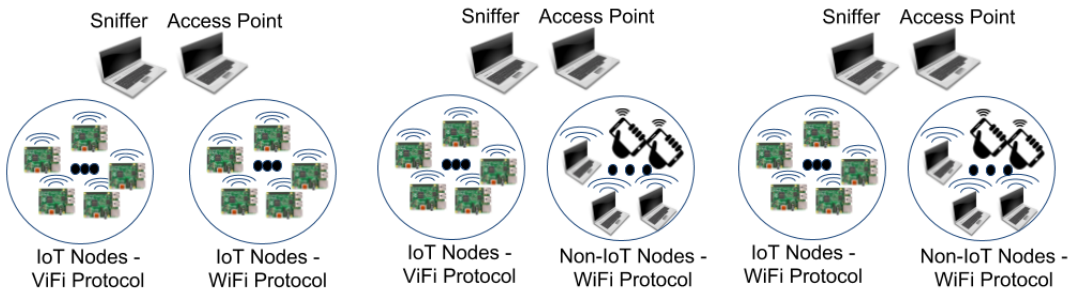
($E.2$) *Impact of ViFi on WiFi:* With the present day heterogeneous WiFi networks IoT nodes often co-exist with non-IoT nodes *i.e.* the WiFi network clients

of an enterprise. The ViFi protocol injects low bit-rate traffic in the network. In heavily utilized WiFi networks, this low bit-rate traffic creates congestion, and thus the performance of all clients in the network suffers. Therefore, it is essential to understand the impact of transmissions with the ViFi protocol on the transmissions with the WiFi protocol. This analysis allows us to choose the better communication protocol – ViFi or WiFi for IoT clients.

5.3.3 Experimental Setup

Figure 5.8 shows the setup of both the experiments. We configure a Ubuntu 14.04 PC with `hostapd` to run as an AP. This setup is similar to an OpenWRT AP. We configure – Raspberry Pi nodes as IoT nodes, laptops and smartphones as non-IoT nodes, and a TP-Link USB WiFi adapter with `ath9k_htc` driver as a sniffer listening on the same channel as AP. As with other experiments, the sniffer is placed close to the AP to maximize the likelihood of recording all frames seen by the AP. All the devices in the setup are synchronized over NTP for calculating one of our metrics *i.e.*, delay. All the devices are instrumented to log timestamps of all the events. While the setup of the AP and the sniffer remain the same for both the experiments, the clients vary. We emulate the data transmission of non-IoT nodes with `iperf` utility.

As shown in the Figure 5.8a, Experiment $E.1$ has 2 categories of IoT nodes – one operating with the ViFi and the other operating with the WiFi protocol. Experiment $E.2$ has IoT and non-IoT nodes. Non-IoT nodes always operate with the WiFi protocol while the IoT nodes in a sub-experiment of $E.2$, shown



(a) Experiment $E.1$ (b) Experiment $E.2$ - Setup 1 (c) Experiment $E.2$ - Setup 2

Figure 5.8: Experiment setup to evaluate ViFi protocol.

in Figure 5.8b, operate with the ViFi protocol and in another sub-experiment, shown in Figure 5.8c, with the WiFi protocol. For both the experiments, the clients generate a sample data of 200 Bytes at regular data generation intervals of T seconds. We consider 2 data generation intervals $T = 1$ second (minimum) and 120 seconds (maximum) so as to understand the performance bounds. We consider these 2 values because – (a) a data generation interval of less than a second is impractical for most IoT nodes and (b) empirically, we see minimal variation in the results for ViFi post 120 seconds of data generation interval. Note that, even though we show results for these 2 intervals, all our experiments were performed for 5 different data generation intervals – 1, 10, 30, 60, and 120 seconds. We show results for a sample duration of 30 minutes of data transfer. Results do not change for higher durations. All the metrics are calculated with the logs collected at the client-end, the AP-end, and the sniffer.

5.3.4 Results

First, we compare the performance of WiFi and ViFi protocol. Next, we test the scalability of our ViFi protocol. After that, we study the impact of both the

protocols on other enterprise non-IoT clients.

5.3.4.1 ($E.1$) *Comparison of WiFi and ViFi Protocol*

(a) #Frames – Figure 5.9 shows the results. For both T, the number of frames with the WiFi protocol are considerably higher than the ViFi protocol. Not only clients with the WiFi protocol transmit a higher (at least 470%) number of frames, most of these frames are retried (32%) in bad network conditions. Along with the data and the probe requests, these clients also transmit *other* frames. The *other* frames are the ones that aid clients in maintaining the WiFi connection such as reassociation related frames.

The results imply that the clients with the WiFi protocol (i) mostly stay in *transmit data* state, (ii) have to attempt a data transmission several times before it gets to the AP, (iii) struggle to maintain the association with the AP. Hence, the clients not only consume higher energy as they are mostly active and transmitting data, but they also occupy channel for a longer time. Also, note that the clients with the WiFi protocol continue to transmit probe requests, as the driver triggers routines for monitoring the quality of association. In fact, at $T = 120$ seconds the number of probe requests from WiFi protocol are 2.25x than the ViFi protocol.

For the ViFi protocol, the only type of frame transmitted is the probe request. Further, there are no retries because as per the `IEEE 802.11` standard, probe requests are never retried. As a result of this, post-transmission the client with

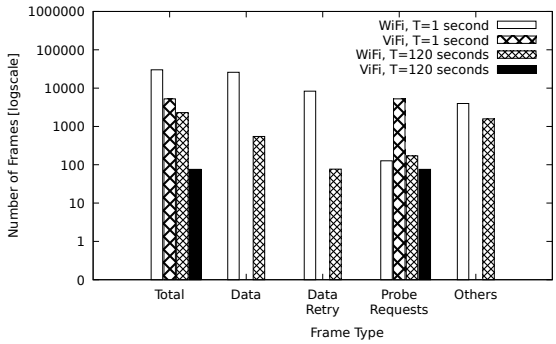


Figure 5.9: Comparing the number and type of frames transmitted with the WiFi and ViFi (Experiment $E.1$). Clients with the WiFi protocol transmit at least 470% more frames, with 32% retries, than the ViFi protocol for the same amount of data to be transmitted. Out of the total 156 KB data generated per client for T=1 second, WiFi could successfully transmit 137 KB while ViFi transmitted 153 KB. For T=120 seconds, total 3 KB data was generated per client. Both protocols were able to transmit the entire data; however, WiFi achieved the transmission with significant retries.

the ViFi protocol transitions to the *sleep* state and saves energy. Given that the amount of actual data being transmitted to the AP is the same for both the protocols, results show that the ViFi protocol is more efficient than the WiFi protocol.

(b) Airtime Utilization – Figure 5.10 shows the results for airtime utilization. We report airtime utilization of all clients for each protocol and data generation interval. For both data generation intervals, data transmissions with the WiFi protocol consumes much higher airtime than the ViFi protocol. The reason of this result is two-fold – (i) client with the WiFi protocol transmits a higher number of frames and (ii) as the quality of association drops, rate adaptation is triggered that results in dropping the bit-rates to as low as 1 Mbps for transmissions. Both of these factors contribute to higher airtime utilization for the WiFi protocol.

Higher airtime utilization signifies that the channel is occupied for a longer duration. That said, as the number of clients operating with the WiFi protocol

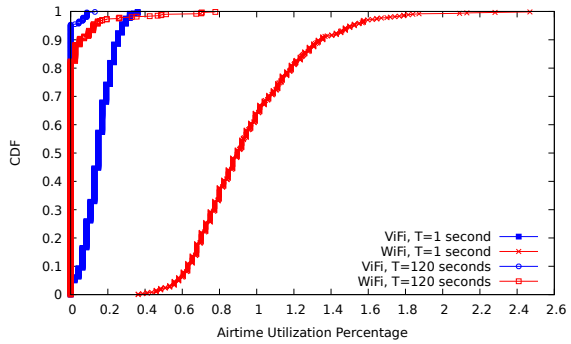


Figure 5.10: Comparing the airtime utilization of WiFi and ViFi (Experiment $E.1$). Data transmission with the WiFi protocol consumes 6x and 8x [90%ile and higher] higher airtime than the ViFi protocol for $T = 1$ and $T = 120$ seconds, respectively.

increase in the bad network conditions, the channel will be mostly busy resulting in backoffs at the clients and lost or delayed acknowledgments. However, that is not the case with the ViFi protocol as the number of frames transmitted are lower and there are no retries. Reduced airtime consumption will allow an increased number of clients to transmit their data.

(c) Delivery Rate – Table 5.3 shows the results for the delivery rate. The ViFi protocol maintains high (at least 98%) delivery rate with a much lesser number of frames transmitted without any frame retries. The delivery rate of the WiFi protocol drops in the bad network condition when maintaining the association is hard. In fact, there will be no delivery when the WiFi association breaks. However, this is not the case with the ViFi protocol. Thus, the ViFi protocol is more reliable than the WiFi protocol.

(d) $Goodput$ – Table 5.4 shows the results for $Goodput$. For $T = 1$ second, ViFi protocol is $\approx 11.25\%$ better than the WiFi protocol. For $T = 120$ seconds, the $Goodput$ stays the same for both the protocols. Note that we initiate the transfer of the same amount of data for both the protocols. However, the

Protocol	Delivery Rate	Frame Retries
Data Generation Interval, 1 second		
WiFi	88.23%	32.19%
ViFi	98.15%	00.00%
Data Generation Interval, 120 seconds		
WiFi	100.00%	14.07%
ViFi	100.00%	00.00%

Table 5.3: Comparing the Delivery Rate of WiFi and ViFi (Experiment $E.1$). Delivery rate of ViFi protocol is $\approx 11\%$ better than the WiFi protocol at data generation interval of 1 second. WiFi protocol always attempts frame retries. Even with 100% delivery rate for the WiFi protocol, frame retries cannot be ruled out; while there are no retries for the ViFi protocol.

	Data Generation Interval	
Protocol	1 **second**	120 **seconds**
WiFi	611 bps	16 bps
ViFi	680 bps	16 bps

Table 5.4: Comparing the *Goodput* of WiFi and ViFi (Experiment $E.1$). ViFi protocol performs better or at par with the WiFi protocol with no frame retries.

WiFi protocol struggles to achieve this *Goodput* not only with significant frame retransmissions but also with several MAC frames transmitted for connection maintenance. Nonetheless, the WiFi protocol will have higher *Goodput* with a lesser data generation interval, frame aggregation and higher standards like `802.11n` and `ac`.

(e) End-to-End Delay – ViFi protocol experiences higher end-to-end delay than the WiFi protocol. We notice an increase of 4.9x and 2.5x [median values] for T=1 second and 120 seconds, respectively. The reason behind the higher delay for the ViFi protocol is the lowest bit rate at which the probe requests are transmitted. This delay reduces with an increase in the data generation interval. Hence, the growth is lesser for T=120 seconds than 1 second.

(f) Energy consumption – We use an off-the-shelf USB power monitor to measure the energy consumption. It reports the voltage, the current, and the

power drawn by a device. It is connected in series, between the power source output and input of a Raspberry Pi. We collected per-second readings for the power consumption of the Raspberry Pi module. Duration of data collection was one hour, which is sufficiently long to get an estimate of energy consumption.

For the WiFi protocol, the module consumed 1.82 kWh energy in good network condition and 1.84 kWh in bad network condition. For the ViFi protocol, the module consumed 1.50 kWh energy in both network conditions. Thus, we can save 18.4% of the energy with the ViFi protocol.

Next, we analyze the performance of the ViFi protocol as the number of IoT nodes increase in the network *i.e.*, as the scale of network increase. For this purpose, first, we experimentally study the behavior of the protocol with incrementally adding up to 10 IoT nodes. Before every incremental addition, data is transmitted for 10 minutes. We do this exercise for the same 5 values of T defined in section 5.3.3.

With 10 clients transmitting at the lowest $T = 1$, the number of frames transmitted are 4877.These frames are at least 280% lesser than the WiFi protocol. We observe that the airtime utilization even for 10 clients stays less than 1% at all T. The minimum delivery rate is 92% when the clients transmit at $T = 1$ second. However, it mostly stays at 100% (with a minimum of 95%) for higher values of T. Next, the ViFi protocol maintains a $Goodput$ of 680 bps for $T = 1$ second and 16 bps for $T = 120$ seconds. Lastly, the maximum end-to-end

delay is observed as 4 seconds at $T = 1$ second, which reduces to mere 221 milliseconds at $T = 120$ seconds. These results stay approximately the same irrespective of the network condition.

5.3.4.2 ($E.2$) *Impact of ViFi on WiFi*

Next, we analyze the impact of transmissions from the IoT clients on the transmissions of non-IoT clients. In experiment $E.2$, we want to understand if the transmissions from IoT clients hamper the transmissions from non-IoT clients. To understand the impact, we set the non-IoT clients to send traffic at maximum possible rate *i.e.*, in saturation condition. However, the IoT clients continue to send data at $T = 1$ second and $T = 120$ seconds. Note that unlike the previous experiment $E.1$, the channel is now saturated with the traffic from non-IoT clients. Table 5.5 shows the contention time for the non-IoT clients.

As we see from the results, transmission of IoT data with the ViFi protocol minimally affects the transmissions from non-IoT clients. This is because the IoT clients transmit a lesser number of frames with the ViFi protocol and also consume lesser airtime. As we discussed in Section. 5.3.1, the non-IoT nodes can run different types of applications such as bulk transfer, interactive

	Data Generation Interval	
Protocol	1 **second**	120 **seconds**
WiFi	5.23 seconds	120 milliseconds
ViFi	0.08 seconds	100 milliseconds

Table 5.5: Comparing the contention time introduced with the WiFi and ViFi (Experiment $E.2$). Lower this metric, better is the performance of non-IoT clients. Non-IoT clients perform 98.5% better at $T = 1$ second and 21.2% better at $T = 120$ seconds when the co-existing IoT clients operate with the ViFi protocol than the WiFi protocol.

traffic. Now, the contention time of a few seconds might not severely affect the application-layer throughput of a bulk transfer but, it will severely affect jitter and latency for interactive traffic. Hence, the co-existence of non-IoT clients with the default WiFi protocol and IoT clients with the proposed ViFi protocol is a favorable design choice.

We discuss the related work of all three problems in the next chapter.

Chapter 6

Related Work

6.1 Active Scanning

6.1.1 Procedure and Extent of Active Scanning

The procedure of active scanning is well-studied in literature. Authors in [105, 21, 106] empirically explain the sequence of events that happen during active scanning. These papers present the intrinsics of active scanning from the perspective of a client. Authors in [105] empirically study the exchange of probe requests and probe responses and analyze their impact on the handover process. Hu et al. in [107] present the dynamics of probe requests in large-scale and dense WiFi networks. Their study covers venues like stadiums and classrooms. They quantify the details of probe requests across frequency bands, scan interval, scan duration, and the number of SSIDs.

6.1.2 Impact of Active Scanning

The impact of active scanning on the performance of a client concerning increased delay during the handover process is studied in detail [21, 105, 23, 108, 109, 110, 111, 112, 113]. Its impact on the energy expenditure of a client is examined by Hu et al. in [107] and Raghavendra et al. in [54]. They recognize the negative impact of probe requests on the performance of a dense WiFi network. They describe the issues with current procedures of active scanning and how they are detrimental to the operation of a WiFi network. The analysis presented in these studies is either limited to client-side, or it partially covers the aspects of probe traffic that affect performance on a network-wide scale. We extended this analysis by covering various aspects of probe traffic – the amount, the inter-frame arrival times, frame sizes, data rates, airtime utilization, and redundant probe traffic.

While existing studies aim to reduce scanning delays, the impact of active scanning on communication delays between an AP and a client in dense WiFi networks remain unknown. Furthermore, the extent of airtime consumed by probe traffic in dense WiFi networks is unknown. The fact that most of this probe traffic is avoidable makes it even more necessary to understand the impact of active scanning on the performance of a network.

6.1.3 Reducing Active Scans

The state-of-the-art has solutions to reduce the delays of active scanning from the perspective of a single WiFi client. With the proposed solutions a client discovers APs with reduced scanning delays.

The solution proposed in [23] suggests synchronized scanning among APs and clients. DeuceScan [108] exploits spatiotemporal information available at a client to find the best AP to roam to with an aim to reduce handoff delay. FastScan and Enhanced FastScan [109], suggest the use of a client-side database of APs to enable faster scan. Authors in [110], suggest an interleaving technique to use an optimal combination of active and passive scanning algorithms. Multiple radios [111] along with recently developed MPTCP (Multipath Transmission Control Protocol) [113] allows a client to stay associated with multiple WiFi networks via virtual interfaces. Thus, reducing the handoff delays. CSpy [112] authored by Souvik Sen et al. reduces the scanning delays by reducing the number of probe requests transmitted by a client. It finds the strongest channel by sending a probe request on only one channel and using this to predict the quality of nearby channels. To do this, CSpy uses the Channel Frequency Response, as given by the chipsets, together with machine learning techniques.

6.1.4 Limitations of Existing Solutions

All the above solutions solve the issue of scanning delays at the client. Most solutions need some change at the client-end either at the hardware or in the driver. Nonetheless, these solutions do not detect the causes of active scanning in a WiFi network. Even though the effects of active scanning are studied in the literature, to the best of our knowledge, we could not find a mechanism that can automatically identify the cause of active scanning.

6.2 Data Transfer in WiFi-based IoT Nodes

6.2.1 Communication Protocols for IoT

A variety of protocols are available for enabling the communication for IoT nodes. Most common examples are 6LoWPAN, Bluetooth, Bluetooth-Low-Energy (BLE), WiFi, Zigbee, Z-wave [114]. Commonalities between these communication technologies are, (a) a comprehensive deployment setup with gateways and servers is needed, (b) the nodes communicate only after establishing a connection. For example, BLE operates with a master-slave operation. Slaves send advertisements and masters discover slaves to establish a connection; after which data is transmitted, and (c) the coverage area is smaller [115, 116, 117]. The basic mode of operation of the technologies mentioned above such as BLE, Zigbee is different from WiFi. Therefore, they cannot co-exist with existing infrastructure-based WiFi network deployment. Given the ubiq-

uity of the WiFi, it is much simpler technically for IoT nodes to co-exist with existing WiFi network deployments; which is why we consider the WiFi as a communication technology.

6.2.2 Efficient MAC for IoT

The recent IEEE 802.11ah [118, 119] and IEEE 802.11ax [120] standards has been launched for WiFi-based IoT nodes. IEEE 802.11ah enables communication in sub-1 GHz band. To limit contention and to save energy of the nodes, IEEE 802.11ah proposes to divide the stations into various groups. The contention is limited to a group only. There are different time slots for groups to communicate. The standard IEEE 802.11ax operates in 2.4 GHz and 5 GHz. It proposes *Target Wake Time* for the nodes, where the target for a node is defined as the sleep and wake-up periods at pre-scheduled times. When a potential trigger comes, the IoT nodes wake-up and transmit. The nodes will have protocol exchanges where they will negotiate their sleep cycles. While these new standards are complementary to our solution and will enhance the gain of our ViFi protocol, they still need an association to take place before any communication to happen.

There are several works that aim to make data transfer efficient with WiFi, particularly for IoT nodes. For example, authors in [121] propose a unified MAC protocol with contention-based CSMA and contention-free TDMA. They demonstrate that their unified MAC is more efficient than the Zigbee protocol. Next, authors in [122] propose an efficient cross-layer MAC design that mini-

mizes the delay and energy consumption. They come up with a multi-objective joint optimization across three layers– the physical layer, the link layer, and the network layer. They claim that the proposed cross-layer MAC takes care of different communication and application needs. Authors in [123] propose a MAC protocol based on CSMA which operates in the visible light spectrum. Authors in [124] employ an interference averaging technique with which they enable simultaneous resource sharing for the users. In their MAC technique, they propose to choose transmission duration and transmit power levels judiciously. Authors demonstrate that the proposed MAC is efficient when compared to default CSMA. Authors in [125] use continuous bit-rates instead of discrete bit-rates to improve the throughput.

All these protocols presume that the client is associated with an AP, unlike ours where we aim to enable the communication in intermittent or even worse, no association at all. Therefore, even though these protocols improve the efficiency of data transfer in IoT, they fail to apply to the scenario that we target.

6.2.3 Applications without WiFi Association

Several innovative usecases have been proposed for association-free WiFi. For example, distributing business coupons [65], communication for IoT nodes [126], location-based announcements [127, 128], concurrent WiFi [129], road safety [130], disaster recovery [69], traffic management [67], interference mitigation [66], radio resource management [131], localization [68], and personal analytics [132].

6.2.4 Limitations of Existing Solutions

Overall, we find that all the previous research focus on end application and not on the communication aspect of transmitting data in the absence of a WiFi association. For example, the application in [127, 128] is *Location based WiFi Services system (LoWS)*. Their application broadcasts emergency information stuffed in AP beacons that are flashed on a client's screen. Similarly, in [66, 131] the end application is interference mitigation. Authors in [65] propose multiple applications such as selecting a network, location-dependent advertisements, and distributing business coupons.

The main focus of all these works is on optimizing the performance of the proposed end application and not *how* an association-free protocol would work in its entirety. For example, specifically when data is transferred with management frames – will there be any impact of this approach when their clients co-exist with other clients that send data frames? how will the behavior of their protocol change when the clients are in a dense and overloaded WiFi network?

In contrast to the existing work, our aim is not only to ensure a mechanism for transferring data from an IoT node to an AP, without needing an active WiFi association but, also to understand in detail the impact of such a protocol on clients running default WiFi protocol. It is essential to study this because WiFi networks, by virtue, are dynamic. The performance of transmissions with data frames in WiFi networks is well understood with various stress tests conducted but with probe frames (for association-free communication) is unknown. We

provide detailed anatomy about the operation of the ViFi protocol from the perspective of the MAC layer.

6.3 Indoor Localization

6.3.1 Fingerprint vs. Model-based Solutions

One of the oldest localization techniques use either a fingerprint-based [133, 28, 134, 135, 136, 137, 138, 139] or model-based [140, 141, 142, 143] approach, or a combination of both [144]. Overall, fingerprint-based solutions tend to have much higher accuracies than other approaches albeit with a high setup and maintenance cost [48]. The fingerprint-based approach was pioneered by Radar [28] and has spurred numerous follow-on research. For example, Horus [136] uses a probabilistic technique to construct statistical radio maps, which can infer locations with centimeter-level accuracy. PinLoc [135] incorporates physical layer information into location fingerprints. Liu *et al.* [139] improved accuracy by adopting a peer-assisted approach based on p2p acoustic ranging estimates. Another thread of research in this line is to reduce the fingerprinting effort, for example, using crowdsourcing [138, 145, 137, 146], and down-sampling [133]. The mathematical signal propagation model approach [142, 140] has the benefit of easy deployment (no need for fingerprints) although its accuracy suffers when the environment layout or crowd dynamics change [143]. Systems, such as EZ, improve the accuracy by additionally using GPS to guide the model construction.

6.3.2 Client vs. Infrastructure-based Solutions

There is a rich history of client-based indoor location solutions, to name a few, SignalSLAM [46], SurroundSense [134], UnLoc [147], and many others [29, 145, 137, 148, 30, 31]. All of them share some commonalities in that they extract sensor signals (of various types) from client devices to localize. The location algorithms usually run on the device itself; however, it is also possible to run the algorithm on a server and use the signals from multiple clients to achieve better performance [139]. Overall, client-based solutions have very high accuracy (centimeter resolution in some cases [136, 135]). An alternative would be to pull signal measurements directly from the WiFi infrastructure, similar to what our solution does. The research community has only lightly explored this approach since it requires full access to the WiFi network controllers, which is usually proprietary. Our main competitors are the commercial WiFi providers themselves. In particular, both Cisco [149] and Aruba [150] offer location services. These solutions use server-side tracking coupled with model-based approaches (to eliminate fingerprint setup overhead).

6.3.3 Other Solutions

There are several other solutions, complementary to the signal strength-based technique. Time-based solutions [34, 35, 36, 37, 38] use the arrival time of signals to estimate the distance between client and AP, while angle-based solutions [45, 151, 152, 32, 33] utilize angle of arrival information, estimated

from an antenna array, to locate mobile users. Recently, the notion of passive location tracking [153, 154, 39, 40, 41] has been proposed, which does not assume people carry devices. In large and crowded venues, however, the feasibility and accuracy of such passive tracking is still an open question. Other systems like light-based localization [155, 156, 157] and acoustic-based localization [158, 159, 160, 161].

6.3.4 Limitations of Existing Solutions

These solutions can achieve higher accuracy, but they have at least one of the following limitations – (a) need of a customized hardware, which cannot be implemented in large-scale deployments, (b) installation of client application, making them hard to scale, (c) rooting client OS - Android or iOS, which limits their generalizability, (d) energy savvy, (e) high error rates in dense networks, and (f) proprietary and expensive to deploy (especially, solutions from vendors like Cisco and Aruba).

To summarize, even though several wonderful solutions are available, their scalability is still a question. Therefore, we advocate using server-side localization approach with fingerprints. Our aim is not to compare the efficacy of different approaches, but to address the challenges of practical and widely deployed device-agnostic indoor localization using today's WiFi standards and hardware, for example, use of 5 GHz band and controller-based architecture.

We conclude in the next chapter.

Chapter 7

Conclusion

To conclude, this thesis addresses three significant problems that emanate from active scanning in present-day WiFi networks. All the solutions proposed in this thesis are device agnostic; thereby making them easily integrable with controller-managed WiFi networks and simpler to scale with a multitude of the WiFi devices.

7.1 Key Contributions

Key contributions of this thesis are as follows:

7.1.1 Understanding and Mitigating the Impact of Unnecessary Active Scans in WiFi Networks

With the help of real-world datasets, we provide a detailed analysis of *why* and *how* does active scanning affect the performance of a WiFi network. We pursue these questions separately from the perspective of an individual client as well as

the whole network with a quantitative comparison for both frequency bands. We study many existing WiFi driver implementations and user space applications to arrive at three causes of active scanning – *Discovery*, *Connection Establishment*, and *Connection Maintenance*. We present an end-to-end solution that consists of a *metric* to diagnose the growth and the need of probe traffic, *inference mechanism* to understand the cause of the growth of probe traffic, and *an algorithm* to reduce unnecessary probe traffic. All the analysis presented is first of its kind to understand and mitigate the impact of unnecessary active scans, at scale.

The metric Probe to Fresh Data Frames Ratio (P/D) measures the increase in probe traffic in realtime and deduce if it is affecting network goodput. We showed that the metric works by testing it in both controlled and uncontrolled network setups. The proposed inference mechanism, can run at the WiFi network controller and help a network administrator diagnose the problems of which excessive probe traffic is symptomatic. We discuss the implications of inferred causes for a network administrator. The proposed client-end solution to reduce the number of probe requests can be installed on any Linux/Android WiFi client with a simple application update.

Few conclusions that can be drawn with the inference mechanism are as follows –

A large percentage of probe traffic because of the causes *Low RSSI*, *Data Frame Losses*, *Beacon Losses*, and *AP-side Procedures* may indicate a need for better network planning. The first three in the list indicate problems with

coverage or excessive interference and may be rectified by introducing more AP(s) and ensuring better channel allocation to mitigate interference due to a high density of AP(s) and clients.

AP-side Procedures reflect triggering of algorithms at the controller that aims to prevent degradation of network performance. For example, the procedure may result in an AP switching to a different channel as part of interference management or an AP disconnecting a client to manage its load. A significant frequency of occurrence of the cause *AP-side Procedures* may indicate an inability of the controller to perform load balancing. This may be because of a faulty controller or because of a poorly planned network.

The significant occurrence of the cause *Periodic Scan* by unassociated clients indicates the presence of rogue clients in the network. Enterprise WiFi networks often have security mechanisms, such as MAC address registration, failing which a client cannot associate with the network. This can lead to excessive probe traffic from unregistered clients who are unable to associate to an AP and degrade network throughput. A high frequency of *Periodic Scan* by associated clients may hint at aggressive client behavior.

7.1.2 WiFi Indoor Localization using Existing Infrastructure in the presence of Minimal Active Scans

We present real-world evidence of "where" and "what" may go wrong for practically localizing clients in a device agnostic manner. We identify and describe two novel and fundamental problems associated with a server-side localization

framework – (a) "Cardinality Mismatch" and (b) "High Client Scan Latency" problems. We discuss the reasons why these problems are non-trivial to be solved given the challenge of no client and/or infrastructure-side allowed. We propose heuristics to improve the accuracy of the localization in the face of these problems. We describe our experiences with deploying, managing, and improving a fingerprint-based WiFi localization system, which has been operational, since 2013, across the entire campus of Singapore Management University. We not only focus on the final "best solution" that uses RTLS data feeds, but also discuss the challenges and pitfalls encountered over the years. Our findings apply to all the server-side localization algorithms, which use fingerprinting techniques.

Real deployments have myriad of practical challenges that hamper the efficiency and the accuracy of an empirical study. For instance, there can be sudden and unexpected crowd movement which is known to increase signal variations. Furthermore, as and when required network administrators either replace old APs or deploy new APs. These administrative decisions are not under our control. However, such changes severely affect the offline fingerprints and change the floor heat maps that ultimately affect location accuracy. Preparing fingerprints for an entire campus with several thousands of landmarks is already tedious, such developments make the process of iterations even more cumbersome.

Beyond insufficient measurement and latency issues, various contextual dynamics makes the fingerprint-based system erroneous. The primary reason

is that such dynamic changes result in a significant fluctuation in RSSI measurements, which affects the distance calculation of the localization algorithms. These fluctuations can happen quite frequently as there are many different factors affecting RSSI between an AP and its clients, such as crowds blocking the signal path, AP-side power-control for load balancing, and client-side power control to save battery. In Section 4.2.2 we shed light on most of these factors. However, we leave a full evaluation for future work. Lastly, all MAC addresses in our system are anonymized. We do not do a device to person mapping to preserve user privacy.

A significant limitation of this work is that we have not considered an exhaustive set of devices. Given a multitude of device vendors, it is practically impossible to consider all set of devices for this kind of in-depth analysis. We did cover the latest set of devices, though, including iPhone and Android devices. The second limitation is that even though we collected the data for both lightly (semester off, very few students on campus) and heavily loaded (semester on, most students on campus) days. We tested our observations on the lightly loaded dataset but, only on a subset of heavily loaded days. We do not yet know the behavior of the system during heavily loaded days, in its entirety. Specifically, the load, concerning the number of clients and traffic is expected to increase interference and thus, signal variations. However, this study is still a part of future work. The third limitation of this work is that we do not consider the effect of MAC address randomization algorithms which make clients intractable. Although there is an active field of research that suggest ways to map randomized

MAC to actual MAC [96], given its complexity, we do not employ these.

7.1.3 Harnessing Active Scans for Data Transfer in WiFi-based IoT Nodes

We propose the ViFi protocol, an association-free WiFi protocol, for IoT nodes and demonstrate its end-to-end functioning. ViFi protocol enables an efficient data transmission for energy constrained IoT nodes in the WiFi networks where the default WiFi protocol fails. Performance of the non-IoT nodes operating with the default WiFi protocol is also unaffected by the transmissions of IoT nodes operating with the ViFi protocol. We demonstrate with a detailed MAC layer analysis *why*, *how*, and *in what* scenarios does the idea of association-free WiFi works. We show that with the ViFi protocol, it is possible to eliminate a lot of network traffic related to the connection establishment and maintenance, while still transmit data successfully. The elimination of this network traffic allows saving several folds of airtime, number of frames, and energy consumption of the nodes. Applications that are not time-critical and that do not require more than a few hundreds of bits-per-second to be transmitted will benefit from the proposed ViFi protocol. ViFi is apt for IoT nodes because, it transmits few bits-per-second without WiFi association, even at low RSSI, in a battery- and channel-friendly manner. Although clients can successfully transmit data with the ViFi protocol, we advise that the usage of this protocol should be meticulously planned. We discuss the advantages and limitations of the ViFi protocol.

The advantages of the ViFi protocol in comparison to the WiFi protocol are – (a) the ViFi protocol is both battery and network friendly as it requires lesser

number of frames and airtime than WiFi to transmit the same amount of data, (b) it is reliable as it can work even in poor network conditions, irrespective of the availability of a WiFi association, (c) it is scalable as the performance remains almost unaffected even with a large number of nodes, (d) it can work with any `IEEE 802.11` standard, therefore it is easy to integrate with existing WiFi network deployments, (e) it can co-exist with non-IoT nodes of the WiFi network with minimal effects (as compared to WiFi protocol) on their performance, and (f) lastly, it is cost effective as it does not need hardware changes to the IoT nodes or specific network deployment.

The limitations of the ViFi protocol are – (a) the probe requests during an active scan are broadcast on all channels at the lowest bit-rate. Excessive probe requests increase contention in heavily utilized WiFi networks, (b) the data generation interval cannot be lesser than the time taken by a client to conduct a full-band (all channels) active scan, and (c) the ViFi protocol experiences higher end-to-end delay than the WiFi protocol in good network conditions.

We propose to avert these limitations by setting an upper limit on the amount of data transmitted and the interval of data generation. Although the maximum number of bytes that can be stuffed in a probe request is 1394 Bytes, typically a probe request is less than 300 bytes. Therefore, we propose to stuff no more than 300 bytes. Intervals of data generation, as low as 1 second, should be avoided. Instead, the data should be aggregated for a specific duration before transmission. Infrequent and small-sized probe requests transmitted in ViFi protocol will not hamper the performance of the WiFi protocol.

We believe that these limitations do not hamper the performance of IoT nodes that are energy constrained, applications running on the top of them have low data requirements and are delay tolerant.

7.1.4 Extending the Insights and the Solutions to the Latest and Future Versions of the WiFi

Although there is active research going on towards mmWave WiFi, we envision that 2.4 GHz and 5 GHz frequency bands will continue to govern the usage of the WiFi for the end-users both in the home and enterprise settings. Primary reason being that most latest WiFi standards such as 802.11`ac` and `ad` as well as the off-the-shelf devices such as the IoT nodes, the smartphones, and the APs operate in either or both these frequency bands. Furthermore, clients need to discover the network before using the services provided by it. Hence, the network and AP discovery protocols, active scanning in our case, remain to be perpetual irrespective of the device or the protocol in use. Dense WiFi deployments will continue to expand as urbanized WiFi deployment is the need of the hour. And so are the services executing on top, for instance, localization and IoT communication. Therefore, we conclude that the problems identified in this thesis in conjunction with the insights learned and the solutions developed continue to be applicable to all present and future WiFi versions. In addition, our solutions are capable of enhancing the latest `IEEE` ammendments for dense WiFi networks such as `IEEE 802.11ai`.

7.2 Future Research Directions

Intelligent and connected networks are gaining momentum. The need of today's WiFi networks is that they should be able to automatically detect the cause of performance drop, even at single client granularity, and autonomously take action to alleviate it. This thesis is an initial step in that direction. Indeed, there remain several exciting and hard problems of the real-world WiFi networks that need an in-depth understanding as well as smart and integrable solutions. All the solutions should be practically realized, validated, and tested. We identify three potential research problems, that extend this thesis, to explore. These are presented as follows:

7.2.1 Performance Enhancement of Modern WiFi networks

We addressed one problem *i.e.* unnecessary active scans. However, there exist many more problems that should be detected as well as resolved in realtime. We used static rule-based metrics with sniffer-based packet captures to detect the problem. Although device agnostic, the sniffer is known to drop frames. A better option is to leverage the latest software defined networking paradigm for data collection from a single vantage point. The data collected across layers can then be mined with the appropriate machine learning algorithms for the problems. Once problems are identified, easy to integrate algorithms are needed to combat them. Both SDN and Machine Learning for WiFi are considered as promising solutions to boost the performance of modern WiFi networks [162].

For example, project BeHop [163] and a machine learning approach to control RTS and CTS [164]. An initial attempt from us in this direction is that, we have collected several thousands of instances of active scans in real-world and we are developing machine learning models to identify their causes.

7.2.2 WiFi-based Indoor Localization

Indoor localization with existing WiFi infrastructure is the most scalable of all existing solutions. Localization is the fundamental need of all cutting-edge contextual applications. However, a device-agnostic solution to accurately track the devices is not trivial, as is identified in this thesis. Furthermore, MAC address randomization to preserve the identity of a device is progressively employed in all latest WiFi devices such as smartphones and tablet. With the randomized MAC addresses, it becomes hard to localize a client unless its original MAC address is known. With a separate line of research claiming it is possible to map randomized MAC to original MAC [76]; localization process is further complicated. Efficient, battery-friendly, and device-agnostic solutions are needed to tackle these challenges.

7.2.3 Association-Free WiFi

This thesis presents a first end-to-end evaluated prototype of association-free protocol, ViFi, with various MAC layer metrics. With IoT progressively becoming the coherent part of large-scale and dense WiFi networks, there remains a significant portion of work left before ViFi matures. For example, the protocol,

ViFi, does not take care of recovering from failed data transmissions. It relies on implicit reliability that comes with active scanning. However, a reliability mechanism is needed to ensure the robustness of the ViFi protocol. Such a mechanism should take into account the need of retry attempts on failed transmissions while taking of the frequency of re-triggering the scanning. Also, this protocol, as of now, realizes uplink transmission only. Data transmission from AP to a client and its in-depth analysis that aids in understanding its feasibility of seamless integration with existing WiFi networks is needed. Since the latest device employs MAC address randomization for probe requests, it poses a significant challenge for the operation of the ViFi as well. Evaluation of this aspect is also left as an future work.

References

[1] D. Jaisinghani, V. Naik, S. K. Kaul, R. Balan, and S. Roy, "Improving the Performance of WLANs by Reducing Unnecessary Active Scans," *ArXiv e-prints*, Jul. 2018.

[2] D. Jaisinghani and H. Fulara and G. Singh and M. Maity and V. Naik, "Sending Data in WiFi-based IoT without the Data Frames," *Conference Not Revealed (Double Blind Submission)*, Under Review.

[3] D. Jaisinghani, V. Naik, S. K. Kaul, and S. Roy, "Realtime detection of degradation in WiFi network's goodput due to probe traffic," in *Proceedings of the 11th International Workshop on Wireless Network Measurements and Experimentation held in conjunction with 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2015.

[4] D. Jaisinghani and V. Naik and S. Kaul and S. Roy, "Sniffer-based Inference of the Causes of Active Scanning in WiFi Networks," in *Proceedings of the 23rd National Conference on Communications*, Chennai, India, Mar. 2017.

[5] H. Fulara and G. Singh and D. Jaisinghani and M. Maity and T. Chakraborty and V. Naik, "Learning to Rescue WiFi Networks from Unnecessary Active Scans," *Proceedings of the 20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2019.

[6] D. Jaisinghani and H. Fulara and G. Singh and M. Maity and V. Naik, "Demo: Sending Data in WiFi-based IoT without the Data Frames," *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018.

[7] D. Jaisinghani, R. Balan, V. Naik, Y. Lee, and A. Misra, "Experiences & Challenges with Server-Side WiFi Indoor Localization Using Existing Infrastructure," in *Proceedings of the EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018.

[8] IEEE, "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016*, 2016.

[9] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren, "Usage Patterns in an Urban WiFi Network," *IEEE/ACM Transactions on Networking*, 2010.

[10] A. Gember, A. Anand, and A. Akella, "A Comparative Study of Handheld and Non-handheld Traffic in Campus Wi-Fi Networks," in *Passive and Active Measurement*. Springer Berlin Heidelberg, 2011.

[11] W. Alliance, "Next generation Wi-Fi: The future of connectivity," [accessed 20-Dec-2018]. [Online]. Available: https://www.wi-fi.org

[12] L. Zhang, L. Zhao, Z. Wang, and J. Liu, "WiFi Networks in Metropolises: From Access Point and User Perspectives," *IEEE Communications Magazine*, 2017.

[13] Google, "Project Fi," [accessed 2-Dec-2018]. [Online]. Available: https://fi.google.com/about/international-rates/

[14] PC-Mag, "Google adds India to countries covered under Project Fi," [accessed 21-Feb-2018]. [Online]. Available: https://in.pcmag.com/google-project-fi/119193/google-adds-india-to-countries-covered-under-project-fi

[15] T. N. News, "Google's free Wi-Fi now at 400 Indian railway stations," [accessed 24-Dec-2018]. [Online]. Available: https://www.timesnownews.com/india/article/googles-free-wi-fi-now-at-400-indian-railway-stations/237170

[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, 2015.

[17] A. J. Ruiz-Ruiz, H. Blunck, T. S. Prentow, A. Stisen, and M. B. KjÃęr-gaard, "Analysis methods for extracting knowledge from large-scale WiFi monitoring to inform building facility planning," in *IEEE International Conference on Pervasive Computing and Communications*, 2014.

[18] K. Jayarajah, R. Balan, M. Radhakrishnan, A. Misra, and Y. Lee, "LiveLabs: Building In-Situ Mobile Sensing & Behavioural Experimentation TestBeds," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016. [Online]. Available: http://doi.acm.org/10.1145/2906388.2906400

[19] H. Chaskar, "Improving WiFi Connections in High-Density Environments," [Online; accessed 14-October-2017]. [Online]. Available: https://www.networkcomputing.com/wireless-infrastructure/improving-wifi-connections-high-density-environments/927799455

[20] IEEE, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Fast Initial Link Setup," *802.11ai*, 2016.

[21] M. Youssef, L. Shahamat, M. Kleene, and A. Agrawala, "The IEEE 802.11 active probing analysis and enhancements," in *International Conference on Wireless Networks, Communications and Mobile Computing*, June 2005.

[22] S. Lee, M. Kim, S. Kang, K. Lee, and I. Jung, "Smart scanning for mobile devices in WLANs," in *IEEE International Conference on Communica-*

tions (ICC), 2012.

[23] I. Ramani and S. Savage, "SyncScan: practical fast handoff for 802.11 infrastructure networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.

[24] Microsoft, "Scanning 802.11 Networks," [Online; accessed 5-January-2015]. [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/hardware/ff564113%28v=vs.85%29.aspx

[25] Cisco, "Voice over Wireless LAN 4.1 Design Guide - Voice over WLAN Roaming [Design Zone for Mobility] - Cisco," [Online; accessed 13-October-2014]. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/vowlan/41dg/vowlan41dg-book/vowlan_ch5.html

[26] I. Purushothaman and S. Roy, "FastScan: A Handoff Scheme for Voice over IEEE 802.11 WLANs," *Wireless Networks*, Oct. 2010.

[27] M. Emmelmann, S. Wiethoelter, and H.-T. Lim, "Opportunistic scanning: Interruption-free network topology discovery for wireless mesh networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops*, 2009.

[28] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings IEEE Conference on Computer Communications (INFOCOM)*, 2000.

[29] R. Nandakumar, K. Chintalapudi, and V. N. Padmanabhan, "Centaur: Locating Devices in an Office Environment," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348579

[30] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner, "Virtual Compass: Relative Positioning to Sense Mobile Social Interactions," in *Pervasive Computing*, 2010. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12654-3_1

[31] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit, "Place Lab: Device Positioning Using Radio Beacons in the Wild," in *Pervasive Computing*, 2005. [Online]. Available: https://doi.org/10.1007/11428572_8

[32] J. Xiong and K. Jamieson, "ArrayTrack: A Fine-Grained Indoor Location System," in *Published in 10th Usenix Symposium on Networked Systems Design and Implementation*, 2013. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xiong

[33] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter Level Localization Using WiFi," in *SIGCOMM*, 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787487

[34] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "PinPoint: An Asynchronous Time-Based Location Determination

System," in *4th ACM International Conference on Mobile Systems, Applications, and Services*, 2006. [Online]. Available: http://doi.acm.org/10.1145/1134680.1134698

[35] Stuart A. Golden and Steve S. Bateman, "Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging," in *IEEE Transactions on Mobile Computing*, 2007. [Online]. Available: http://dx.doi.org/10.1109/IEEETransactionsonMobileComputing.2007.1002

[36] Domenico Giustiniano and Stefan Mangold, "CAESAR: Carrier Sense-based Ranging in Off-the-shelf 802.11 Wireless LAN," in *Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies* , 2011. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079306

[37] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, "SAIL: Single Access Point-based Indoor Localization," in *MobiSys*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594393

[38] S. Sen, D. Kim, S. Laroche, K. Kim, and J. Lee, "Bringing CUPID Indoor Positioning System to Practice," in *Proceedings of the 24th International Conference on World Wide Web* , 2015. [Online]. Available: https://doi.org/10.1145/2736277.2741686

[39] Y. Zhu, Y. Zhu, B. Zhao, and H. Zheng, "Reusing 60GHz Radios for Mobile Radar Imaging," in *Proceedings of 21st Annual International*

Conference on Mobile Computing and Networking, 2015. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790112

[40] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu, "See Through Walls with COTS RFID System!" in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* , 2015. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790100

[41] T. Wei and X. Zhang, "mTrack: High-Precision Passive Tracking Using Millimeter Wave Radios," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* , 2015. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790113

[42] F. Zafari, A. Gkelias, and K. Leung, "A Survey of Indoor Localization Systems and Technologies," *ArXiv e-prints*, 2017.

[43] R. Balan, A. Misra, and Y. Lee, "LiveLabs: Building an In-situ Real-time Mobile Experimentation Testbed," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2565585.2565597

[44] Coleman and Westcott , *CWNA Certified Wireless Network Administrator Official Study Guide*. John Wiley & Sons, 2009.

[45] D. Niculescu and Badri Nath, "Ad hoc positioning system (APS) using AOA," in *22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003.

[46] P. Mirowski, T. K. Ho, Saehoon Yi, and M. MacDonald, "SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals," in *International Conference on Indoor Positioning and Indoor Navigation*, 2013.

[47] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 2012.

[48] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2007.

[49] Statista, "IoT: number of connected devices worldwide 2012-2025. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/," 2017, [Online; accessed 10-April-2018]. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[50] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart User Authentication Through Actuation of Daily Activities Leveraging WiFi-enabled IoT," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc, 2017.

[51] S. R. Pokhrel, C. Williamson, C. Williamson, and S. R. Pokhrel, "Modeling Compound TCP Over WiFi for IoT," *IEEE/ACM Transactions on*

Networking, 2018.

[52] Amazon, "Amazon announces Wi-Fi Simple Setup to easily setup new IoT devices. https://mspoweruser.com/amazon-announces-wi-fi-simple-setup-to-easily-setup-new-iot-devices/amp/," [Online; accessed 22-Sep-2018]. [Online]. Available: https://mspoweruser.com/amazon-announces-wi-fi-simple-setup-to-easily-setup-new-iot-devices/amp/

[53] M. Maity, B. Raman, and M. Vutukuru, "TCP Download Performance in Dense WiFi Scenarios: Analysis and Solution," *IEEE Transactions on Mobile Computing*, 2016.

[54] R. Raghavendra, E. Belding, K. Papagiannaki, and K. Almeroth, "Unwanted Link Layer Traffic in Large IEEE 802.11 Wireless Networks," *IEEE IEEE Transactions on Mobile Computing*, 2010.

[55] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI, 2011.

[56] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill, "Designing High Performance Enterprise Wi-Fi Networks," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2008.

[57] J. Shi, L. Meng, A. Striegel, C. Qiao, D. Koutsonikolas, and G. Challen, "A walk on the client side: Monitoring enterprise Wifi networks using smartphone channel scans," in *IEEE INFOCOM - The 35th Annual IEEE International Conference on Computer Communications*, 2016.

[58] Cisco, "MAC Filters with Wireless LAN Controllers (WLCs) Configuration Example. https://www.cisco.com/c/en/us/support/docs/wireless-mobility/wlan-security/91901-mac-filters-wlcs-config.html," 2018. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/wireless-mobility/wlan-security/91901-mac-filters-wlcs-config.html

[59] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra, "Per-frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design," *IEEE/ACM Transactions on Networking*, 2015.

[60] Y. Sun, J. Chen, Y. Tang, and Y. Chen, "Energy Modeling of IoT Mobile Terminals on WiFi Environmental Impacts," *Sensors*, 2018.

[61] W. Sun and J. Liu, "Coordinated Multipoint-Based Uplink Transmission in Internet of Things Powered by Energy Harvesting," *IEEE Internet of Things Journal*, 2018.

[62] U. Muncuk, K. Alemdar, J. D. Sarode, and K. R. Chowdhury, "Multiband Ambient RF Energy Harvesting Circuit Design for Enabling Batteryless Sensors and IoT," *IEEE Internet of Things Journal*, 2018.

[63] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian, "Wi-Fi enabled sensors for internet of things: A practical approach," *IEEE Communications Magazine*, 2012.

[64] L. Huang and T.-H. Lai, "On the Scalability of IEEE 802.11 Ad Hoc Networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, ser. MobiHoc, 2002.

[65] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman, "Beacon-Stuffing: Wi-Fi without Associations," in *Proceedings of the 8th IEEE Workshop on Mobile Computing Systems and Applications*, 2007.

[66] M. Olbrich, A. Zubow, S. Zehl, and A. Wolisz, "WiPLUS: Towards LTE-U Interference Detection, Assessment and Mitigation in 802.11 Networks," in *Proceedings of the 23th European Wireless Conference*, 2017.

[67] E. R. Lira, E. Fynn, P. R. S. L. Coelho, L. F. Faina, L. Camargos, R. S. VillaÃğa, and R. Pasquini, "An Architecture for Traffic Sign Management in Smart Cities," in *Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications*, 2016.

[68] O. A. Oun, C. Bloch, and F. Spies, "Indoor positioning using CoLDE: An IEEE 802.11 connectionless extension," in *International Conference on Indoor Positioning and Indoor Navigation*, 2014.

[69] T. D. Nguyen, Q. M. Tran, and V. P. Tran, "Poster: Decentralized Disaster Recovery Networks Using Beacon Stuffing," in *Proceedings of the 14th*

Annual International Conference on Mobile Systems, Applications, and Services Companion, 2016.

[70] Y. chung Cheng, J. Bellardo, P. BenkÃű, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw: Solving the puzzle of enterprise 802.11 analysis," in *SIGCOMM Computer Commununication Review*, 2006.

[71] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "Pie in the sky: online passive interference estimation for enterprise wlans," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011.

[72] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale Measurements of Wireless Network Behavior," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM, 2015.

[73] Cisco, "MIB Compilers and Loading MIBs," [accessed 14-November-2016]. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/26015-mibcompilers.html

[74] L. Phifer, "Buyer's Guide to Enterprise WLAN Controllers," 2011, [Online; accessed 9-October-2014]. [Online]. Available: http://www.enterprisenetworkingplanet.com/netsysm/article.php/3924291/Buyers-Guide-to-Enterprise-WLAN-Controllers.html

[75] J. Teng, C. Xu, W. Jia, and D. Xuan, "D-Scan: Enabling Fast and Smooth Handoffs in AP-Dense 802.11 Wireless Networks," in *The 28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[76] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC Address Randomization in Mobile Devices and When it Fails," *Proceedings on Privacy Enhancing Technologies*, 2017.

[77] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda, "Characterizing and Improving WiFi Latency in Large-Scale Operational Networks," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services* , 2016.

[78] M. Muuss, "ping(8) - linux man page," [accessed 12-January-2016]. [Online]. Available: https://linux.die.net/man/8/ping

[79] A. Schulman, D. Levin, and N. Spring, "CRAWDAD dataset umd/sigcomm2008 (v. 2009-03-02)," Mar. 2009.

[80] W. L. Tan, M. Portmann, and P. Hu, "A Systematic Evaluation of Interference Characteristics in 802.11-Based Wireless Networks," in *IEEE International Conference on Advanced Information Networking and Applications*, 2011.

[81] W. Kim, J. Lee, T. Kwon, S.-J. Lee, and Y. Choi, "Quantifying the Interference Gray Zone in Wireless Networks: A Measurement Study," in *IEEE International Conference on Communications*, 2007.

[82] U. Das and C. Hood, "Using a Shielded Room to Characterize UDP Performance in the Presence of Interference in IEEE 802.11 Wireless Networks," in *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2008.

[83] S. Khurana, A. Kahol, and A. Jayasumana, "Effect of hidden terminals on the performance of IEEE 802.11 MAC protocol," in *Local Computer Networks. Proceedings., 23rd Annual Conference on*, 1998.

[84] Y. Zhou and S. Nettles, "Balancing the hidden and exposed node problems with power control in CSMA/CA-based wireless networks," in *Wireless Communications and Networking Conference, IEEE*, 2005.

[85] L. Wang, K. Wu, and M. Hamdi, "Combating Hidden and Exposed Terminal Problems in Wireless Networks," *Wireless Communications, IEEE Transactions on*, 2012.

[86] M. Loiacono, J. Rosca, and W. Trappe, "The Snowball Effect: Detailing Performance Anomalies of 802.11 Rate Adaptation," in *IEEE Global Telecommunications Conference*, 2007.

[87] iperf, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool," [accessed 12-October-2016]. [Online]. Available: https://iperf.fr/

[88] J. Yeo, M. Youssef, and A. Agrawala, "Characterizing the IEEE 802.11 traffic: the wireless side," in *CS-TR-4570*. UMD, 2004.

[89] Prism, "LINKTYPE_IEEE802_11_PRISM | TCPDUMP/LIBPCAP public repository," [accessed 12-October-2016]. [Online]. Available: http://www.tcpdump.org/linktypes/LINKTYPE_IEEE802_11_PRISM.html

[90] Radiotap, "Radiotap," [accessed 12-October-2016]. [Online]. Available: http://www.radiotap.org/

[91] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, 2011.

[92] H. C. Lee, K. T. Kim, H. Y. Youn, and O. Song, "An efficient AP channel scanning scheme for real-time streaming over WLAN," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops*, 2010.

[93] J. Malinen, "Linux WPA/WPA2/IEEE 802.1X Supplicant," [accessed 24-February-2017]. [Online]. Available: https://w1.fi/wpa_supplicant/

[94] ——, "wpa_supplicant / hostapd 2.5," [accessed 24-February-2017]. [Online]. Available: https://w1.fi/wpa_supplicant/devel/bgscan_8c.html

[95] Aruba, "Deep dive Radio technologies for indoor location," https://community.arubanetworks.com/aruba, 2017.

[96] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC Address Randomization in Mobile Devices and When it Fails," *CoRR*, 2017.

[97] Y. Chapre, P. Mohapatra, S. Jha, and A. Seneviratne, "Received signal strength indicator and its analysis in a typical WLAN system (short paper)," in *38th Annual IEEE Conference on Local Computer Networks*, 2013.

[98] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC, 2009.

[99] Libnl, "Netlink Protocol Library Suite (libnl)," 2018, [Online; accessed 10-April-2018]. [Online]. Available: https://www.infradead.org/~tgr/libnl/

[100] IEEE, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *802.11-2016*, 2016.

[101] Ubuntu, "iw - show / manipulate wireless devices and their configuration. http://manpages.ubuntu.com/manpages/trusty/man8/iw.8.html," 2018, [Online; accessed 10-April-2018]. [Online]. Available: http://manpages.ubuntu.com/manpages/trusty/man8/iw.8.html

[102] Hostapd, "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator," 2018, [Online; accessed 10-April-2018]. [Online]. Available: https://w1.fi/hostapd/

[103] R. Raghavendra, J. Padhye, R. Mahajan, and E. Belding, "Wi-Fi Networks are Underutilized," Microsoft Research, Tech. Rep. MSR-TR-2009-108, 2009.

[104] K. Kim, H. Nam, and H. Schulzrinne, "WiSlow: A Wi-Fi network performance troubleshooting tool for end users," in *IEEE INFOCOM Conference on Computer Communications*, 2014.

[105] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *SIGCOMM Computer Communication Review*, 2003.

[106] V. Gupta, R. Beyah, and C. Corbett, "A Characterization of Wireless NIC Active Scanning Algorithms," in *IEEE Wireless Communications and Networking Conference*, 2007.

[107] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is There WiFi Yet?: How Aggressive Probe Requests Deteriorate Energy and Throughput," in *Proceedings of the Internet Measurement Conference*, ser. IMC, 2015.

[108] Y.-S. Chen, M.-C. Chuang, and C.-K. Chen, "DeuceScan: Deuce-Based Fast Handoff Scheme in IEEE 802.11 Wireless Networks," *IEEE Transactions on Vehicular Technology*, 2008.

[109] I. Purushothaman and S. Roy, "Fastscan: A handoff scheme for voice over ieee 802.11 wlans," *Wireless Networks*, October 2010.

[110] H. C. Lee, K. T. Kim, H. Y. Youn, and O. Song, "An efficient AP channel scanning scheme for real-time streaming over WLAN," in *Published in*

8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010.

[111] S. Jin and S. Choi, "A Seamless Handoff With Multiple Radios in IEEE 802.11 WLANs," *IEEE Transactions on Vehicular Technology*, 2014.

[112] S. Sen, B. Radunovic, J. Lee, and K.-H. Kim, "CSpy: Finding the Best Quality Channel without Probing," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, September 2013.

[113] A. Croitoru, D. Niculescu, and C. Raiciu, "Towards Wifi Mobility without Fast Handover," in *Published in 12th Usenix Symposium on Networked Systems Design and Implementation*, 2015.

[114] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, 2015.

[115] R. Frank, W. Bronzi, G. Castignani, and T. Engel, "Bluetooth Low Energy: An alternative technology for VANET applications," in *11th Annual Conference on Wireless On-demand Network Systems and Services*, 2014.

[116] M. Hasan, E. Hossain, and D. Niyato, "Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches," *IEEE Communications Magazine*, 2013.

[117] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," in *IEEE Wireless Communications and Networking Conference Workshops* , 2012.

[118] Evgeny Khorov and Andrey Lyakhov and Alexander Krotov and Andrey Guschin, "A survey on ieee 802.11ah: An enabling networking technology for smart cities," *Computer Communications*, 2015, Special Issue on Networking and Communications for Smart Cities.

[119] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, "IEEE 802.11ah: the WiFi approach for M2M communications," *IEEE Wireless Communications*, 2014.

[120] B. Bellalta, "IEEE 802.11ax: High-efficiency WLANS," *IEEE Wireless Communications*, 2016.

[121] E.-J. Kim, J.-H. Kwon, K. Choi, and T. Shon, "Unified Medium Access Control Architecture for Resource-Constrained Machine-to-Machine Devices," *ACM Transactions on Embedded Computing Systems*, 2016.

[122] C. Han, J. M. Jornet, E. Fadel, and I. F. Akyildiz, "A cross-layer communication module for the Internet of Things," *Computer Networks*, 2013.

[123] M. Rahaim, T. D. Little, H. Elgala, and S. Govindasamy, "An Ultra-Dense IoT Architecture Using Hybrid CSMA with Sector Based Scheduling (CSMA/SS) via Visible Light Communications," in *Proceedings of*

the International Conference on Embedded Wireless Systems and Networks, ser. EWSN, 2017.

[124] A. Bakshi, L. Chen, K. Srinivasan, C. E. Koksal, and A. Eryilmaz, "EMIT: An efficient MAC paradigm for the Internet of Things," in *IEEE INFOCOM - The 35th Annual IEEE International Conference on Computer Communications*, 2016.

[125] H. Lu and W. Gao, "Continuous Wireless Link Rates for Internet of Things," in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN, 2018.

[126] S. Ham, J. Lee, and K. Lee, "QuickTalk: An Association-Free Communication Method for IoT Devices in Proximity," *Proceedings of the ACM Interactive, Mobile, Wearable, and Ubiquitous Technologies*, 2017.

[127] S. Zehl, A. Zubow, and A. Wolisz, "Demo: Stuffing Wi-Fi Beacons for Fun and Profit," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom, 2017.

[128] S. Zehl, N. Karowski, A. Zubow, and A. Wolisz, "LoWS: A complete Open Source solution for Wi-Fi beacon stuffing based Location-based Services," in *Proceedings of the 9th IFIP Wireless and Mobile Networking Conference*, 2016.

[129] H. Wirtz, T. Zimmermann, M. Ceriotti, and K. Wehrle, "CA-Fi: Ubiquitous mobile wireless networking without 802.11 overhead and restric-

tions," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[130] K. Dhondge, S. Song, B. Y. Choi, and H. Park, "WiFiHonk: Smartphone-Based Beacon Stuffed WiFi Car2X-Communication System for Vulnerable Road User Safety," in *IEEE 79th Vehicular Technology Conference*, 2014.

[131] S. Zehl, A. Zubow, M. Doring, and A. Wolisz, "ResFi: A secure framework for self organized Radio Resource Management in residential WiFi networks," in *IEEE 17th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"*, 2016.

[132] U. G. Acer, A. Boran, C. Forlivesi, W. Liekens, F. PÃl'rez-cruz, and F. Kawsar, "Sensing WiFi network for personal IoT analytics," in *Proceedings of the 5th International Conference on the Internet of Things*, 2015.

[133] J. Krumm and J. Platt, "Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System," NIPS, Tech. Rep., 2003.

[134] M. Azizyan, I. Constandache, and R. Choudhury, "SurroundSense: Mobile Phone Localization via Ambience Fingerprinting," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking* , 2009. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614350

[135] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You Are Facing the Mona Lisa: Spot Localization Using PHY Layer Information," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307654

[136] M. Youssef and A. Agrawala, "The Horus WLAN Location Determination System," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services* , 2005. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067193

[137] A. Rai, K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort Crowdsourcing for Indoor Localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348580

[138] Z. Yang, C. Wu, and Y. Liu, "Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348578

[139] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the Limit of WiFi Based Localization for Smartphones," in *Proceedings of the 18th Annual International*

Conference on Mobile Computing and Networking, 2012. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348581

[140] K. Chintalapudi, A. Iyer, and V. N. Padmanabhan, "Indoor Localization Without the Pain," in *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking* , 2010. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860016

[141] H. Lim, L. Kung, J. C. Hou, and H. Luo, "Zero-configuration Indoor Localization over IEEE 802.11 Wireless Infrastructure," in *Wireless Networks*, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11276-008-0140-3

[142] A. Goswami, L. E. Ortiz, and S. R. Das, "WiGEM: A Learning-based Approach for Indoor Localization," in *Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies* , 2011. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079299

[143] D. Turner, S. Savage, and A. C. Snoeren, "On the empirical performance of self-calibrating WiFi location systems," in *IEEE 36th Conference on Local Computer Networks* , 2011.

[144] L. Li, G. Shen, C. Zhao, T. Moscibroda, J. Lin, and F. Zhao, "Experiencing and Handling the Diversity in Data Density and Environmental Locality in an Indoor Positioning Service," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639118

[145] C. Wu, Z. Yang, Y. Liu, and W. Xi, "WILL: Wireless Indoor Localization without Site Survey," in *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[146] Yuanfang Chen, Noel Crespi, Lin Lv, Mingchu Li, Antonio M. Ortiz, and Lei Shu, "Locating Using Prior Information: Wireless Indoor Localization Algorithm," in *Proceedings of the ACM SIGCOMM Computer Communications Review*, 2013. [Online]. Available: http://doi.acm.org/10.1145/2486001.2491688

[147] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. Choudhury, "No Need to War-drive: Unsupervised Indoor Localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307655

[148] S. Tarzia, P. Dinda, R. Dick, and G. Memik, "Indoor Localization Without Infrastructure Using the Acoustic Background Spectrum," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, 2011.

[149] Cisco, https://www.cisco.com, 2017.

[150] Aruba, https://www.arubanetworks.com, 2017.

[151] S. Kumar, S. Gil, D. Katabi, and D. Rus, "Accurate Indoor Localization with Zero Start-up Cost," in *Proceedings of the 20th Annual*

International Conference on Mobile Computing and Networking, 2014. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639142

[152] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson, "Phaser: Enabling Phased Array Signal Processing on Commodity WiFi Access Points," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639139

[153] D. Huang, R. Nandakumar, and S. Gollakota, "Feasibility and Limits of Wi-Fi Imaging," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems* , 2014. [Online]. Available: http://doi.acm.org/10.1145/2668332.2668344

[154] W. Wang, A. Liu, M. Shahzad, K. Ling, and S. Ling, "Understanding and Modeling of WiFi Signal Based Human Activity Recognition," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* , 2015. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790093

[155] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao, "Epsilon: A Visible Light Based Positioning System," in *Published in 11th Usenix Symposium on Networked Systems Design and Implementation*, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2616448.2616479

[156] P. Hu, L. Li, C. Peng, G. Shen, and F. Zhao, "Pharos: Enable Physical Analytics Through Visible Light Based Indoor Localization," in

Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks , 2013. [Online]. Available: http://doi.acm.org/10.1145/2535771.2535790

[157] N. Hassan, A. Naeem, Md A. Pasha, T. Jadoon, and C. Yuen, "Indoor Positioning Using Visible LED Lights: A Survey," *ACM Computing Surveys*, 2015. [Online]. Available: http://doi.acm.org/10.1145/2835376

[158] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "SwordFight: Enabling a New Class of Phone-to-phone Action Games on Commodity Phones," in *10th ACM International Conference on Mobile Systems, Applications, and Services*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307638

[159] S. Sur, T. Wei, and X. Zhang, "Autodirective Audio Capturing Through a Synchronized Smartphone Array," in *Proceedings of the 12th International Conference on Mobile Systems, Applications, and Services*, 2014. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594380

[160] V, Erdélyi, T. Le, B. Bhattacharjee, P. Druschel, and N. Ono, "Sonoloc: Scalable positioning of commodity mobile devices," in *16th ACM International Conference on Mobile Systems, Applications, and Services*, 2018.

[161] Md A. Khan, H. Hossain, and N. Roy, "Infrastructure-less Occupancy Detection and Semantic Localization in Smart Environments," in *Proceedings of the 12th EAI International Conference on Mobile*

and Ubiquitous Systems: Computing, Networking and Services, 2015. [Online]. Available: http://dx.doi.org/10.4108/eai.22-7-2015.2260062

[162] Á. López-Raventós, F. Wilhelmi, S. Barrachina-Muñoz, and B. Bellalta, "Machine Learning and Software Defined Networks for High-Density WLANs," *CoRR*, 2018.

[163] Y. Yiakoumis and M. Bansal and A. Covington and J. Reijendam and S. Katti and N. McKeown, "BeHop: A Testbed for Dense WiFi Networks," in *Proceedings of the 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2014.

[164] Y. Edalat and K. Obraczka and B. Amiri, "A machine learning approach for dynamic control of RTS/CTS in WLANs," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018.

[165] J. Meyers, "Python_DFA," [accessed 24-March-2017]. [Online]. Available: https://github.com/meyersj/python_dfa/blob/master/dfa.py

[166] D. Jaisinghani, "GitHub Code Repository," [accessed 20-March-2017]. [Online]. Available: https://github.com/dheryta/CausalAnalysis

Appendices

Appendix A

Inference Mechanism – Implementation Details

A.1 Inference Mechanism Theory

In this section, we discuss the theory behind the inference mechanism that makes it suitable for cause detection. We developed a Causal Model of Active Scanning (CMAS) to implement the inference mechanism.

Preliminaries First, we introduce the notion of states, events, and state transitions that we derived from empirical experience of analyzing the traffic from various clients. These states, events, and state transitions form the CMAS. Figure A.1 represents CMAS. We identify 11 states of a client with scanning and not-scanning as final states, that can be reached from any state. Note that we do not show final states in CMAS due to brevity. Table A.1 lists these states. A client transitions from one state to another on occurrence of an event. We define an event as a change in the characteristics of frames transmitted by a client or

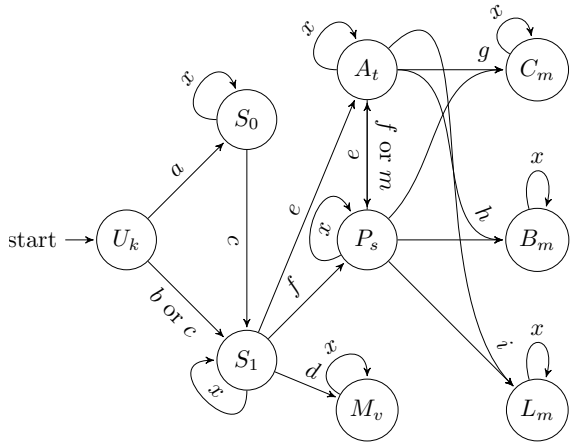


Figure A.1: The Causal Model of Active Scanning. Abbreviations of states are given in Table A.1. Details of events, which are input to CMAS are given in Table A.2.

Table A.1: CMAS States

State	Abbreviation
Unknown	U_k
Unassociated	S_1
Associated	S_0
PowerSave	P_s
Active	A_t
Moving	M_v
Connection Monitor	C_m
Beacon Monitor	B_m
Loss Monitor	L_m
Scanning	S_g
Not Scanning	$\neg S_g$

by an AP to which the client is associated. These events indicate the state of client by monitoring the association status, authentication status, variation in RSSI, frame arrival rate, loss of beacons, unsuccessful frame delivery, and type of probe requests. Table A.2 lists these events.

These events are analogous to the inference rules, and metrics discussed in Section to detect the 7 causes of active scanning. For example, event i that represents an increase in frame retries or ACK losses or reduction in data rates corresponds to the *Data frame losses* cause. Similarly, event d that accounts

Table A.2: Details of Events

Event	Represents
a	Un-Successful Association or Authentication or Re-association or Deauthentication
b	Class 3 Frames - Data frames between client and AP, PS-Poll, Block ACK, Block ACK Request, Action Frames.
c	Successful Association or Authentication or Re-association or Class 3 Frames
d	Average signal values in frames fall below -72 dBm and standard deviation is more than 12
e	Frame arrival rate $>$ 2 frames per second
f	Frame arrival rate $<=$ 2 frames per second
g	AP de-authentication
h	Un-Acknowledged Null Data Frame with Power Save Bit = 0 or Inter-Frame Arrival Time of Beacon more than Beacon Interval
i	Frame retries increase by more than 50% or Acknowledgement losses increase by more than 50% or Data rates reduce by more than 50%
j	Probe Requests
k	Probe Requests directed to AP
l	No Probe Requests
m	Client de-authentication
x	Miscellaneous Input

for a variation in signal values for frames from a client corresponds to the RSSI cause. For every cause, we have defined one or more state transitions. Table A.3 lists these state transitions.

Episode of active scan To detect the cause of an active scan, we first need to identify it uniquely. However, in a given sniffer-based log with lots of probe requests it is a challenge to determine different *episodes* of active scanning uniquely. Reason being we are using a sniffer to record traffic, which ends up capturing the probe requests on overlapping channels as well due to overhearing. To address this challenge, we observe the time intervals of probe requests from one client as recorded by the sniffer. For one active scan, recording of a group of probe requests is at intervals of milliseconds or lesser. The timestamp of first

probe request from such a group identifies an episode of active scan. Different *episodes* of active scans are separated by one or more seconds.

Window We define a *window* as a group of frames between the previous and current episode of an active scan. For a given *window*, we do not consider each frame individually. Rather, we extract the events from the *window* of frames. Along with radiotap or prism headers, the information carried in MAC header of frames help us to retrieve these events. Table A.2 provides a list of events.

An example of active scanning episodes separated by a *window* is shown in Figure 3.7.

Regular expressions for CMAS Now, we prepare the input for CMAS. A letter is assigned to each event, as shown in Table A.2. x denotes a miscellaneous event at a state whose transition is undefined in that state. We process all frames in a window, to look for the presence of these events. As a result of this processing, we get a string of the form – ^[abc]*[$defghimx$]?[jkl]$. Strings of this form are input to CMAS; thereby guessing the states and the corresponding transitions for the client. Eventually, these state changes detect a particular cause of active scanning. We show a mapping of state transitions and causes in Table A.3.

We define CMAS as a DFA that captures how the causes result into active scanning. Figure A.1 shows the CMAS. DFA is an apt representation for this purpose because – (i) every frame is transmitted due to some event on a client or AP and (ii) these events form strings of a regular language that identify a

Table A.3: State Transitions for Cause Detection

State Transitions	Sample Strings	Detected Cause
($S_0 \rightarrow S_g$) or ($S_1 \rightarrow S_g$) or ($A_t \rightarrow S_g$) or ($P_s \rightarrow S_g$) or ($U_k \rightarrow S_g$)	aj, bj	*Periodic Scan*
($S_0 \rightarrow S_1 \rightarrow S_g$) or ($S_0 \rightarrow S_1 \rightarrow A_t \rightarrow S_g$)	acj, $acej$	*Associated State: Unassociated to Associated*
($P_s \rightarrow A_t \rightarrow S_g$)	$bemej$, $cfek$	*Power State: Low to High*
($B_m \rightarrow S_g$)	$cehk$, $bfhk$	*Loss of Beacons*
($C_m \rightarrow S_g$)	$acegj$, $acfgj$	*AP-side Procedures*
($M_v \rightarrow S_g$)	bdk, $acdk$	*Low RSSI*
($L_m \rightarrow S_g$)	$ceik$, $bfik$	*Data Frame Losses*

cause of active scanning if accepted by CMAS.

Detecting the cause String processing begins in CMAS from U_k state and ends with either of S_g or $\neg S_g$ state. Frames in the window allow us to derive events which we represent with the alphabets. For a cause to be detected, the strings begin with a, b, or c and end with j, k, or l. With start and end specified, remaining substring can take any of the alphabets – d, e, f, g, h, i, m, or x. We consider only those events that are required to detect the causes, x represents rest of the events. Thus, strings of the form ^[abc]*[$defghimx$]?[jkl]$, when presented to CMAS, will detect a particular cause of active scanning. For example, string aj and bj detects periodicity when the client is in unassociated and associated state, respectively. Similarly, string $cefhj$ recognizes *Loss of beacons*. Table A.3, lists example strings for each cause. We show a detailed example of detecting the cause – *AP-side procedures* in Figure A.2.

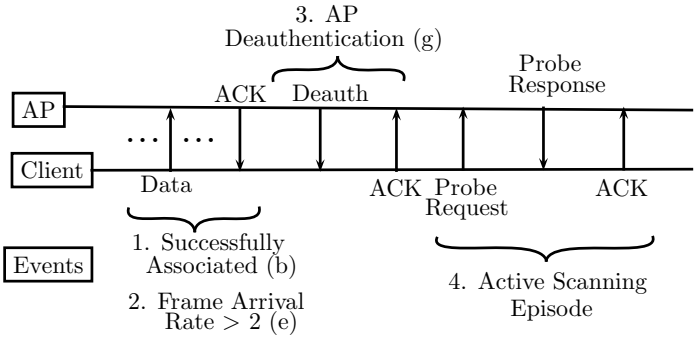


Figure A.2: An example to show the process of detecting a cause through a window of frames. In this example, the detected cause is *AP-side procedures*. The client is associated and transmitting more than 2 frames per second (dots signify frame exchange). Thereby, causing events b and e, respectively. Then, AP sends a deauthentication (deauth) frame to the client, which is acknowledged by the client; this is event g. As a result of this deauthentication, client triggers active scanning, hence causing event j. The string formed is $begj$. The corresponding state transition in CMAS is: $C_m \rightarrow S_g$, that detects the cause *AP-side procedures*.

Algorithm 1: Cause Detection Procedure

```
1  MAC_Addresses[] ← Extract_Clients();
2  for MAC ∈ MAC_Addresses[] do
3      episodes[] ← Extract_Episodes(MAC);
       // Initialize, end of previous window as invalid
4      prevWndEnd ← INVALID;
5      for currEpisode ∈ episodes[] do
6          if prevWndEnd == INVALID then
7              window ← Extract_Frames(currEpisode − 1, currEpisode, MAC);
8          else
9              window ← Extract_Frames(prevWndEnd + 1, currEpisode, MAC);
10         prevWndEnd ← currEpisode;
11         events ← Extract_Events(window);
12         detectedCause ← CMAS(events);
13         Print detectedCause
```

A.2 Algorithm to Implement the Inference Mechanism

The CSVs containing network traffic captured by a sniffer are processed for each client to find the episodes, windows, and the events which are then fed to the CMAS algorithm. CMAS algorithm then processes these events to detect a cause for the episodes of active scanning. Algorithm 1 summarizes steps for detecting the causes. Algorithm 2 summarizes the functions used in Algorithm

Algorithm 2: Cause Detection Functions

```
1  Function Extract_Clients()
2      Return MAC addresses of all devices that are not transmitting beacons;
3  Function Extract_Episodes(MAC)
4      Return timestamps of probe requests s.t. inter-probe request time > 1 sec;
5  Function Extract_Frames(timestamp_previous, timestamp_current, MAC)
6      Return all frames transmitted or received by MAC between timestamp_previous and
         timestamp_current;
7  Function Extract_Events(window)
8      Navigate the events table (Table A.2), return the string of letters corresponding to all events
         matching in window;
9  Function CMAS(events)
10     Based on events, find the state transition from DFA, match in State Transition table A.3, return
         the cause;
```

1. Algorithm 1 works as follows. First, it extracts MAC Addresses of all clients in Line 2. For every MAC address found in previous step, it partitions captured active scanning frames corresponding to the client into non-overlapping *episodes* of active scanning in Line 4. Now, it finds cause of each active scanning episode found in previous step. For every such instance, we find a *window* of frames that precedes the active scanning episode. This *window* extends to the end of the episode of active scanning, of the same client, that preceded the current episode. Lines 7-9 find the window of frames to be analyzed. It looks up the information carried in these frames and derives the events in Line 11 using Table A.2. Finally, it detects and prints the cause of active scanning by matching the events through the CMAS in Lines $12 - 13$ using Table A.3.

CMAS Implementation CMAS is implemented with Shell scripts and Python. Source code of the entire system, except the part that implements DFA [165], is written by us. It is open-source and is available here [166].

Appendix B

Inference Mechanism – Larger Dataset

Entire dataset is collected in a controlled environment where no causes overlap. Table B.1 shows the duration in hours for which the data collected for each cause and the corresponding number of episodes collected. This dataset was collected to train the machine learning models to automate the process of inferring the cause. Details about the machine learning models can be found in [5]. The number of episodes, N, presented in Table B.1 are several folds higher than the prior dataset presented in Table 3.6. We automated the process of data collection with several client-side scripts, apps, and AP-side automation. For instance, we enabled automatic triggering of the cause *AP-side Procedures* with the Selenium Web Browser Automation Tool to change password dynamically and accordingly made changes at the client with `wpa_supplicant` to manage the ongoing WiFi association. For the cause *Loss of Beacons*, we used `hostapd` daemon to periodically turn the AP on and off. For the cause *Power State: Low to High*, we developed an Android app that turned the screen of the phone off and on at regular periods. We were not able to automate the cause

Cause	Hours	N	$\mathbf{T_p}$	$\mathbf{F_n}$	$\mathbf{F_p}$	P	R	F − Score
Periodic Scan (Unassociated)	21	7070	7070	0	2162	0.77	1.00	0.86
Periodic Scan (Associated)	22	4849	2820	2029	35	0.99	0.58	0.73
Connection Establishment	121	29463	28883	580	1666	0.95	0.98	0.96
Power State: Low to High	25	2549	1224	1325	111	0.92	0.48	0.63
Loss of Beacons	49	36273	34698	1575	3471	0.91	0.96	0.93
AP-side Procedures	97	3339	1711	1628	2	0.91	0.51	0.68
Low RSSI	7	657	347	310	310	1.00	0.53	0.69

Table B.1: Accuracy of the proposed inference mechanism on a larger dataset.

Low RSSI, hence there are few episodes for this cause as compared to other causes. Note that Table B.1 does not include episodes for the cause *Data Frame Losses*. There are a couple of reasons for that. First, as we included a lot more and newer devices than in Table 3.6 we find that not all drivers perceive connection quality with data frame losses, instead they rely on beacon losses, and secondly, even if they do, it is hard to create a scenario in the controlled setup with a limited set of devices where we ensure frame retries to such a level that client triggers active scan. Nevertheless, we have left its further exploration as part of future work. Table B.1 also shows the accuracy of the inference mechanism. The accuracy of the inference mechanims is at par even with a large number of episodes collected.