5-29-2008

A Dynamically Refocusable Sampling Infrastructure for 802.11 Networks

Udayan Deshpande
Dartmouth College

A Dynamically Refocusable Sampling Infrastructure for 802.11 Networks

(Dartmouth Computer Science Technical Report TR2008-620)

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Udayan Deshpande

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 29th, 2008

Examining Committee:

(chair) David Kotz

Brian Noble

Andrew Campbell

Chris Bailey-Kellogg

Charles Barlowe
Dean of Graduate Students

Abstract

The edge of the Internet is increasingly wireless. Enterprises large and small, homeowners, and even whole cities have deployed Wi-Fi networks for their users, and many users never need to— or never bother to— use the wired network. With the advent of high-throughput wireless networks (such as 802.11n) some new construction, even of large enterprise buildings, may no longer be wired for Ethernet. To understand Internet traffic, then, we need to understand the wireless edge.

Measuring Wi-Fi traffic, however, is challenging. It is insufficient to capture traffic in the access points, or upstream of the access points, because the activity of neighboring networks, ad hoc networks, and physical interference cannot be seen at that level. To truly understand the MAC-layer behavior, we need to capture frames from the air using Air Monitors (AMs) placed in the vicinity of the network. Such a capture is always a *sample* of the network activity, since it is physically impossible to capture a *full* trace: all frames from all channels at all times in all places.

We have built a monitoring infrastructure that captures frames from the 802.11 network. This infrastructure includes several "channel sampling" strategies that will capture representative traffic from the network. Further, the monitoring infrastructure needs to modify its behavior according to feedback received from the downstream consumers of the captured traffic in case the analysis needs traffic of a certain type. We call this technique "refocusing". The "coordinated sampling" technique improves the efficiency of the monitoring by utilizing the AMs intelligently.

Finally, we deployed this measurement infrastructure within our Computer Science building to study the performance of the system with real network traffic.

ACKNOWLEDGEMENTS

Foremost, I thank my advisor, David Kotz, by whose research insights and patience I am always amazed. He has taught me a great deal not only about how to pursue research but also how to conduct any effort in general.

From Chris McDonald I have learnt how to keep a positive attitude in the face of tough odds. I have also learnt from him that usually the best way to attack any systems problem is to write a bit of code.

I thank Andrew Campbell, for showing me how sheer persistence (especially on the squash court) can win you many points. Thanks to Chris Bailey-Kellogg and Brian Noble for their willingness to be on my committee and for their advice in improving this dissertation.

Thanks to Bennet Vance to whose vast store of knowledge in Computer Science and other subjects I have often referred and hope to continue doing so in the future.

Thanks are due to the members of the CMC lab (Tristan, Keren, Yong, Josh) whose help has been invaluable at many points during the course of this research.

I thank my wife, Ramnika, because of whom these final months of writing have never felt overwhelming.

I thank my parents, Vijaya and Jayant, for putting me on the path that led to this goal. It was indeed a worthwhile experience.

I thank my sister, Amrita, and my brother-in-law, Awadesh, for their invaluable advice at important moments.

I thank Geeta and Srdjan (and Ila) for their unstinting support, whether I deserved it or not.

I would like to thank Tim Tregubov for his cheerful help in setting up many experiments when I was finding my feet.

Thanks are due to the innumerable friends in and around Hanover (especially Bhavnesh Kaushik) on whom I relied to provide relief during the long winters in Hanover.

This research is a project of the Institute for Security Technology Studies, supported under Award number NBCH2050002 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Science and Technology Directorate.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Wireless 802.11 LANs [3, 4] have become increasingly ubiquitous [33, 67]. As the access to these networks has steadily grown, dependence on them has also grown. Earlier, Airports, cafes and college campuses were examples of "hotspots" of 802.11 technology (also known as Wireless Fidelity – Wi-Fi) deployment. But as the density of deployment has grown (for example, some airlines have started deploying WIFI on airplanes [33]) the applications have become more compelling. Indeed, in highly dense Wi-Fi deployments, like urban city centers, Voice over Wireless LAN (WLAN) can actually be considered as a potential replacement for cell phones. Many businesses, universities and individuals are using them for critical applications. These applications often include VoIP [29, 31] (Voice over Internet Protocol), first responder apparatus [51], secure data transfer, inventory and shop-floor control systems, and video conferencing [52]. Key infrastructure is increasingly being deployed on wireless networks. Consequently the security and management of these wireless networks is increasingly important.

Often, the data being transferred on 802.11 networks is confidential. Access to this data may put the privacy of individuals, or the secrecy of sensitive corporate or military data, at risk. Cryptographic security [26] ensures the privacy of data through encryption. However, it does not guarantee protection from malicious attacks that aim to disrupt key services in the network and consequently the workings of the organization that the network

serves. For example, a Denial of Service (DOS) attack removes service from legitimate users of a wireless network. A Reduction of Quality (ROQ) attack allows service to continue but degrades the bandwidth, latency, or jitter in a way that negatively affects the user experience, and potentially provides some gain to the attacker. For example, some cheating attacks on the network will enable the attacker to gain unfair bandwidth on the network, and other attacks simply disrupt normal network service. An unscrupulous company may be motivated to run ROQ attacks on on a competing company's network during periods of critical use causing (possibly) monetary harm to the competition.

Because the wide commercial application of 802.11 wireless technology is relatively new, the attacks and the vulnerabilities are only now being understood. So are the countermeasures. New vulnerabilities to the network are discovered every week [8]. With the increasing proliferation of small Wi-Fi devices (like the iPhone) there are increasing points of failure (as each of these devices are susceptible to known 802.11 vulnerabilities and new ones introduced by bugs in those new devices) that are not well understood. Most enterprises that deploy 802.11 wireless networks rely on reactive diagnoses of network problems; they do not monitor the wireless network full-time. These are usually human-intermediated with both the reporting and the administering parties adding delay in the response to the problems. This turnaround time to respond to attacks or other such problems may be unacceptably long.

Many of the vulnerabilities are in the protocol. There are a large number of devices already deployed in the field. Even if, eventually, the vulnerabilities are understood and fixed, the implementation of the protocol is susceptible to bugs.

With a real-time monitoring system, however, the 802.11 frames may already have been captured, and if there was a malicious attack, the attack may be detected and reported and the attacker possibly identified right away. However, if the system waits for the client to complain, the attacker might have left the scene, or the problem might be undiagnosed for a long period. The offending frames are certainly no longer available. Ideally the problems need to be automatically diagnosed or even pre-empted before ever being encountered by humans. It is critical that these networks are constantly monitored and that problems are addressed rapidly with minimal dependency on people to report problems.

Rogue Access Point. If there is an Access Point (AP) that does not belong to the enterprise infrastructure but is plugged into the wired network of the enterprise, it is called a "Rogue AP" [16, 73, 68]. This AP represents an entry point into the enterprise network. If it is insecure, outsiders can gain unauthorized access to the network. Such an AP may not be intentionally malicious, as it may be placed there by an employee who wishes extend the network for ease of use, or to remedy a "hole" in the wireless coverage of the enterprise. The AP may, however, be put to malicious use by someone who wishes to enter the network. If the installer of the AP is malicious, the intent might be to gain access to clients of the network who might inadvertently connect to the AP. Determining the existence and locations of such rogue APs is much more difficult without any monitoring infrastructure. If the existence of such an AP is known, to locate the rogue AP so as to disable it would be a non-trivial task for an administrator without the help of such an infrastructure. The administrator would need to go to the location of the complaint and then measure the traffic around that area then attempt to triangulate. A pre-installed monitoring infrastructure could help localize the transmitter, reducing the effort on the part of the administrator.

There are some rogue AP detection techniques that can be enabled with such a preinstalled infrastructure [16, 73, 68].

Connectivity Problems. Users of an infrastructure wireless network may sometimes experience reduced quality of service, in the form of dropped sessions, slowness in browsing speeds, or unavailability of the network. In wired systems, the locations where this problem occurs are easily determined and monitored as they are localized to an Ethernet port on a switch or to the switch itself. In wireless systems, the locations are not so easily determined as the client may be geographically situated in any part of the AP's coverage area. Furthermore, the coverage area of the AP may shift from time to time. The following few paragraphs outline a few scenarios in which a client's connectivity may suffer.

Lack of coverage. The problem experienced by the client may be due to the area not being adequately covered. As these phenomena can be stochastic, they may not be easily observable once missed. A dedicated monitoring infrastructure, however, can observe and capture these phenomena *as they occur.* Any complaints that are later made can then be correlated with events observed by the monitoring system.

Malicious attacks. If the problem is caused by intentional malicious attacks, like a deauthentication or disassociation attack (there is no frame-level authentication of frames and therefore any frame, like a disassociation frame, can be spoofed), this reason may never be known if the wireless trace is not captured (as the attacker may have already left the scene). Unless every client is willing to record the transmissions it observes, the only option is to have a monitoring infrastructure with complete coverage of the wireless network. An example of an attack on the MAC protocol is a *disassociation* attack [27]. In this form of attack, an attacker spoofs the disassociation frames of a legitimate AP and

forwards them to a client associated to the AP. The client, thinking that the frames are genuine, disassociates from the AP. Later it tries to associate back to the AP. A flood of such spoofed disassociation frames has the effect of denying service to the client.

Interference. There might be sources of microwave radiation in the vicinity of the wireless network that interfere with the transmissions by the clients and the APs in the network (for example, from microwave ovens [41]). The interference may also be caused by misbehaving wireless cards (intentional or unintentional). This interference can, again, be observed reliably by the administrator only if there is a dedicated monitoring infrastructure. More so, if the interference is intermittent.

1.1 Monitoring Systems

The network may be monitored in several different ways. Packet "sniffers" (devices that capture frames or packets from either the wire, routers, switches, or the air. Also called Air Monitors (AMs)) may be installed on the wired network (or indeed, the AP controllers themselves may capture frames), as shown in Figures 1.1- 1.3, where wireless APs connect to the wired network. These sniffers will be able to observe much but not all of the wireless traffic and only at the IP or higher layers. The MAC-layer (802.11) headers will not be included. The management and control frames will be missed, as they are not forwarded past the AP. A system could monitor all APs using SNMP(Figure 1.2), and indeed this is a common approach. SNMP, however, only reports statistical information about the the current or past state of the AP. Some APs report major events with syslog records, but these typically only capture high-level information. Neither of these approaches supply

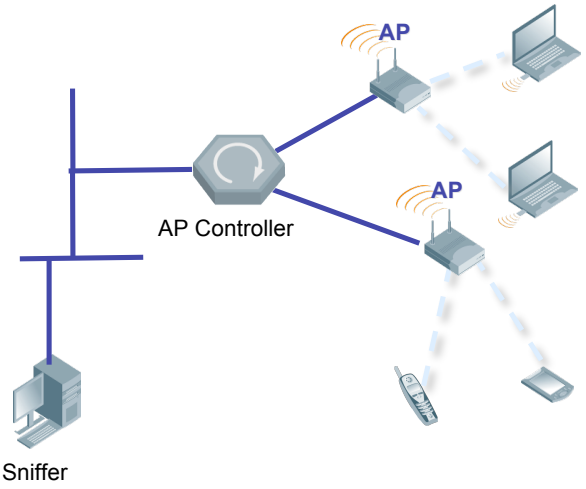


Figure 1.1: Packet Sniffers at on the wired network.

MAC layer information about traffic or information about ad-hoc traffic in the network. In addition, transmissions that are not a part of the infrastructure network will not be observed using these monitoring methods as they will not be forwarded past the AP. This deficiency is important because there are several known vulnerabilities in the 802.11 MAC protocol (e.g., its lack of frame-level authentication enables spoofing of MAC addresses), attacks on which cannot be detected without monitoring at the MAC layer. The APs themselves may monitor traffic for signs of attacks, but they will need to multitask between serving clients and monitoring. We believe that the solution is to use real-time frame capture from the air (Figure 1.4), using wireless sniffers, as it offers the capability to capture 802.11 wireless traffic at the MAC layer.

Wireless monitoring involves coverage of three dimensions — space, time and frequency spectrum. *Coverage, in the wireless monitoring context, is the ability to capture 802.11 frames at a particular location, and on a particular channel, for a given period*

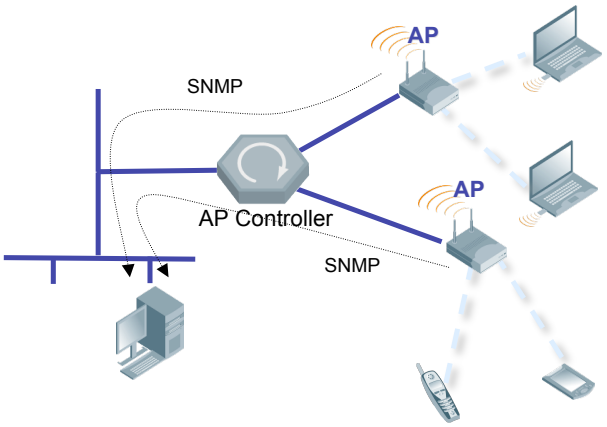


Figure 1.2: SNMP based monitoring.

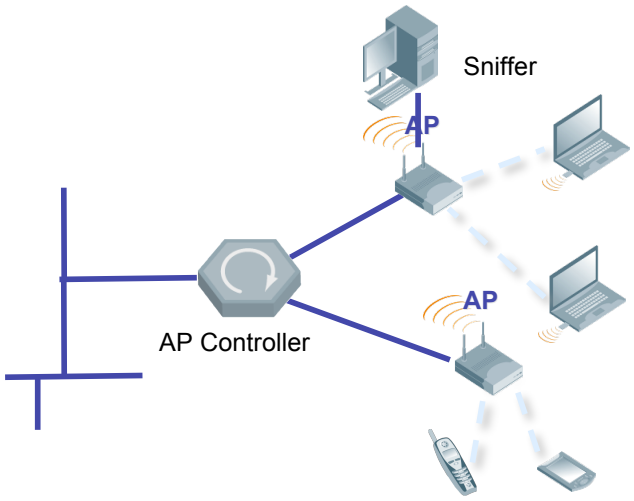


Figure 1.3: Wireless traffic captured on the wired port of an AP.

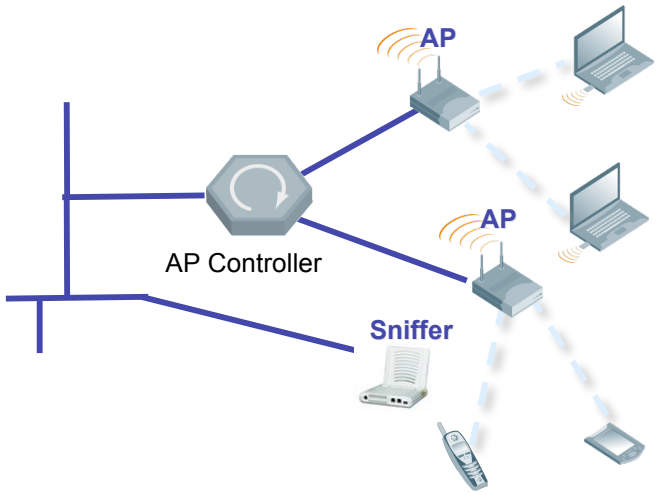


Figure 1.4: Wireless traffic captured off the air.

of time. It is important to maximize coverage so that any frames that carry attacks on the network can be captured by the monitoring system. "Holes" in the coverage may allow some such frames to slip by unnoticed. In our work, we attempt to maximize coverage over these three dimensions using the available resources. The limiting resources are the number of AMs, the number of radios on each AM, the computational and memory resources on the AMs and servers, and the bandwidth available for forwarding frames and other measured data. We developed this monitoring infrastructure as the "Measurement" portion of the MAP [24] (Measure, Analyze, Protect) project [62]. We place our AMs in the geographical area where the wireless network is located, sample the valid 802.11 spectrum and carefully merge streams of captured traffic from multiple AMs. Changing one aspect of the system (in terms of placement and sampling) affects the coverage in one or more of the three dimensions. For example, if an AM changes from channel A to channel B, there is no coverage of channel A at that particular geographical location until the

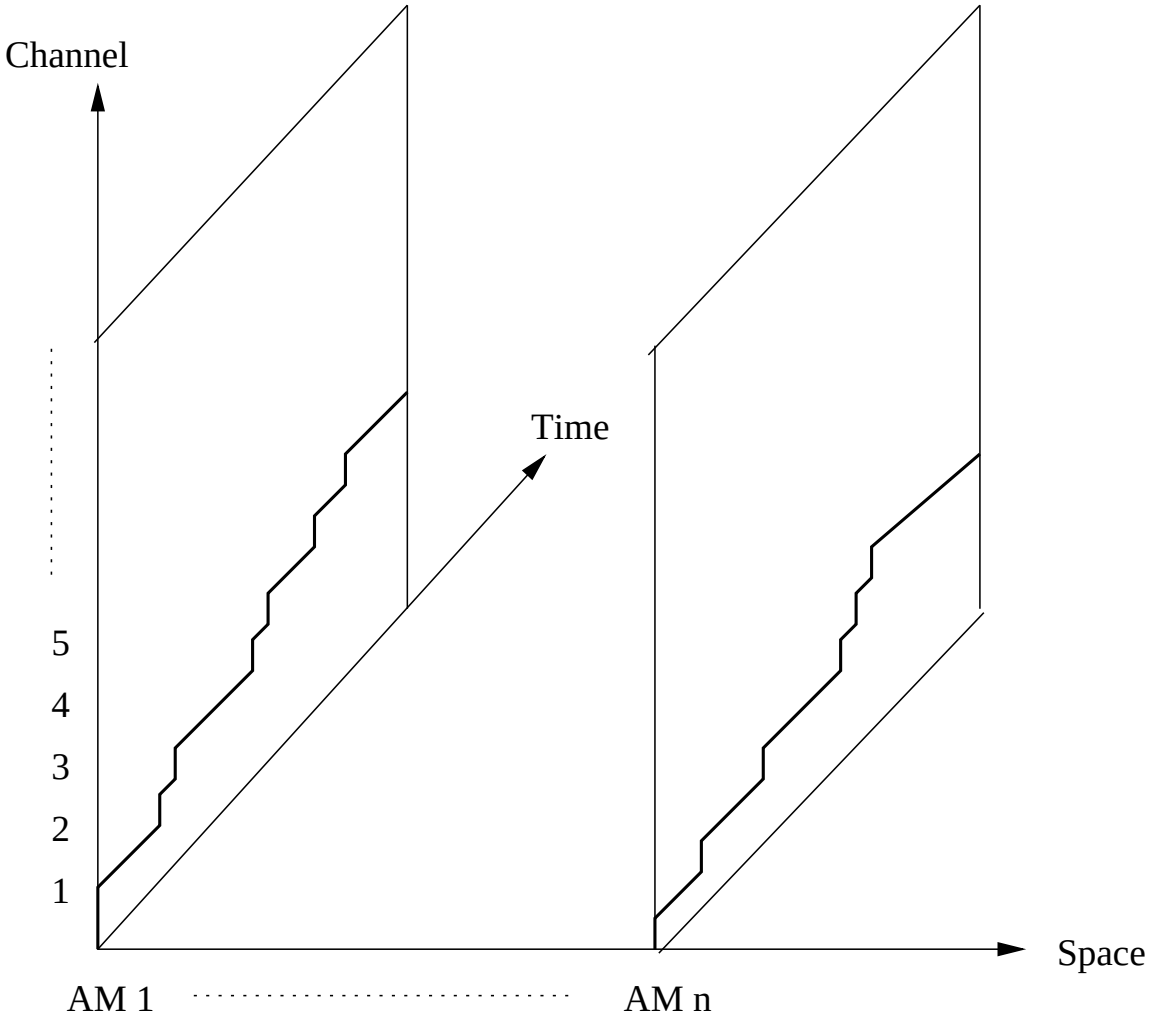


Figure 1.5: Dynamic multichannel monitoring

AM changes back to channel A. Over time, the measurement system monitors different subspaces within the space, time and frequency universe (Figure 1.5). Even with a large number of AMs, the coverage in the monitoring space is likely to be incomplete; capturing every 802.11 frame in the monitoring space is simply infeasible. There may always be channels that have occasional transmissions that cannot be observed by any AM near that location. Therefore, even highly provisioned monitoring systems must necessarily *sample* the available space of transmissions. Since global coverage (no holes) is not practical (and perhaps not even possible), we wish to maximize extent of coverage as much as possible with available resources and by using new sampling techniques.

Therefore, we developed new techniques for monitoring the wireless network effectively. To achieve this goal, we have incrementally approached the problem of sampling the monitoring area in the context of a fully functional system. Our sampling techniques are practical; we validate this claim by deploying this system and evaluating its efficacy in capturing traffic. Our contribution, in the end, is to provide a flexible sampling platform to capture traffic that adapts its behavior according the changes in the characteristics of the traffic, and that can be modified according to changes in the consuming application's focus. This platform is made efficient by coordinating the behavior of the AMs to reduce redundant capture. Finally, we provide metrics to compare different traces and apply these metrics in comparing sampled traces and the corresponding full capture traces (where a "full capture" trace is one which is captured by a sniffer that stays on one particular channel for the entire period of capture).

The following sections describe some background needed for understanding the problems in monitoring wireless networks. The next chapter describes related work we have

in the literature. Chapter 3 describes and evaluates the basic two sampling techniques of Equal and Proportional sampling. Chapter 4 describes Refocusing and the capture of traffic that is important to the consumers. Chapter 5 describes how to make the monitoring system more efficient using Coordinated Sampling. Chapter 6 evaluates the techniques described in Chapters 3–5 using two metrics that we believe have not been used in this context before. Chapter 7 presents a summarizes the dissertation and discusses future work.

1.2 Background

The transmitting nodes in wireless networks may be mobile. Also, the transmission area of wireless transmitters is not circular. As the propagation medium is a non-circular, possibly changing area, the physical location for monitoring the wireless network is not easily determined. Therefore the first challenge in monitoring wireless networks is the placement of the AMs. As the transmission range of the transmitters changes, the placement of the AMs is only approximate and the coverage is variable.

Also, there might be multiple transmissions (on different frequency channels) at a single location in space. Indeed the transmission loads on different channels are also variable. Therefore, we need to measure all channels and recognize the variable nature of the transmissions. Given the number of channels, as shown in Figure 1.6 (up to 14 channels in 802.11b/g, 25 more in 802.11a and all of these reused in 802.11n, except with different hardware), there are a large number of simultaneous transmissions possible in a single location.

1.2.1 Wired vs. Wireless

There has been a vast quantity of work in monitoring and measuring *wired* networks. Due to the spatially diverse nature of wireless network transmissions, the challenges involved in monitoring them differ with wired networks where the precise location for sniffing is the wire used in the network.

In wireless networks, it is not known beforehand where in *space* the transmissions can be observed. The traffic in the wireless network may originate at APs, clients associated to these APs, or even APs or clients that lie outside the control of the network administrator, but affect transmissions in the network. Two wireless nodes may have set up an ad-hoc network and may be transmitting with no AP in between. Clients may show up in areas where no transmission has ever been seen before, and once observed, they may change their location during transmission. They may change their associations (to APs), and in doing so may use a different channel. APs that are unknown to the network managers may appear in the network. We want to capture all the traffic between these entities irrespective of their dynamic nature. We use AMs to capture traffic.

As we do not know the location of most transmitting radios (we do in the case of APs), the propagation distance of the transmitted frames is not predictable. Even the coverage area of an AP (the area in which it can serve clients) might vary from time to time due to environmental factors. Due to this changing coverage, an AM might sometimes capture frames from an AP but later, might not. Therefore, the placement of AMs is a difficult challenge (compared to traffic capture in wireless networks).

We do not know precisely what area is covered by an AM. We can only estimate the capture areas by experimentation with some AMs and clients. Due to the uncertainty in the

capture area of the AMs and the transmission areas of APs, the capture areas of neighboring AMs must necessarily overlap to ensure spatial coverage. This overlap means that, on occasion, the same frames are captured at multiple AMs, and yet some frames are captured by only one AM. Figure 1.7 shows that AM1 will capture many frames of the AP but AM2 will capture only a few as it is on the border of the coverage of the AP. Similarly, AM2 will only capture many client frames but AM1 may capture only a few. In this situation, the sets of captured frames need to be merged to create a complete picture of the dialogue between the AP and the client; of course, the duplicates in the two streams also need to be removed.

1.2.2 Multi-channel monitoring

In wired networks, sampling is sometimes necessary in high-speed links simply because the measurement hardware may not able to keep up with the rate of frames being transferred. As the bandwidth available in wireless networks is substantially lower than that of wired networks, this challenge is not currently a concern (It may however be an issue in 802.11n).

In wireless networks, however, the challenge is different: there are multiple channels at the same geographical location that can be used to transfer 802.11 frames. We wish to monitor all of these channels simultaneously as there may be transmissions on any of them, even if the network infrastructure is using only one, or a few, of them. Therefore, even if the coverage of neighboring AMs overlaps, there are holes on *particular* channels at any given time. Therefore the problem of holes is not resolved by merely overlapping adjacent AM coverage areas. There are two possible solutions to this problem. The first solution is to attach multiple radios to one device so that each radio can monitor one channel. This way, each channel can be completely monitored. At the moment, this method

(or a method in which we place multiple single-radio devices at one location – which has similar drawbacks) is not feasible because the resulting hardware may be bulky and expensive. An example of such a multi-radio device is the "porcupine", developed at Indiana University [6]. Any solution that relies on increasing the total number of radios on the hardware is, in the end, not scalable. Also, even after provisioning as many radios as there are channels, there may still be situations in which all the transmissions are not captured. It is difficult to guarantee to capture *all* traffic that was transmitted within an area. Therefore, we believe that all 802.11 monitoring systems necessarily sample, i.e., they only capture a subset of all the traffic on the network.

A compromise solution is to monitor multiple channels with one radio, periodically changing the channel (Figure 1.5). We call this technique "channel sampling" as it results in collecting only a sample of the frames passing through all the channels. In its simplest form, channel sampling involves the radio moving sequentially through each channel in the wireless network, in a predetermined order, and spending equal amounts of time on each. We call the technique where each radio is autonomous in its pattern of channel sampling, *independent* sampling. If the radios sample the different channels available, there must be a provision to modify the sampling mechanism according to the needs (also called *focus*) of the monitoring system. It may be necessary to dynamically adapt (*refocus*) the system in order to capture more relevant traffic. Finally, if there is a controlling mechanism that coordinates the sampling on multiple radios externally, we call the sampling *coordinated*.

In this thesis, we make four contributions:

- We developed and studied the two basic techniques of *equal* and *proportional* sampling.

- We introduced the concept of *refocusing* and developed software for it that allows consumers of monitored traffic to request a *focus* that is most suitable to their needs.
- We introduced and implemented smart *coordination* among AMs so that the monitoring system can monitor the space of $\{\text{frequency} \times \text{space} \times \text{time}\}$ using as little hardware as possible.
- We used two metrics to compare 802.11 traces captured using various sampling techniques to traces captured using full capture on a single channel.

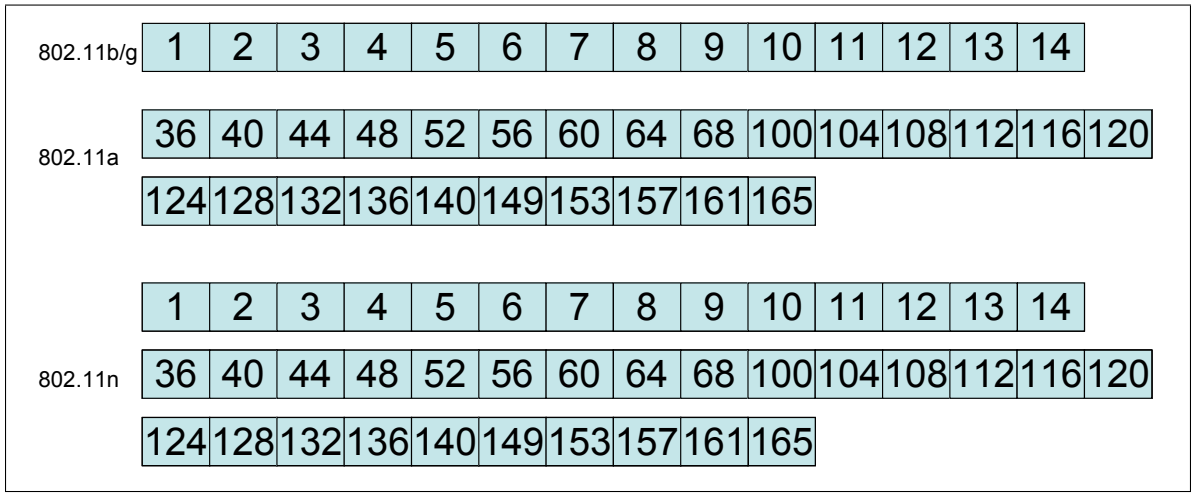


Figure 1.6: 802.11 channels

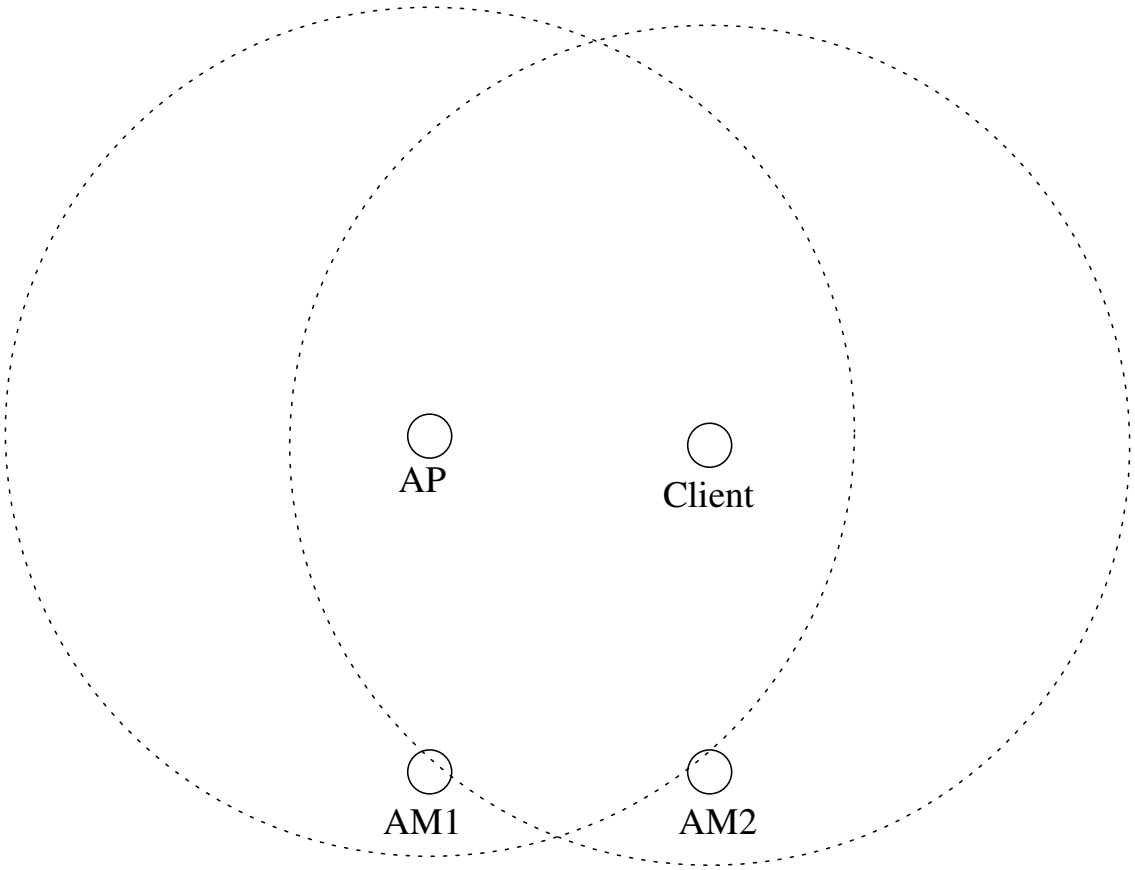


Figure 1.7: Overlapping coverage

CHAPTER 2

RELATED WORK

There has been work focused on various applications of wireless monitoring. There have been few studies about the mechanisms of wireless monitoring. The previous work has focused on the architecture of wireless monitoring and about merging of wireless traces from different sources. In our project (MAP) we have implemented a trace merger that we use in many of our experiments. We identify *channel sampling*, *coordinated sampling*, and *refocusing* as three primary challenges in 802.11 wireless monitoring. To the best of our knowledge, there has been no prior work in this exact area.

In the following sections, we describe previous work done that highlights the importance of wireless monitoring and measurement-based studies and analyses that could benefit from using our techniques.

2.0.3 802.11 flaws and security issues

Cam-Winget et al. discuss several security flaws in 802.11 network [18]. They describe authentication and encryption mechanisms used in 802.11i to alleviate the flaws in WEP (Wired Equivalent Privacy). Neither WEP nor 802.11i, however, are solutions to the inherent lack of authentication for MAC-layer management and control frames in the 802.11 protocol.

Bellardo and He [14, 34] discuss several 802.11 attacks. One method for detecting these attacks is to capture all frames and recognize particular attack signatures [32]. Clearly, all frames cannot be captured using any channel sampling mechanism. However, merging can be used to create a more complete picture that can be used for signature detection. Another mechanism is to detect the effect of the attack on the network (also known as anomaly detection), such as looking for clients that obtain unfair bandwidth allocation [47]. In a later chapter, we discuss *channel sampling strategies* that enable the AM to spend more time on channels that appear more important. We expect that using the appropriate sampling strategy is critical in anomaly detection.

2.0.4 Fault detection and analysis

Adya et al. [9] discuss a management and fault-detection infrastructure that depends on wireless monitoring by clients. They, however, assume the existence of some control over the clients in the network as they expect to deploy software on the clients. Our system (which we detail in later chapters) monitors the network without any direct control over or cooperation from the clients.

An early study of wireless LANs by Duchamp and Reynolds focused on the study of wireless networks in terms of throughput and error rates [25]. The measurement, however, did not involve sniffers and used the transmitter and receiver statistics for its results. Their system, though it *measures* wireless statistics, is not a *monitoring* system. Using a dedicated system with channel sampling strategies would enhance the study. MOJO is a physical layer anomaly detection system using wireless sniffers, but no channel sampling strategies were used to gather a sample from across all channels [63].

2.0.5 Wired-side sniffing

There have been many wireless network characterization studies that use wired-side methods like SNMP polling, syslog collection or wired-side sniffing to monitor wireless networks [37, 12, 66, 13, 36, 44, 23]. Others use wired side monitoring to predict client mobility in wireless networks [65, 64] and to build realistic mobility models from traces [42]. Henderson and Kotz provide good primer on measuring wireless LANs [35]. Mishra et alůse internal measurements from their wireless driver to generate the neighbor graphs that are needed in their roaming algorithm [54, 53]. Wired-side monitoring collects traces that are either aggregated summaries or incomplete. These studies can benefit from wireless sniffing as it will provide deeper knowledge of the traffic.

2.0.6 Wireless-side sniffing

Characterization studies using wireless sniffing are less common, and are typically concerned with only measuring the known channels on which APs are assigned [40, 60, 71]. Jardosh et al. [40, 39] deploy three sniffers, one for each known infrastructure channel. They simply co-locate the three sniffers and do not use any sophisticated placement strategy. The study focuses on congestion in networks. We focus on detecting attacks, and expect attackers to use any means available to them. Therefore, we use Channel Sampling to monitor the network so that we can detect anomalies or problems that exist on the channels on which APs are deployed as well as those that do not have APs deployed, but still may be used for attacks. Mishra et al˙ [54] use wireless sniffing on known channels for their empirical study on MAC-layer handoffs.

Portoles-Comeras et al. assess the performance of multi-radio sniffers as compared to multiple single-radio sniffers [56]. They extend their work in an evaluation of the efficacy of various off-the-shelf sniffers in capturing traces when deployed as multi-radio sniffers in various different permutations of vendor origin [57].

Chandra et al. demonstrate a cooperative client-based monitoring system for automated fault diagnosis using multi-layer information [20]. The clients themselves periodically capture traffic and share it with other clients through a peer-to-peer network for a cooperative diagnosis. Their system responds to requests by peers with specific multi-layer attributes, not full network traffic traces.

Bahl et al. implemented a monitoring system ("DAIR") using inexpensive USB based 802.11 radios deployed around a corporate environment on preexisting desktops that have wired network connections [10]. They do not, however, tackle the problem of channel sampling, nor do they use refocusing as a measurement technique. They only deploy sniffers on known infrastructure channels.

Another DAIR paper by Chandra et al. [19] provides a management system for wireless LANs. Their system, however, focuses solely on infrastructure channels and does not consider security applications, and does not attempt any channel sampling mechanism.

A few recent papers describe offline tools to capture and merge wireless frames from multiple AMs located around a building [22, 50, 72]. These papers concentrate on methods for synchronizing traces collected across multiple AMs into a single chronological trace, inferring missing frames, reconstructing transport-layer flows, and detecting performance artifacts and network inefficiencies. Most of these tools work only on offline traces and those that are online do not use channel sampling. Yeo et al. [70, 71, 69] describe an

infrastructure for wireless monitoring. The authors identify three challenges of wireless monitoring: capacity of each sniffer, sniffer placement and data collection. The authors perform several experiments using synthetic traffic and a multi-sniffer monitoring system. They describe an offline merging algorithm using beacons for time synchronization and heuristics for sniffer placement using signal strength.

Jigsaw [22] requires four radios per location, clearly an expensive solution. When few AMs are available, each radio must sample many channels, and our system of refocusing helps to gather the most relevant information with limited resources. In Jigsaw the authors place 39 monitoring "pods" around the building with four radios each. Each radio (AM) monitors a separate channel (infrastructure channels 1, 6, 11 and another channel). In their coverage experiments, their clients associate with APs and transfer data using `scp`. They report that their sniffers capture about 90% of all the `scp` frames sent to and from the clients. This experiment assumes that only traffic on the same channels as the APs that can be observed by *both* the AM *and* the client, or that can be observed by *both* the AM *and* the AP, needs to be monitored. There is no experiment in the paper that reports the results in the scenario where only the AP or the client is in the range of the transmitting radio but not the AM. (Or, is it the authors' claim that there cannot be any such transmitters? This claim is, of course, much harder to verify.) Due to the static allocation of channels to AMs, if there is an AM in range, it may be on a different channel. This case is important in a security scenario where an attacker may be hidden from the AM but may be in range of a client or AP. With the increasing numbers of channels available for transmission in 802.11 networks, simply increasing the number of radios in a "pod" cannot be the answer. It is clear, therefore, that channel sampling is the only practical technique

to cover a large monitoring area. The claim made in the second Jigsaw paper [21] that monitoring platforms from DAIR [10] and Jigsaw provide "the ability to observe every link-layer network transmission across location, frequency and time" is overly optimistic, to say the least.

Mahajan et al. [50] describe a tool "Wit" that includes an offline merging mechanism to combine multiple traces and an inference engine to determine missing frames. They use this tool to infer frames that were missed in the capture from SIGCOMM '04; this trace is also used in an earlier paper by Rodrig et al. [60] that analyzes the traces and presents statistics about airtime utilization, transmission rate adaptation and retransmission probability in those traces. In these papers, there is no attempt at dynamic channel sampling.

The popular "war driving" tool, Kismet [5], includes configuration options for channel-hopping sequences and channel-dwell times (channel intervals) but it does not possess the ability to dynamically modify its behavior based on current channel conditions. Only static schedules are supported.

2.0.7 What's new?

The bulk of my work is about the relatively unstudied challenges of channel sampling, refocusing and coordinated sampling. Most prior work approaches the problem of monitoring by making assumptions about the traffic. The assumption is generally that there is only one channel or a small fixed number of channels to monitor. We approach the problem of monitoring from the other direction. We assume that there may be a large, unknown number of channels to be monitored and that we have far fewer radios than are sufficient to capture traffic on all the channels at all locations. We believe that such an approach is supe-

rior because the conditions may change and sometimes too rapidly for human intervention. We are unaware of any work that looks at channel-sampling or channel-hopping strategies, refocusing and coordinated sampling.

CHAPTER 3

INDEPENDENT SAMPLING

An ideal wireless monitoring system would capture every transmitted frame on every channel at every location in its deployment area. As a matter of fact, the guidelines of the NSA for 802.11 Wireless Intrusion Detection Systems (Wireless IDS) recommend that every "over the air" 802.11 frame be captured and analyzed [2]. Certainly some monitoring systems attempt to reach this goal by using dense radio deployments. It is, however, trivially obvious that every frame cannot be captured at every location and every channel. Therefore, any monitoring system must necessarily sample the dimensions of $channel \times space \times time$. Using the available sniffing hardware most efficiently to capture the maximum number of frames is a goal of our system.

To capture as much relevant traffic as possible, we first try to maximize the total number of unique frames captured by arranging for AMs to spend most of their time on the busiest channels observed. At the same time, we believe that AMs also need to spend a minimum time on the quieter channels to capture a sample of the traffic. This is one type of channel sampling *strategy*. This strategy attempts to maximize capture of traffic on the network.

Any sampling strategy in which each AM individually makes the decision of changing channels is an *independent* strategy. If decisions to change channel are made by looking at information from multiple AMs, we call that strategy *coordinated*. Some of the strategies are adaptive in nature, so that more time is spent on channels that are considered more "im-

portant" in some respect (e.g., channels with more traffic are more important than channels with less traffic) even if the actual channel that is "important" changes with time.

A *sampling cycle* is one pass through every channel being monitored. We consider two strategies that visit all available channels, in order, once per sampling cycle. Each sampling strategy is, therefore, defined in terms of the time spent on each channel, and how that time is adjusted in response to current conditions.

In the first strategy, the AM subdivides every sampling cycle into equal intervals, one for each channel being sampled. We call this strategy the "Static/Equal" sampling strategy.

In the other strategy, the AM counts the number of frames per second that are observed on each channel, extracts the proportion of load (in terms of frames per second) on each channel, and uses that number to compute proportion of the next sampling cycle to spend on each channel. We call this strategy the "Frames/Proportional" strategy. The input is "Frames/sec" observed on each channel, we compute the "Proportion" for each channel, and we use this information to affect the time we spend on each channel [1]. We avoid spending zero time on any channel by ensuring a minimum time is spent on every channel. We pick this time to be greater than $2\times$ the time taken to transmit the largest possible 802.11 frames at the slowest rate (50 ms).

Similarly, we can easily imagine a Clients/Proportional strategy (in which AMs count the number of client stations seen) or the BSSIDs/Proportional strategy (in which the AMs count the number of BSSIDs (Basic Service Set Identifiers) seen on each channel). Given

[1] The net effect of the strategy is that the number of frames captured is increased. Clearly, the strategy that maximises the frame capture while using a single radio is one that is always on the channel with the highest frame-rate. As we cannot know instantaneous channel frame rates, and we depend on historical information, we end up "hedging our bets" among the channels according to the observed frame-rates in the last cycle.

any counter with an observed value on each channel, in each cycle the time spent on each channel is in proportion to the observed values from the preceding cycle.

We can conceive of various strategies to satisfy different needs of the monitoring system. For example, we may deem that a channel with more clients is more "important." In that case, a Clients/Proportional strategy is appropriate as it would spend more time on the more "important" channels. In this way, we can implement various strategies where each strategy represents a possible "focus" of a consumer of the traffic.

We conducted experiments using the Frames/Proportional and the Static/Equal schemes. Henceforth we call these the *Proportional* and *Equal* strategies respectively.

3.1 Mobility and Sampling

Mobility can affect the ability of a monitoring system to capture samples of traffic. As clients move through the monitoring area, the performance of the sampling techniques may worsen.

Speed. Clients moving at different speeds can affect the sampling in different ways or each to a different extent. A fast moving client (e.g., moving at the speed of a car) may move through the coverage area of an AM in a few seconds. Moving at the speed of 100 kilometers per hour (kph), a car would pass through the diameter of the coverage area of an AM, approximately (a conservatively high) 100 meters, in 3.6 seconds. If this duration that a client in a moving car spends in the coverage area is less than the cycle time being used in sampling, the proportional strategy may not be able to react in time to spend more time on the channel being used by the client. If the cycle time was one that we use in

our experiments (2.2 seconds) the proportional strategy may or may not be able to see a higher number of frames on the channel being used by the client, depending on whether the channel was early or later in the second cycle. We can see, therefore, that unless very small cycle times are used, 802.11 networks may be unable to detect the source of "hit-and-run" types of attacks. To make matters worse, in most cases, coverage area of AMs in urban environments will be much less then 100 meters. (In the order of 25 meters.) Therefore, the opportunity to dynamically adjust the channel sampling time would be even less.

At lower speeds, where a mobile client takes several seconds to move through the coverage area of an AM, a proportional sampling strategy can have sufficient time to react to the increase in traffic. For example, a bicyclist moving at a speed of 20 kph would take 14.4 seconds to pass through the coverage area. If the proportional strategy were using our cycle time of 2.2 seconds, it would be multiple cycles before the client is outside the coverage area, enabling the strategy to adapt several times over.

Remember that, however, changing channels takes a non-zero amount of time. The shorter the cycle lengths, the greater the number of times that channel switches are done. If channel switching time was zero, then shorter cycle times would be preferable. As it is not (it is in the order of tens of milliseconds), it is better to strive for a low number of channel changes. Therefore, a balance needs to be maintained between cycle time and channel switching time.

3.2 Implementation of the Proportional Strategy

Let n be the number of 802.11 channels to be monitored, and T be the time for one complete sampling cycle. Let $f_{i,j}$ be the frame count for channel i in the cycle j and $I_{i,j}$ be the time spent (the *interval*) on channel i in cycle j.

On initializing the sniffer, there are no observations on which to calculate our frame count metric. We therefore begin with an Equal cycle, where the radio spends $I_{i,1} = T/n$ time on each channel. At the end of this first cycle, $\{f_{i,1}\}$ contains the number of frames captured on each channel in the first cycle.

We cannot allow any channel's interval to become too small to capture a frame, because then $f_{i,j} = 0$ and for all future cycles, the interval for channel i will be zero. In our implementation we therefore limit the minimum interval size M.

The time to be spent on channel i in the next cycle $j + 1$ is based on the proportions observed during cycle j:

$$I_{i,j+1} = \frac{(f_{i,j}/I_{i,j})}{\sum_{i=1}^{n}(f_{i,j}/I_{i,j})} \times (T - (M \times n)) + M \tag{3.1}$$

Equation (3.1) calculates the proportion of frames per unit time that is captured on channel i in the cycle j. This proportion is multiplied by the total cycle time T less the minimum time spent on every channel so that the interval $I_{i,j+1}$ spent on channel i in the next cycle is in the same proportion as the observed frame rate [2].

[2] nearly the same proportion, as we first allot the minimum interval size M and then the remaining time

The above equation implements the Frames/Proportional strategy. If the channel with the highest frame rate were the most important, this strategy would be well suited to capture important traffic.

If, however, the most important channel is the one with the highest number of clients, the above equation would change. The $f_{i,j}$ in the above equation would be replaced by $c_{i,j}$ which is the number of clients observed in the cycle j on channel i. This change would result in implementation of the Clients/Proportional strategy. Similarly, other strategies can be easily implemented according to the definition of what is "important'.

3.3 Dingo: a controllable sniffer

We developed a set of software components, named *dingo*,[3] that collectively enable a variety of frame sampling policies to be defined and controlled, and their effects monitored. dingo comprises two main components: *amsniffer*, which runs on each AM device, and *amcontroller*, which runs on a more powerful central Linux server. dingo also employs an additional software component, a *merger* developed as part of earlier work, and described below. Figure 3.1 shows the principal components of this software and the communication paths between them.

The role of the *merger* is to receive the streams of frames captured by the AMs and to merge these into a chronologically consistent order, with any duplicates removed, to enable analysis of the traffic. The information in each consolidated frame is retained for a period of one second during which the merger anticipates the arrival of duplicate frames from other AMs. Duplicates are counted and discarded, and a record of the receiving AMs

[3] A dingo is an Australian native dog renowned for its ability to track prey in bleak conditions.

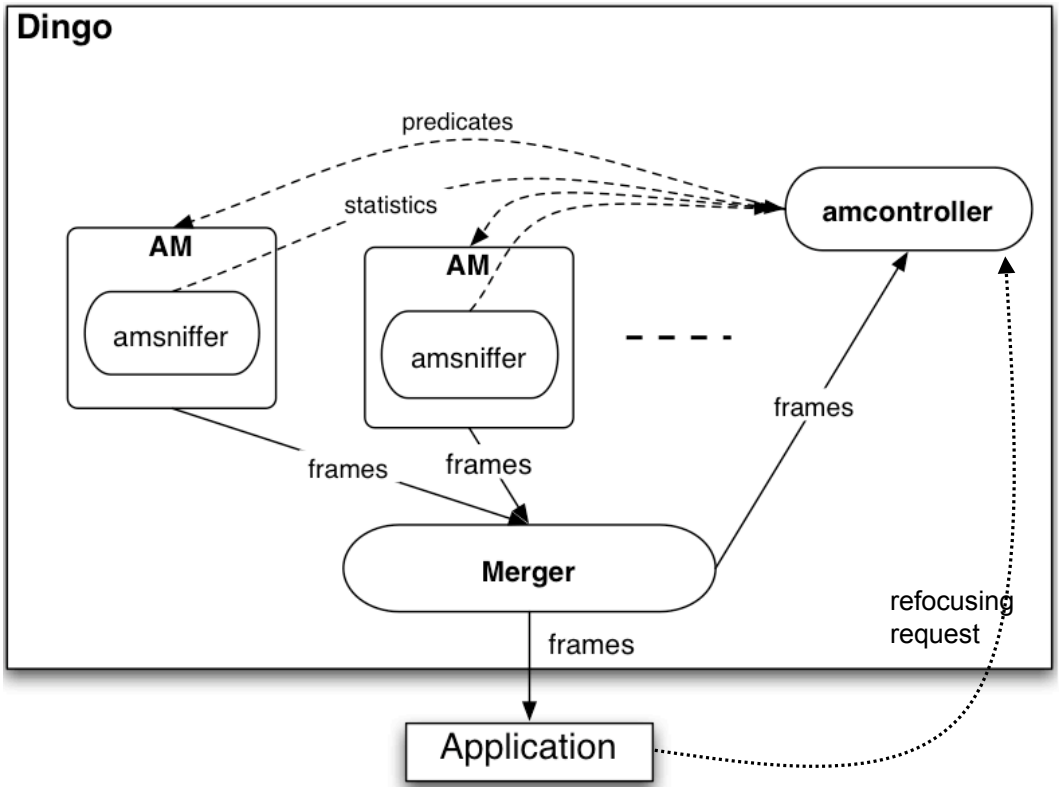


Figure 3.1: The dingo Architecture

and their frequencies are collated and forwarded as a UDP/IP frame stream to any number of subscribers, such as our amcontroller. The details of the merger are available in the Appendix.

The *amsniffer* component runs on each AM; multi-radio AMs can run an instance for each radio. (In our experiments we found that it is more effective to invoke two instances of amsniffer, each listening on a different interface, than it is for a single process to monitor two interfaces in an interleaved manner.) Command-line options to amsniffer indicate which wireless interface should be employed, the default sniffing policy to be followed, and the destination for captured frames.

The amsniffer captures features from each frame header and transmits it over a wired Ethernet infrastructure to the merger using UDP/IP. The role of the merger is to interleave the AMs' streams of frames into a chronologically consistent ordering, and to remove frames captured in duplicate by multiple AMs. For duplicates, the output record includes a list of the receiving AMs and signal strength. The merger's output is forwarded to subscribing applications and to our amcontroller.

The role of the amcontroller is to determine *scheduling policies* and to disseminate them to the AMs. Policies specify a sequence of channel numbers, and the duration for which the interface should listen on each channel. A typical scheduling *cycle* will involve visiting each channel, collecting a variety of statistics about the traffic observed on each channel. Each instance of amsniffer executes its current scheduling policy for a requested number of cycles or until directed by amcontroller to execute a new policy, either an existing pre-stored policy or one computed by the amcontroller. We found that our devices can experience a significant delay when changing from one channel to another, and that

this delay is minimized by visiting requested channels in ascending order (approximately 30ms when ascending, 300ms when descending), so we limit all schedules to this order, descending only once at the end of the cycle.

Notice that our approach does not require specific policies to be "hard-wired" into amsniffer. Each amsniffer may receive a distinct scheduling policy, perhaps determined from the type and extent of traffic recently sampled by that amsniffer and its neighbors, or to consistently monitor traffic in a particular geographical region. The ability to remotely program the AMs provides the greatest opportunity to experiment with new sampling strategies.

In the experiments described in the subsequent section, we used an older version of dingo, called basset. Basset was not as well-instrumented as Dingo and was missing the amcontroller component. It ran independently on each sniffer without any interaction with other components other than to forward captured frames to the merger.

3.4 Testbed

Our testbed consisted of two Intel x86 sniffers with Atheros-based wireless cards. The sniffers ran Linux (Fedora Core 4 with kernel 2.6.14 and the MadWiFi driver) for our experiments. The two sniffers were placed 90 cm apart in a research lab that had a crowded radio environment with several 802.11 experiments in progress at any given time. The production network in the building used channels 1, 4, 8, and 11.

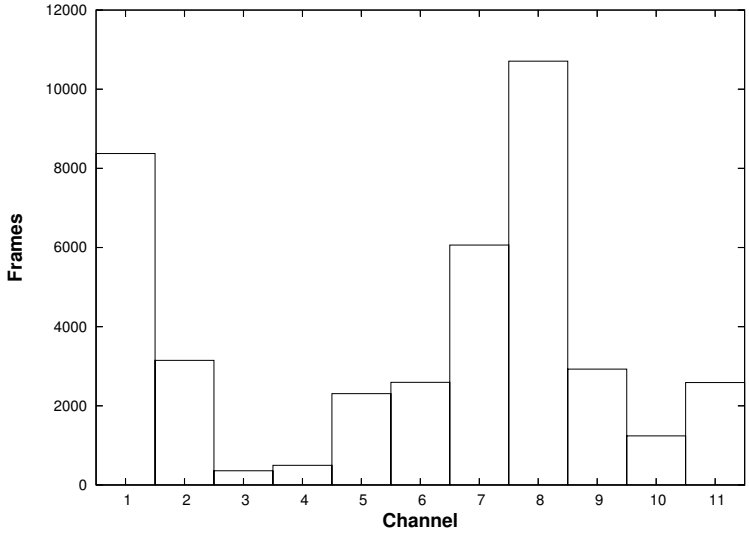
3.5 Experiment

We sniffed the 11 legal 802.11b channels using the Equal sampling strategy on one sniffer and the Proportional Count sampling strategy on the other sniffer. We experimented with various cycle times $T = \{1.1 \text{ sec}, 2.2 \text{ sec}, \ldots, 22 \text{ sec}\}$. Each of these experiments was run for 10 minutes. (The whole process was then repeated, swapping the strategy assignment to sniffer to discount for radio-propagation differences. We merely summed the results for each of these two configurations as the result was only volume-based.)

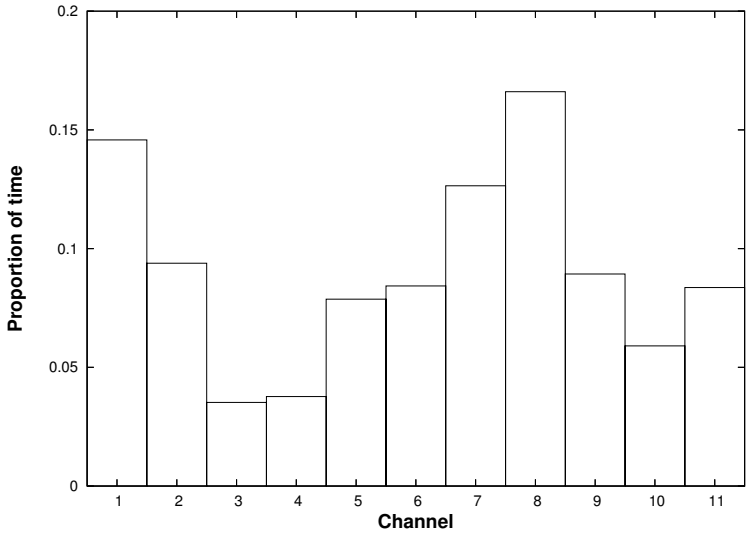
3.5.1 Results

We first consider the results of a 10-minute experiment with T=11sec, examining the volume of frame capture and the time spent on each channel.

Figures 3.2(a), 3.3(a), 3.4(b) and 3.4(a) show the volumes of traffic collected on each channel using different strategies and time periods and Figures 3.2(b) and 3.3(b) show the time spent on each channel over that 10 minute period using the two strategies. In Figure 3.2(a) we can see that using the Proportional Strategy results in a capture that is skewed towards the most busy channels (1,7,8) resulting in a greater number of frames being captured by the sniffer than when the sniffer uses the Equal strategy (we believe that the channel 7 appeared busy because of transmissions bleeding from channel 8). Figures 3.2(b) and 3.3(b) show that the time spent on the channels was proportional to the observed frame rates on each channel (Proportional) and equal (Equal) as expected. We can see that the channels with maximum capture volume were indeed the busier channels by looking at Figure 3.3(a).

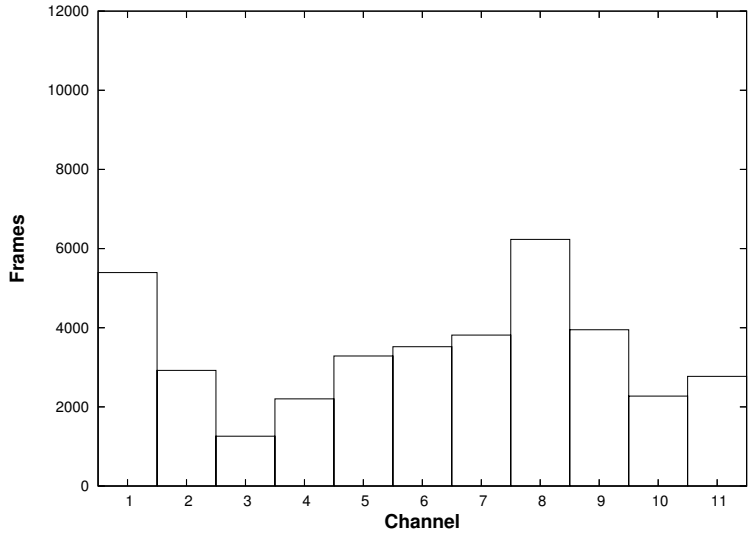


(a) Proportional strategy ($T = 11$ sec).

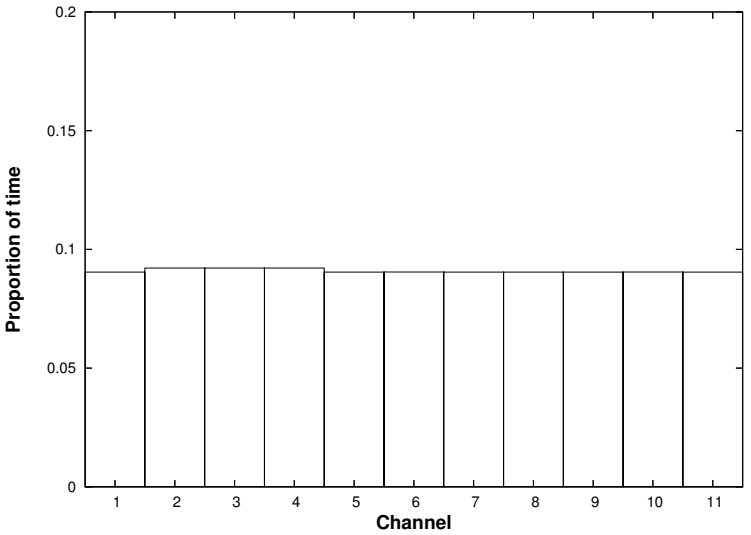


(b) For Proportional strategy, time spent on each channel is proportional to traffic observed ($T = 11$ sec).

Figure 3.2: Frame volume and time spent on each channel using the Proportional strategy.

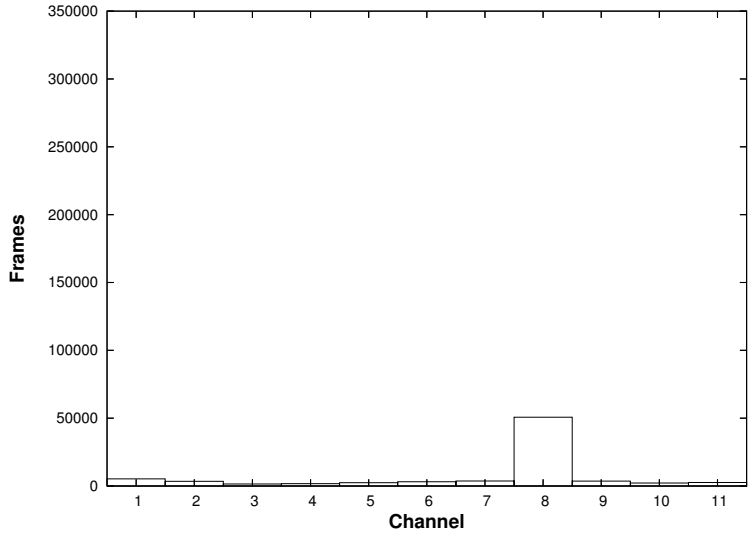


(a) Equal strategy ($T = 11$ sec).

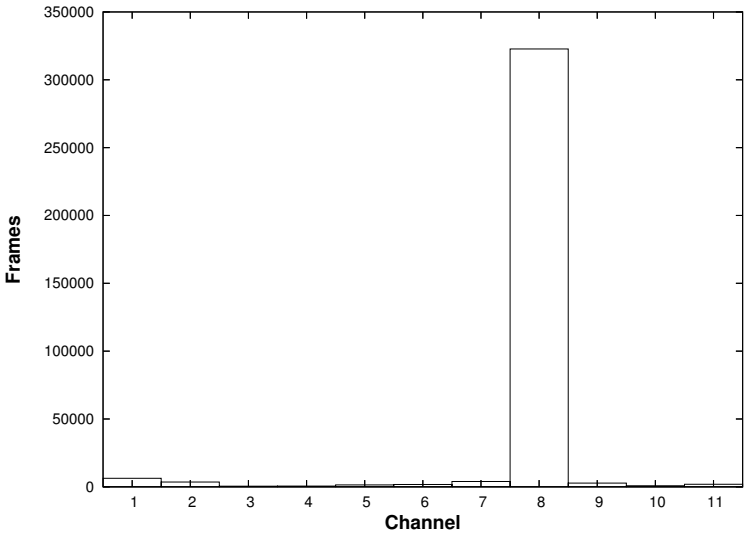


(b) Confirming equal time spent on each channel ($T = 11$ sec).

Figure 3.3: Frame volume and time spent on each channel using Equal strategy.



(a) Equal strategy ($T = 3.3$ sec).



(b) Proportional strategy ($T = 3.3$ sec).

Figure 3.4: Frame volume observed on each channel in Proportional strategy and Equal strategy.

In a separate experiment with a smaller cycle time (T=3.3sec), there was, by chance (Figure 3.4(a)), a huge spike on channel 8. On further investigation we determined that the data was HTTP traffic from one client. Using the Equal sampling strategy, about 19 megabytes of traffic was captured in the run that lasted 10 minutes—about twice the volume as in other runs. Simultaneously, however, the sniffer running the Proportional sampling strategy (Figure 3.4(b)) captured nearly 80 megabytes of data. This volume was approximately 8 times the volume collected by other runs of the Proportional strategy. This result indicates that the Proportional sampling strategy is indeed successful in increasing the data capture to match the observed load. The Proportional sampling strategy captured six times as many frames on channel 8 than the Equal strategy in this particular case. All the other cases with cycle times T=1.1, ... , 22sec had results very similar to those shown in Figures 3.2(a)–3.3(b).

3.5.2 IDS Experiment

One use of sampling is for detecting security breaches in wireless networks. In the case of denial-of-service attacks, a malicious attacker may be motivated to disrupt as much traffic as possible. To achieve this goal, the attacker will need to monitor the channels to determine which channel has the highest load. Once this channel is determined, the attack can be launched on that channel.

To determine the effectiveness of our system in such a threat scenario, we mimicked this attack strategy. We periodically measured the traffic on each channel and launched an attack on the channel with the highest number of frames. Each attack was of random

duration (between 0 and 12 seconds) with intervals of random length (between 5 and 12 seconds) between each attack.

We used a Linux laptop as the attacker's machine, running the Auditor distribution of Linux. We ran the deauthentication attack *file2air*, which spoofs the MAC address of an AP and sends flood of deauthentication messages to a victim so as to deny service. We configured *file2air* to send a flood of deauthentication frames with interframe intervals of 1 millisecond.

We sampled the 11 802.11b/g channels using the Proportional the Equal strategies. To detect the attack, we ran the popular IDS tool *snort-wireless* [7] on the traces captured using the two sampling strategies.

We observed that snort-wireless detected a greater number of abnormal sequence number gaps (which indicate MAC spoofing) in the traces collected from the Proportional (Figure 3.5) strategy than the Equal sampling strategy. Snort-wireless also generated more alerts (Figure 3.6) in the Proportional trace. This indicates that the Proportional Count strategy captured more attack instances than in the Equal sampling strategy. A paired t-test indicates that the two strategies perform differently at the 1% level. Figures 3.5 and 3.6 show the number of sequence number gaps and alerts flagged, with total cycle times varying from 1.1 seconds to 3.1 seconds, and a minimum interval time of 0.5 seconds in each run. Although there is no trend over the increasing cycle times, snort-wireless consistently detects a greater number of abnormal sequence number gaps and flags more alerts in the Proportional Count sampling strategy than the Equal strategy. Our expectation that more attacks would be detected using smart sampling was therefore correct.

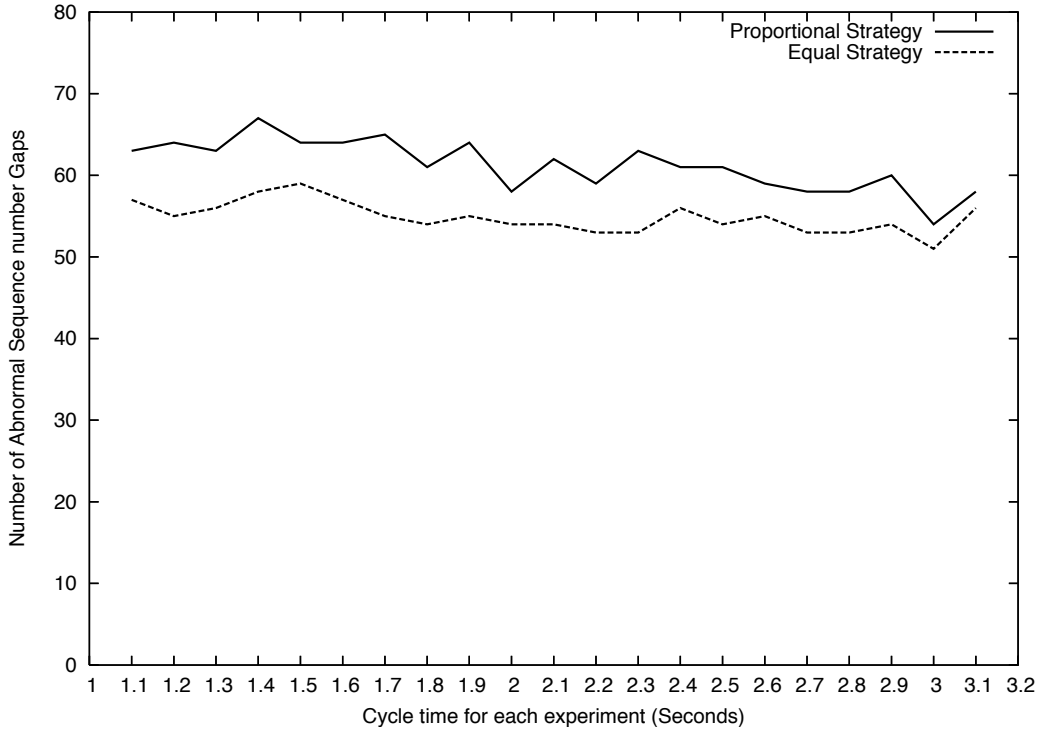


Figure 3.5: Number of abnormal sequence number gaps detected by snort-wireless. snort-wireless detects more abnormal sequence number gaps in the trace captured by the Proportional Count sampling strategy than the Equal strategy.

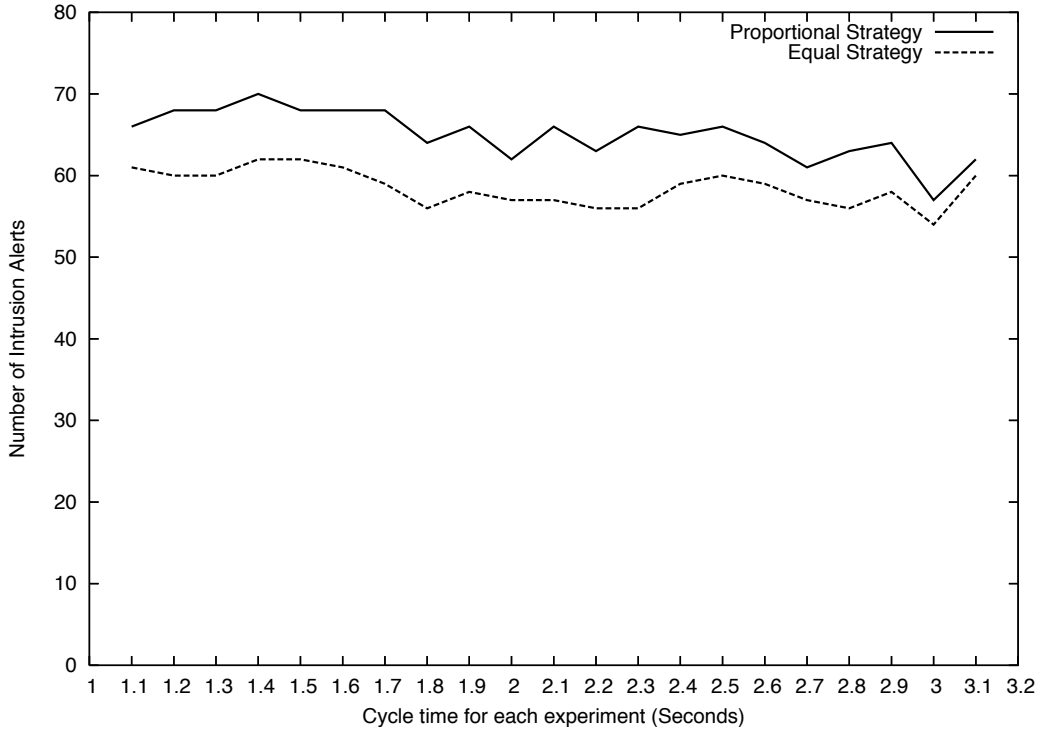


Figure 3.6: Number of Alerts flagged by snort-wireless. snort-wireless consistently flags more alerts in the trace captured using the Proportional Count sampling strategy than the Equal strategy.

3.6 Deployment and Performance of Sniffing

We deployed 20 Aruba AP70 AMs among and between production 802.11a/b/g Aruba APs, as shown in Figure 3.7. The AP70 has a 266MHz MIPS IDT32434 CPU, 32MB DRAM, two Atheros AR5212 802.11a/b/g NICs (network interface controllers), and two Ethernet NICs. We installed OpenWRT Linux (Kamikaze branch, r5494) and Madwifi (v0.9.2) on each, and a copy of *dingo*, our channel sampling software that sniffs through libpcap (v0.9.5). Dingo can sniff on both of the NICs, but in the experiments it sniffed only one radio and only 802.11b/g, as our 802.11a network is in limited use. We connected all the AMs to the merger through our wired building network, which is switched 100Mbps Ethernet, without routers in the middle.

Since frames (not bytes) are the unit of capture and analysis, we[4] focus on frame-related performance metrics. In our experiments we computed, for every 60-second interval, the workload of each sniffer as *frames per second received by the NIC* (FPS) [5]. the frame *drop rate*, and the *CPU load rate*, defined as follows.

$$\begin{aligned} \text{NIC-received FPS} &= \frac{\text{\# frames received by NIC in one minute}}{\text{60 seconds}}, \\ \text{drop rate} &= 1 - \frac{\text{\# frames captured by sniffer}}{\text{\# frames received by NIC}}, \\ \text{CPU load rate} &= \frac{\text{virtual CPU time in seconds}}{\text{60 seconds}}. \end{aligned}$$

[4] I am grateful to Yong Sheng and Keren Tan for running these performance evaluation experiments.

[5] We count the number of frames received by the NIC by retrieving statistics of the Atheros HAL provided by the *Madiwifi* driver.

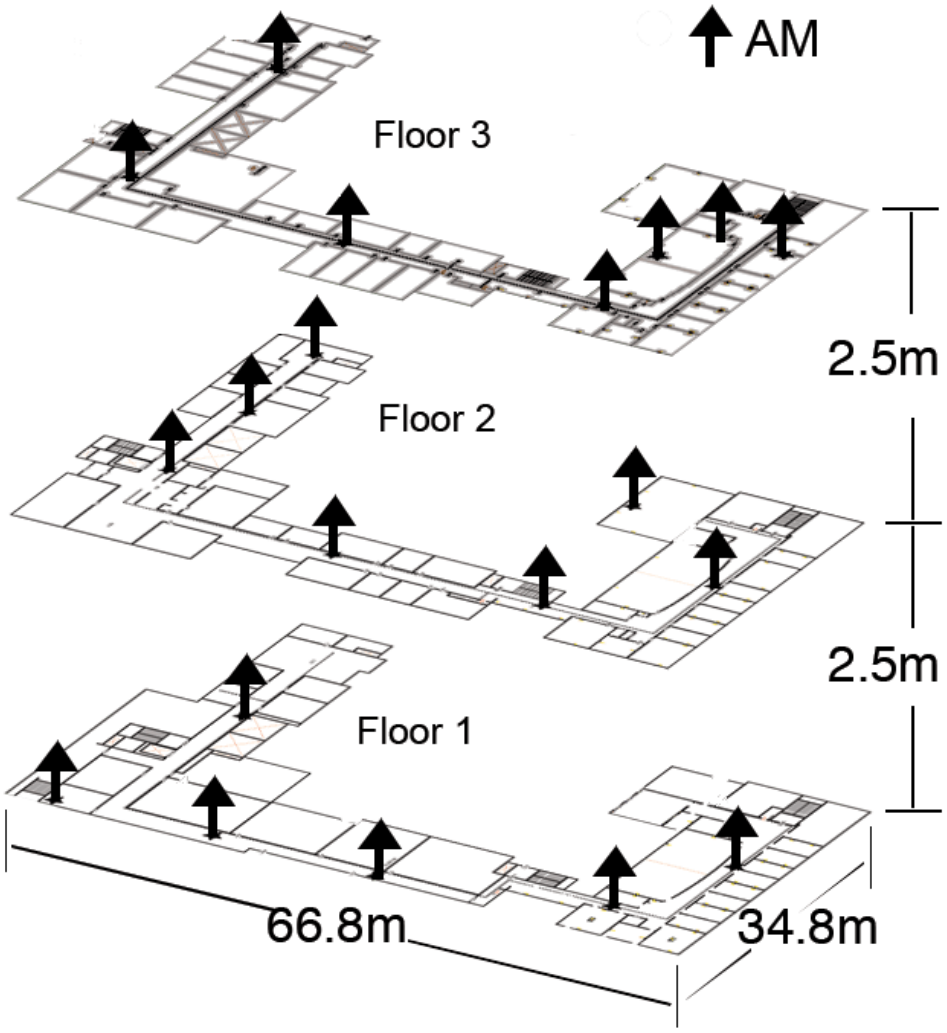


Figure 3.7: Testbed deployment in our CS department building. The 19 Aruba AP70 APs (not shown) provide 802.11a/b/g service to over 80 faculty, students and staff members in about 1,600 square meters of usable space. We deployed 20 Aruba AP70 AMs (arrows) throughout.

The 60-second granularity is appropriate for our purposes, since our day-long, building-wide experiments provide a large enough population of samples, while remaining moderately sensitive to bursty traffic.

We ran three experiments, each with 20 AMs configured as follows.

1. *Normal-nosampling*: locked on channel 11, the busiest one in our building;
2. *Normal-equal*: Equal strategy, sampling channels 1 to 11 over a cycle of 5.5 seconds,
3. *Normal-proportional*: Proportional strategy, sampling channels 1 to 11 over a cycle of 5.5 seconds.

Each experiment lasted 24 hours, during the period Monday–Wednesday November 27–29, 2006. Experiments (b) and (c) both included an AM that was fixed on channel 11 (the busiest channel).

During the 24 hours of Monday, November 27, 2006 (experiment *Normal-proportional*), our 20 AMs (each covering approximately 80 m^2) captured 317.1 million frames, which were merged to 161.1 million frames (50.82%). Our system observed 98 distinct access points (specifically, BSSIDs) and 696 distinct clients (STAs) in or close to our CS department building.[6] The pre-merger trace consumed 37.8 GB, while the post-merger trace consumed 23.4 GB.[7] The statistics logged from all MAP components totaled about 1 GB.

[6] Although our building has 19 production APs, our sniffers could hear some APs from neighboring buildings, and many research APs. Also, each production AP advertised multiple BSSIDs

[7] All traces collected are pcap format including all AMEX features, and overhead of pcap, Ethernet, IP and UDP headers in each AMEX packet. To first approximation, it is the amount of bandwidth consumed by MAP.

3.6.1 Without Channel Sampling

We plotted the mean value (with 20-80% error bars) of frame drop rate and CPU load rate against the NIC-received FPS in 3.8, over data collected in experiment *Normal-nosampling*. We also plotted the CDF (cumulative distribution function) of NIC-received FPS; in more than 90% of cases (one-minute intervals over the three-day experiment for all AMs) our sniffer dropped fewer than 20% of frames. Overall, AMs captured 86.9% of total frames received by NICs.

Due to the limited CPU processing power, the AMs dropped many more frames during heavy traffic bursts. Most of these frames were dropped by *pcap*, since its internal buffer overflowed. We should be able to reduce the drop rate by tuning the AMEX feature-extraction code, by extracting fewer features, and by configuring *pcap* and the Madwifi driver with larger buffers. However, the linear relationship between CPU load and workload size (when the NIC received fewer than 1000 frames per second) suggests that some dropped frames are inevitable.

This result suggests that running any computation-intensive analysis software on AMs will degrade sniffing performance badly. For similar reasons, the APs themselves would be inadequate for sniffing or detecting attacks.

3.6.2 With Channel Sampling

To evaluate the impact of channel sampling on the performance of sniffing, we compared the *Normal-nosampling* experiment with those of the *Normal-equal* and *Normal-proportional* experiments.

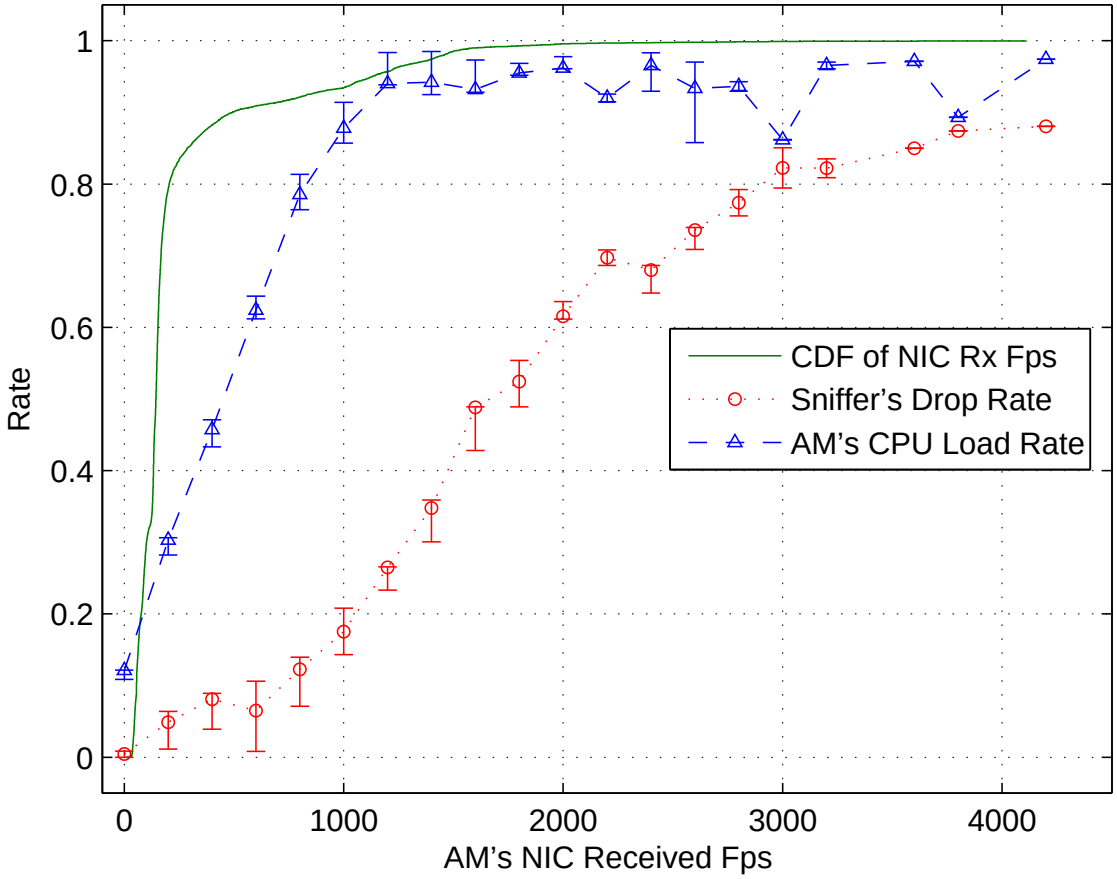


Figure 3.8: Performance of sniffing, no channel sampling.

According to Figure 3.9, at the same level of NIC-received FPS, the frame drop rate with channel sampling was significantly higher than staying on one channel. These drops were due in large part to the five-millisecond overhead delay that Atheros AR5212 chipset typically needs to switch frequency and resynchronize [59], during which time the sniffer is "deaf" and unable to capture frames. At a given NIC-received FPS, the Proportional strategy dropped fewer frames than the Equal strategy, because it spent more time on busy channels and was thus able to capture more frames between hops.

Figure 3.10 shows that the Proportional strategy allowed the NIC to receive even more frames did the AM fixed on the busiest channel, because it adaptively switched channels in response to bursty traffic.

3.11 examines the long-term frame rate, *per AM per hour*. We use this data to focus on the different drop rates; the drop rate is 100% minus the capture rate marked here. Averaged over the whole day, the Proportional strategy had a slightly higher drop rate than the Equal strategy (44% and 40% dropped, respectively), because the proportional policy brought more frames to the NIC and pushed the system to work at a higher FPS— at which level the AM tends to drop more frames. In terms of total frames captured per hour, however, the Proportional strategy captured more than double the number of frames than the Equal strategy, while still covering all channels.

In summary, frequent channel hopping had a noticeably negative effect on capture effectiveness. On the other hand, the Proportional strategy captured far more frames overall. Therefore the impact of frame loss due to channel hopping is balanced by the greater capture using the Proportional strategy.

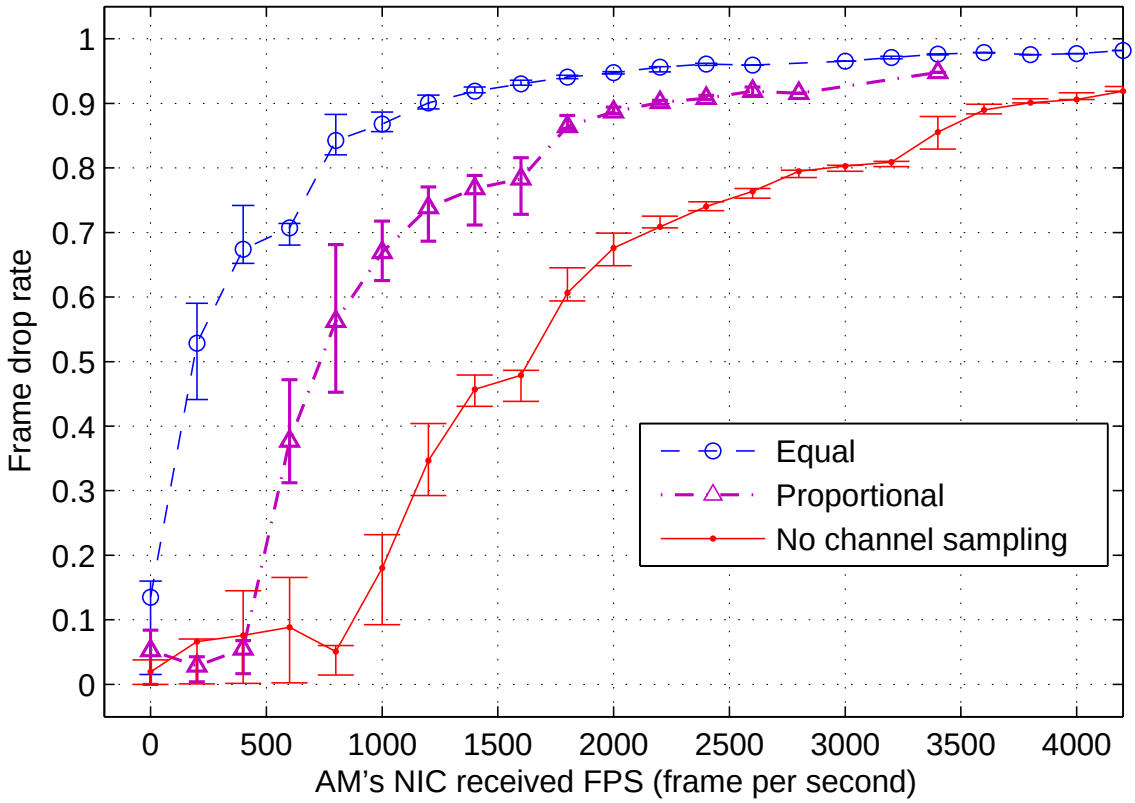


Figure 3.9: Frame drop rate of each strategy across 1 minute intervals through the day.

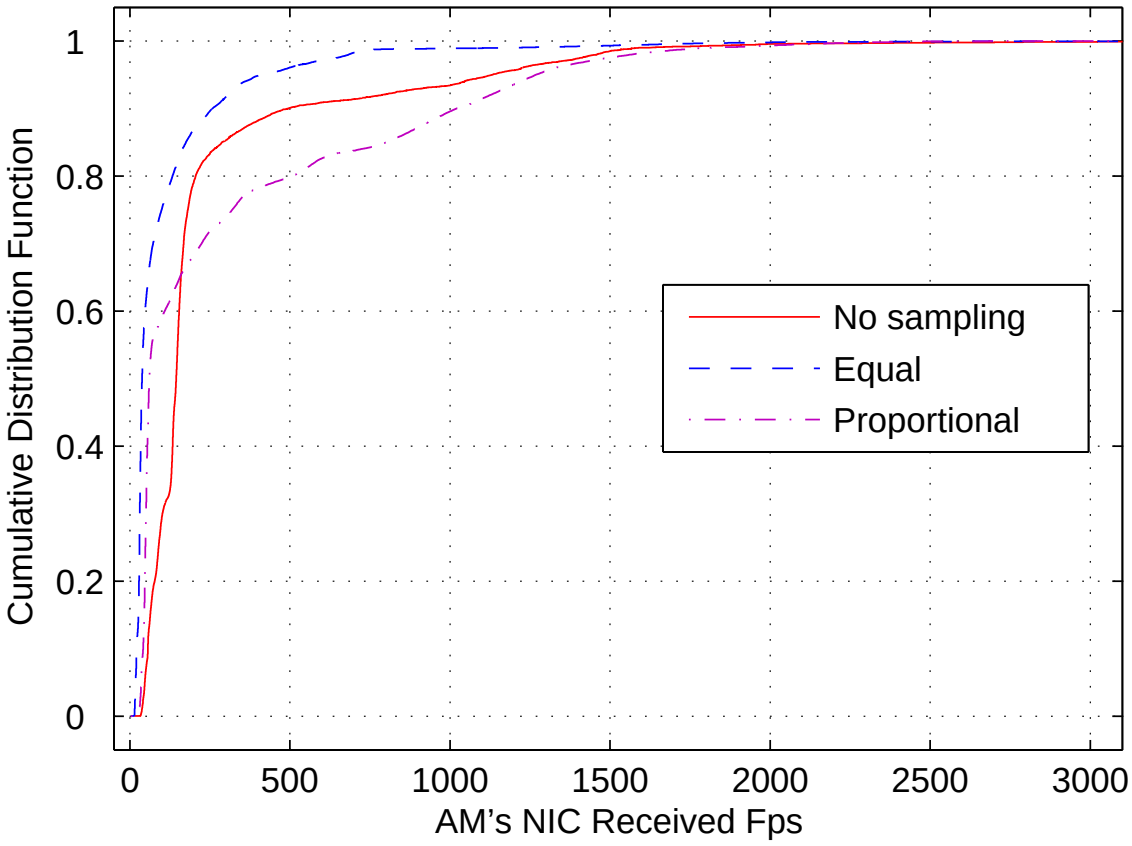


Figure 3.10: CDF of NIC-received FPS for each strategy across 1 minute intervals through the day.

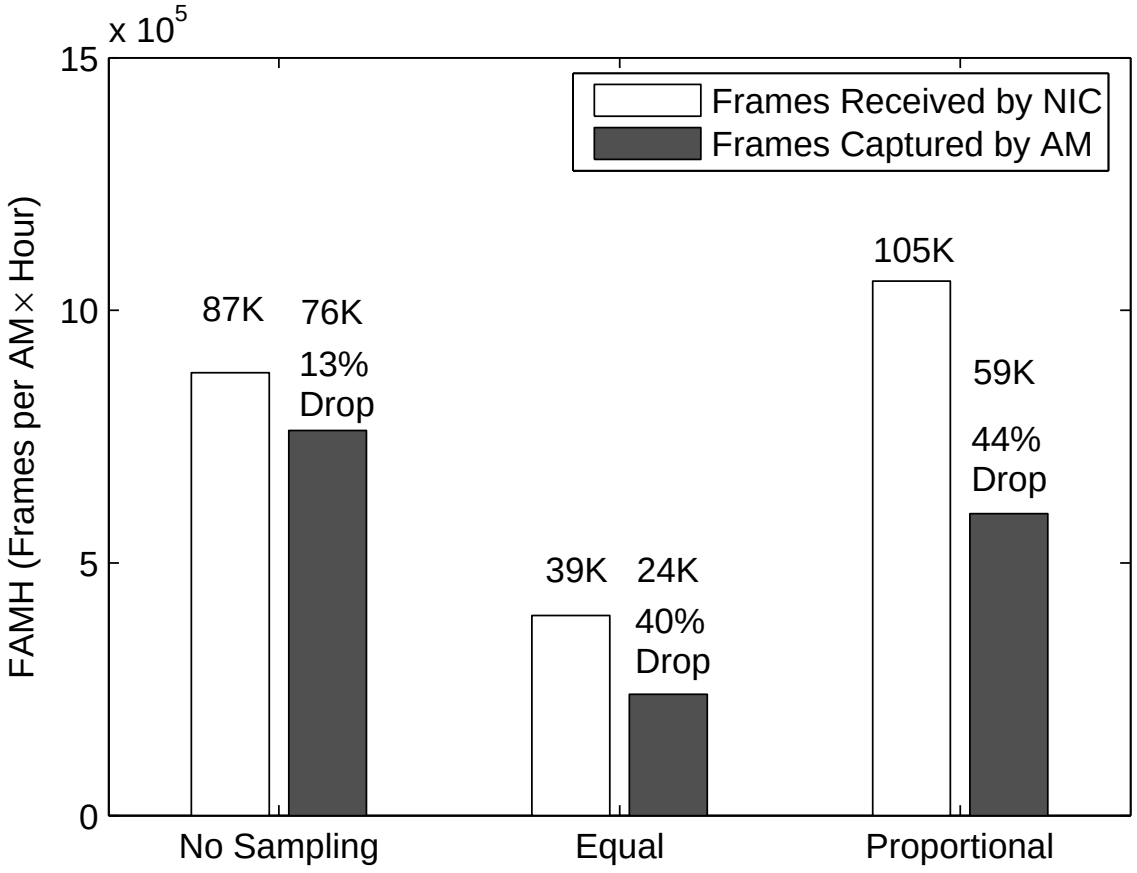


Figure 3.11: Long-term performance.

3.7 Conclusions

In this chapter we motivate the need for smart channel-sampling strategies. We experimented with the Equal and Proportional sampling strategies and demonstrated that the Proportional strategy was successful in capturing more traffic and was also more useful in capturing some types of attacks. We demonstrated the deployment of our software across the computer science department with 20 sniffers and live traffic.

The strategies discussed in this chapter adapt to the changing character of the wireless traffic. However, these strategies cannot be changed according to the needs of the consumer of the traffic. In the subsequent chapters, we explore more general and flexible sampling strategies and we show how to make sampling more efficient.

CHAPTER 4

REFOCUSING

Monitoring systems capture traffic in wireless networks so that it can be forwarded to downstream components that analyze this traffic. Different components have different goals. For example, a wireless intrusion-detection system has a goal of detecting ongoing attacks or intrusions on the wireless network. Channel sampling enables the monitoring system to capture a subset of the frames being transmitted in the air. Our dynamic sampling strategies capture traffic by spending time on the channels according to the *importance* of the channels. This *importance* metric is defined statically in our strategies. As the relative importance of the channels changes, the strategy will capture more traffic from the appropriate channels. If, however, the monitoring system has multiple downstream components, there can be multiple definitions of what is important and the definition may change from time to time. A wireless-network administrator may desire real-time information to the administrator about the health of the network. A rogue-AP detector will attempt to scan the airwaves for traffic from unknown APs. As the different components have different goals, the nature of the traffic that they wish to see varies (i.e., each one has a different *focus*). The administrative utility may need a wide perspective of the network. The IDS may wish to frequently modify the type of traffic it processes, i.e., *refocus* so that it can identify different types of intrusions possibly being caused by different attackers.

If all the traffic on all channels at all locations is captured, this change of focus is merely a filter in the collection pipeline of a trace in the case of channel sampling; however, the complete traffic may not be available in a trace. Therefore, to bridge the gap between full capture and channel sampling with fixed strategies, our monitoring infrastructure needs to be able to change focus quickly according to the needs of downstream components (consumers of the captured traffic).

Refocusing is the act of directing the monitoring system to pay more attention to certain types of traffic than others. (Alternatively, it can be defined as the dynamic modification of the channel-sampling strategy according to the needs of downstream applications.)

In particular, there are many definitions of "important". The channel with more traffic is not necessarily the most important channel. It may be the channel with the most traffic from a *particular* client, the channel with more traffic with particular PHY layer characteristics, or the channel with a large number of new clients.

For example, consider a channel that wants the monitoring system to spend more time capturing traffic from MAC address aa:bb:cc:dd:ee:ff and destined to MAC address 11:bb:33:dd:55:ff. In this case, the controller needs to direct the AMs to spend more time capturing traffic on the channels that observe these MAC addresses. The channels that carry more frames of this type are more important than those channels that carry fewer frames of this type. In this chapter I describe our approach for enabling changing definitions of importance.

In the previous chapter this concept is partly represented by the few different strategies: the Frames/Proportional strategy gives greater importance to channel with higher frame rates, whereas the Clients/Proportional strategy gives more importance to channels with a

greater number of clients. These strategies are static and predefined, however; we wish to support the creation and installation of new strategies on the fly. We provide functionality to address several different goals simultaneously in addition to quickly changing the focus to suit the varying goals of the downstream components.

4.1 Integration with Dingo

We introduce dingo, our controllable sniffer, in the previous chapter. Here we describe dingo's facility that enables quick refocusing.

While sampling traffic, each *amsniffer* maintains a small number of counters, including the number of frames captured on each channel, the total length of those frames, and the number of frames matching one or more Boolean *predicates* provided by the *amcontroller*. At the end of each cycle, each *amsniffer* sends its counters to the *amcontroller* for consideration in future scheduling decisions. The range of policies described in the earlier chapter are based on these simple counts gathered at the AMs. For example, a policy employing Proportional sampling spends time on each channel proportional to the recently observed frame rate on that channel.

The dingo predicates are written in a small language, similar to C's expressions. The language supports all precedence levels, equality and relational operators, and data types including integer, Boolean, string, and MAC address. About 30 keywords in the language correspond to the attributes of each captured frame and the wireless environment in which it was captured. Our predicates provide access to the 802.11 header attributes and a few PHY-layer attributes, and are analogous to the expressions supported by the popular `tcpdump`

utility and Berkeley packet filter. For example, predicates may determine whether a captured frame was a control, management, or data frame, may examine the source, destination, and BSSID MAC addresses of frames, examine a frame's length, payload length, the channel on which it arrived, or its relative signal strength.

To support refocusing, dingo's *amcontroller* uses the predicate counters in a modified form of proportional sampling, scheduling each *amsniffer* to spend time on each channel in proportion to the number of frames matching the predicate. In this manner, *amsniffers* focus on the traffic of interest, while still devoting a small amount of time on other channels to determine if the traffic pattern is observed there. For example, the predicate "src == 00:16:cb:b7:18:82" could be used to focus on traffic from a stolen laptop's wireless interface. Any *amsniffers* capturing frames matching this predicate will be instructed by the *amcontroller* to devote more sampling time to the channels recently carrying that traffic. AMs not capturing traffic from this laptop will continue to follow a default sampling policy. If the laptop associates with a different access point using another channel, or moves within range of different AMs, the shorter time spent on other channels will facilitate those AMs to focus on the laptop. A short cycle time, typically 1 or 2 seconds, enables each *amsniffer* to quickly identify and focus on required traffic patterns. Again, this ability to remotely program the AMs with a wide variety of predicates facilitates experimentation. A few examples of the predicates are shown in Table 4.1.

Table 4.1: Examples of predicates that can be used in dingo for matching against frames.

Description	Predicate
Is the source address in the frame "aa:bb:cc:dd:ee:ff"?	$src ==$aa:bb:cc:dd:ee:ff
Is length of the frame > 100?	$len > 100$
Is the frame a beacon?	$isbeacon$
Is the frame a deauthentication frame?	$isdeauth$
Is the frame a disassociation frame OR is the destination MAC address equal 11:aa:22:bb:33:cc?	$isdisassoc || dst ==$11:aa:22:bb:33:cc
Match everything (Proportional sampling)	$true$
Match nothing (Equal sampling)	$false$

4.2 Applications of refocusing

We believe that refocusing has many applications in wireless research, security, and network management. Any application that requires more than cursory scanning of the traffic in the wireless medium will sometimes desire an increased focus on some subset of the traffic, and yet other applications will simultaneously need a baseline broad sampling. We consider three classes of application.

Localization. If a Wi-Fi device needs to be geographically localized, the refocusing system can focus more attention on it by capturing more frames to and from it. Refocusing may aid in better localizing the laptop, by capturing more frames from as many different perspectives (AMs) as possible. We can capture more samples in less time, increasing the accuracy or reducing latency for estimating the location of the laptop using any of the state-of-the-art methods. We describe one such experiment in Section 4.3.2.

VOIP-quality measurement. Consider an enterprise network manager who wishes to monitor the quality of Voice-over-IP calls. If there are known VoIP clients using the Wi-Fi network, we can focus on those MAC addresses and thus monitor the relevant channels, more closely. Alternately, we could focus on channels with observed VoIP activity (by recognizing the use of particular protocols) or through a higher-level metric like the jitter, per-frame delay in the VoIP calls, or the observed congestion in a channel. For example, the predicate may take the form "jitter > 30 ms". Such high-level predicates cannot yet be matched in dingo. This capability is part of our future work.

Security monitoring. For example, we can refocus on channels that carry an excessive number of deauthentication messages, or on MAC addresses that are known to have been recently spoofed. In the future, using our techniques, we can focus on channels where new

clients appear, then study their packets to discern whether they seem especially vulnerable to attack. The system can fingerprint new clients to determine if they are employing drivers, cards, or operating systems with known vulnerabilities [17]. If indeed they are vulnerable, we can refocus our sampling to more closely monitor them.

Network administrator tasks. Table 4.2 extracted from the Department of Defense's "Draft Joint Wireless Administrator Checklist" [1] contains a few examples of tasks that a wireless-network administrator performs. Many of these tasks may benefit from a wireless monitoring system with refocusing. Several of these tasks require the network administrator to locate the wireless device. In the next section, we describe our experiments that demonstrate that localization is more accurate with refocusing than it is without refocusing.

4.3 Results

We set out to investigate whether refocusing can be a valuable tool in wireless measurement systems. We seek to demonstrate the potential value of this approach by applying it to some likely scenarios.

4.3.1 Improved volume of capture

In our CS department, we had deployed 20 Aruba AP70 AMs throughout the three floors of the building as shown in Figure 4.1 [1]. The building also has 19 802.11a/b/g access points. In this experiment, we only used one of the two wireless NICs on each AM.

[1] Unfortunately, in the course of these experiments, one of them was not working. Our results are therefore based on these 19 AMs.

Table 4.2: Network Administrator Tasks

Description of Task	Does it benefit from monitoring?	Does it benefit from refocusing?
Discover and physically locate rogue devices	✓	✓
Identify and research failed access attempts	✓	✓
Identify and research communication problems	✓	✓
Track performance and activity on the wireless network	✓	✓
Track the production AP software and patch or update as appropriate	✗	✗
Track performance of encryption/ authentication devices (RADIUS)	✗	✗
Verify that devices comply with wireless policy	maybe	maybe
Verify that devices are not vulnerable to known firmware/bugs	✓	✓
Use mobile device to identify and document signal coverage of wireless network devices	✓	✓
Use mobile device to identify and document residential/commercial wireless devices that are visible during site surveys	✓	✓

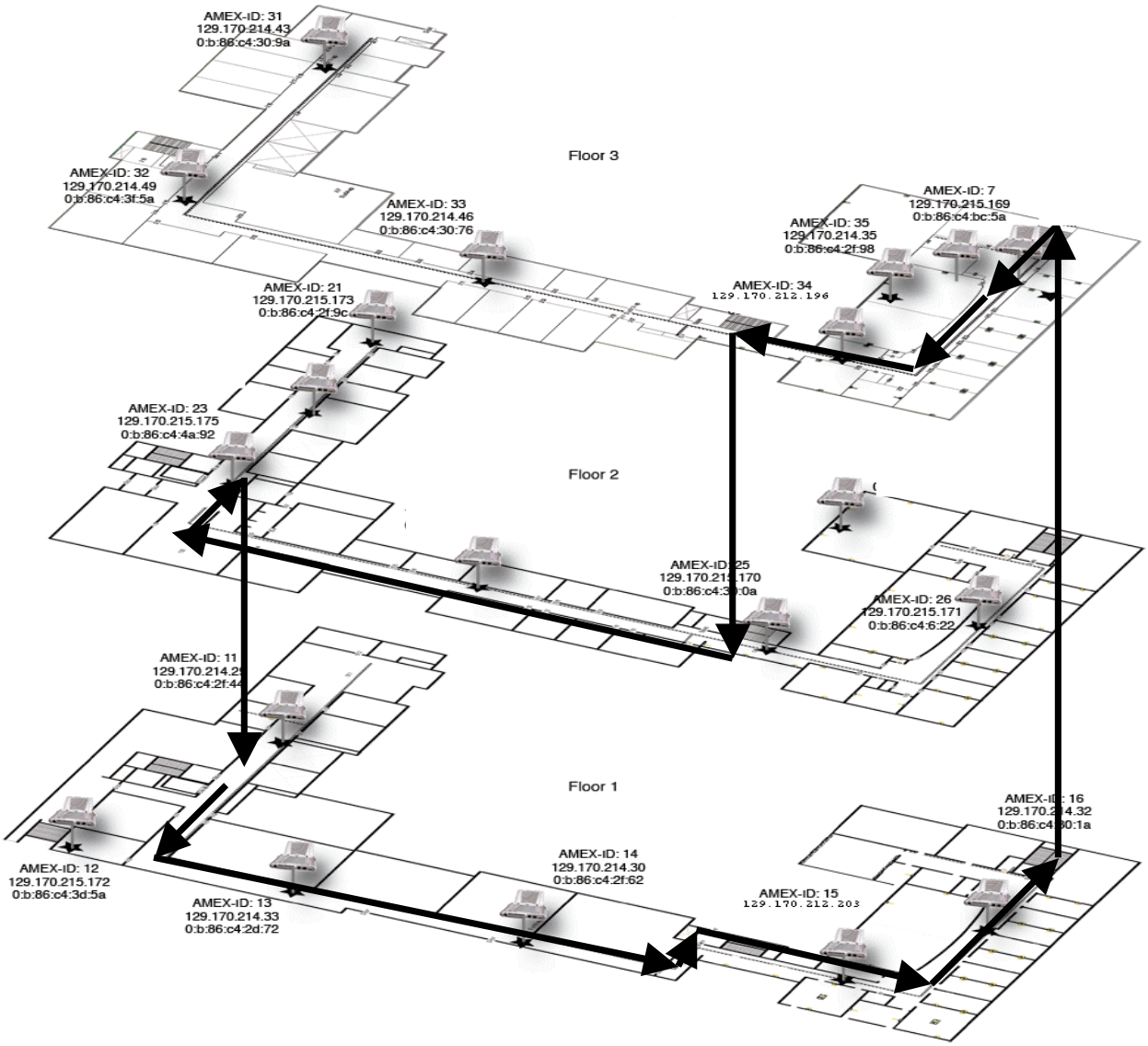


Figure 4.1: Path taken in the refocusing experiment

We performed two experiments in which a laptop transmitted 10 UDP frames per second to the non-existent MAC address 22:22:22:22:22:22 on a channel randomly selected from the 11 802.11b channels. The laptop changed channels every 10 seconds. (The reason that we changed channel every 10 seconds was because we wanted the the refocusing to adjust the proportions on each channel fairly frequently and therefore test the refocusing system more thoroughly. It would have taken far many more seconds for the Laptop to reassociate to different APs in the network had we transmitted to a real destination MAC address.) In each experiment, the laptop was carried around the fixed path (shown in Figure 4.1) in the CS department building for a period of 10 minutes. As we were only walking (1–2 meters per sec) the refocusing strategy on an AM went through several cycles while the transmitting laptop was in its coverage area.

In the first experiment our AMs used the traditional equal-time sampling strategy in which the AMs spend equal time on all the channels. In the second experiment, we refocused the AMs to spend more time on the channels that were observed to capture more frames from the experimental laptop, using the predicate "dst == 22:22:22:22:22:22". The reason we used equal-time sampling was that when an AM sees no matching frame, the strategy it falls back to is equal sampling. Also, we believe that if we had used proportional sampling, our low rate of transmission of 10 frames per second would have been insufficient to cause the proportional strategy to give more time to the appropriate channel (as the frame rate on the busy channels in the building are much higher). On the less busy channels, therefore, the equal strategy would capture more matching frames than the proportional strategy.

Figure 4.2 plots the number of frames that matched the predicate, as seen in the output of the AMs in both cases. We can see that every AM consistently captured more frames from our mobile laptop when we ran the refocusing strategy than when we ran the equal-time strategy.

In Figure 4.3 we present the number of frames that *did not* match the predicate. Although the refocused strategy captured fewer such frames than the equal-time strategy, it still provided a flow of such baseline traffic sufficient for use by other subscribers. That is, the refocusing requested by one application does not preclude ongoing monitoring by background activities, at least in this case.

4.3.2 Localization experiment

Our hypothesis is that refocusing will allow an application to more accurately, and more quickly, determine the location of a given wireless client. We chose a technique, the Nearest Neighbor in Signal Space (NNSS) method, described by Bahl et al. [11]. This localization algorithm uses observed signal strengths of frames heard by clients from APs. We used the dual of this algorithm and constructed the signal space by using the RSSI of captured frames from the client at AMs to populate our signal space.

Firstly, we calibrated the corridor of the third floor of our building. We measured the signal strength at every AM from the frames of a client transmitting 50 frames at every five feet along the corridor. In this phase, we configured all of the AMs to capture traffic on channel 1, and configured the client to transmit on channel 1. In the second phase, we configured the AMs to sample equally on every channel, and we captured a trace of the client transmitting 10 frames at every 10 feet along the corridor. Finally, we configured

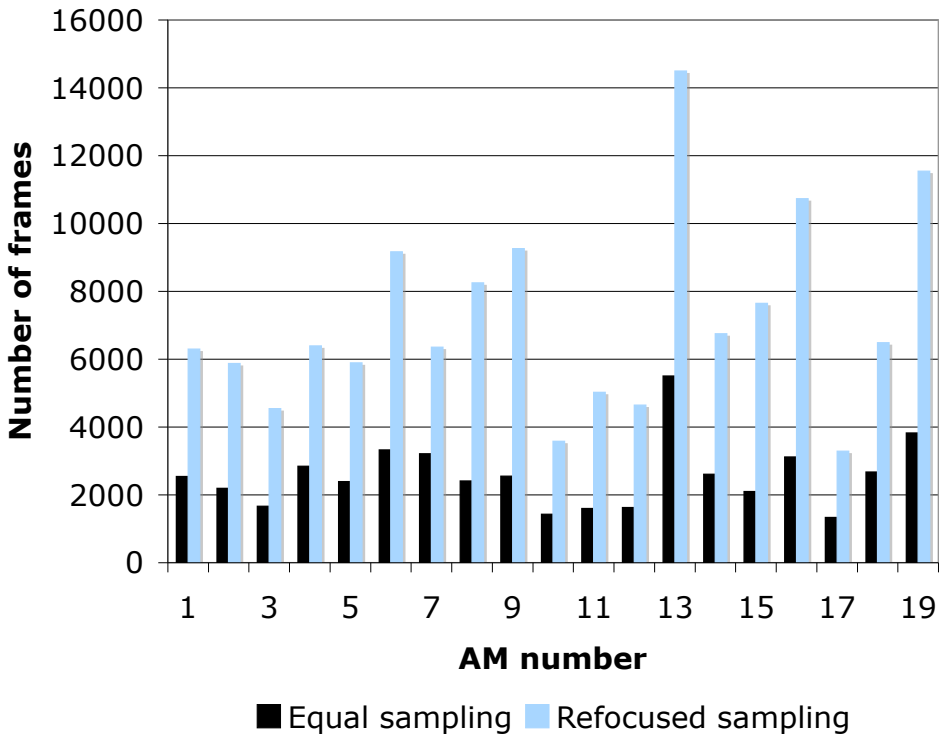


Figure 4.2: Number of frames captured that matched predicate

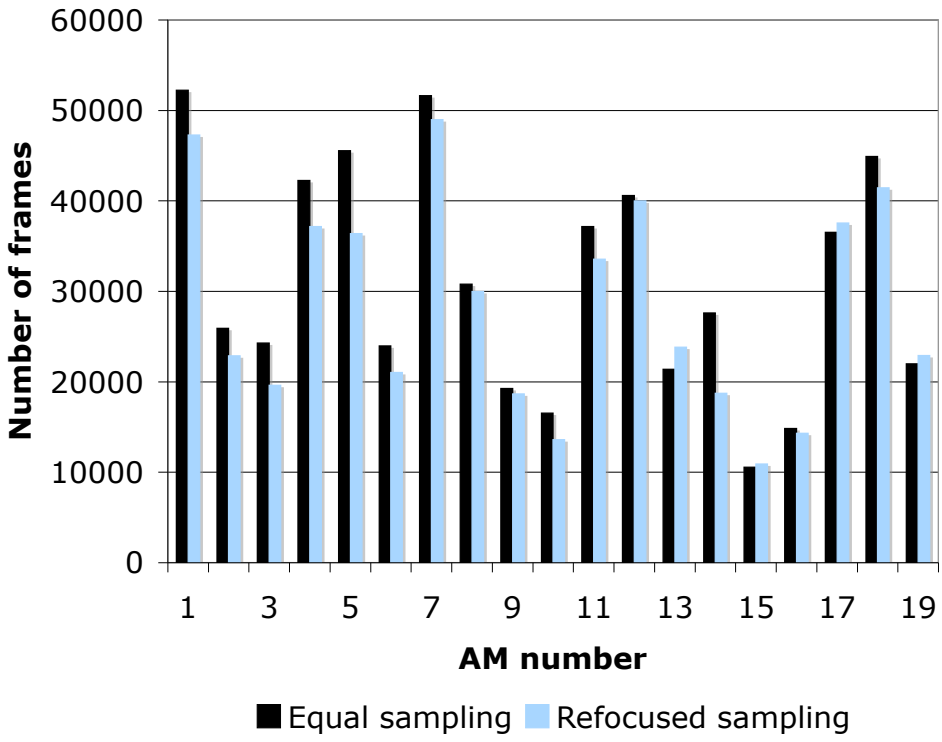


Figure 4.3: Number of frames captured that did not match predicate

the AMs to refocus on the MAC address of our client and captured a trace of the transmissions of the client at the same locations as in the second case. With our refocusing mechanism we observe localizations that were, on average, 1.95 feet more accurate than without refocusing.

Across all locations in the trace collected using refocusing, we averaged the error in localization using 1, 2, 3, ..., 15 frames. Figure 4.4 shows that average error decreased (from 34 feet when we use only 1 frame to about 25 feet when 15 frames are used,) as we use an increasing number of frames for estimating the location of the transmitting laptop. There are upward jumps where in some cases the average error worsens with increasing number of frames, but overall there is a clear downward trend. This trend shows that there may be cases when a higher number frames are desirable when using channel sampling. Refocusing will enable the system to more quickly capture a larger number of matching frames.

4.4 Conclusion

In these experiments we demonstrated that refocusing achieves the stated goal of capturing a subset of the traffic that is more relevant to the consumers of the traffic. The focus of the monitoring system can be based on any field of the 802.11 header and the physical-layer headers that are available for each frame that is captured. The predicates can be composed using Boolean operators, and a combined focus that is needed satisfy the refocusing requirements of multiple downstream components can be met. The Proportional strategy described in the previous chapter is simply a "true" predicate, in which the counter for the

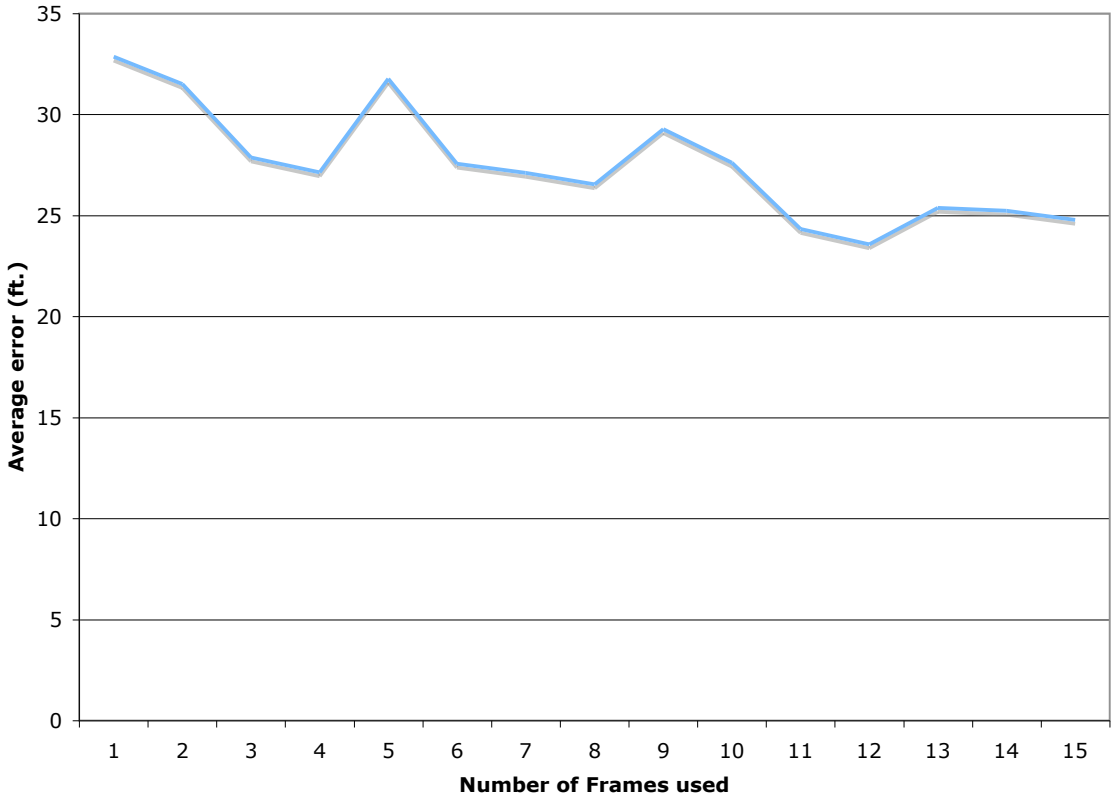


Figure 4.4: Error in localization on using varying number of frames for estimation.

appropriate channel is incremented for every frame observed on the channel. Similarly, the Equal strategy from the previous chapter can be described with a “false” predicate, which matches no frame and the proportions on the channel remain equal (at 0).

Therefore we claim that the refocusing functionality makes our framework substantially more general and it subsumes the strategies described in the previous chapter.

CHAPTER 5

COORDINATED SAMPLING AND MORE EFFICIENT CAPTURE

Consider a monitoring system with AMs deployed densely enough to be in range of any client in the monitored area; there will necessarily be some areas covered by more than one AM. That is, more than one AM can sniff frames from the same client. We say that two AMs are *neighbors* if they have recently captured a redundant frame. Redundant frames are, in most cases, wasted effort (that is, the time that a sniffer spent capturing a redundant frame could have been used to capture a frame, on another channel, that would otherwise have been unseen). When employing proportional (or refocused) sampling, neighboring AMs will observe the same channel to be busy (or to contain more frames matching the refocusing predicate) and therefore choose to spend more time on that same channel. We define *overlap* as the total amount of time that neighboring AMs spend on the same channels. This overlap results in redundant frame capture by neighboring AMs. Therefore, to better address the goal of maximizing *unique* frame capture we need to reduce the amount of overlap, hopefully resulting in the AMs capturing more frames from distinct channels. When considering the resources required to capture frames, a smaller overlap results in a higher efficiency.

Our hypothesis is that scheduling the channels on AMs, such that the coverage includes minimal overlap, as shown in Figures 5.1 and 5.2, should result in greater unique frame capture.

In this chapter we describe a "coordinated sampling" strategy and compare it to an independent sampling strategy that does not consider neighbor relationships. As clients move, we anticipate that the neighbor relationships among AMs in the network will change and the traffic volume on channels will fluctuate. Therefore, our coordination must dynamically change the schedule provided to the AMs. For example, a client may move from a location covered by a single AM to a location where the coverage of two AMs overlaps, causing those two AMs to recognize that they are now "neighbors".

Our approach has three goals:

- maximize unique traffic capture through proportional sampling,
- capture representative traffic by ensuring that all channels are sampled and that there is coverage over space and time, and
- minimize redundant frame capture by coordinating neighbor's schedules.

Our approach recognizes three constraints:

- a single radio can capture traffic only on one channel at a time,
- deploying a sniffer costs money and space, hence limits deployment, and
- no frames are captured during channel changes, which take time.

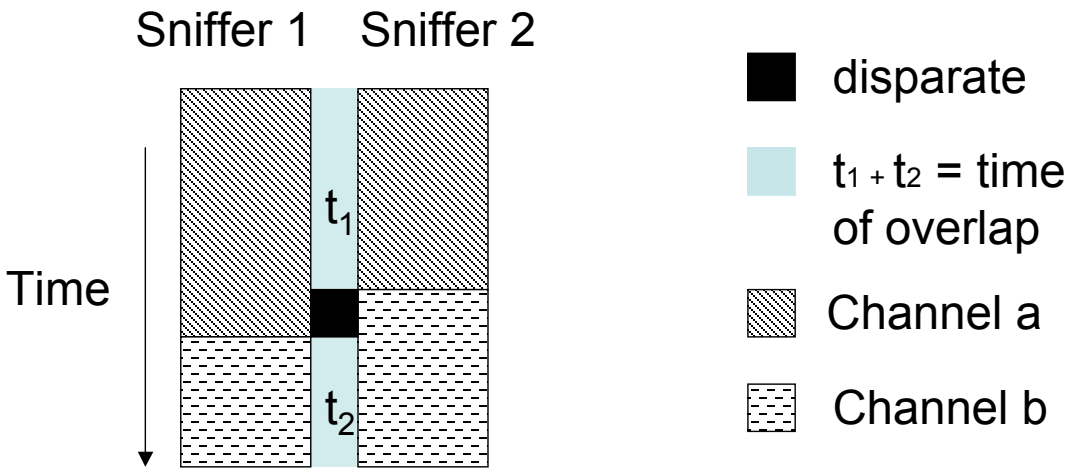


Figure 5.1: Poor overlap between AMs.

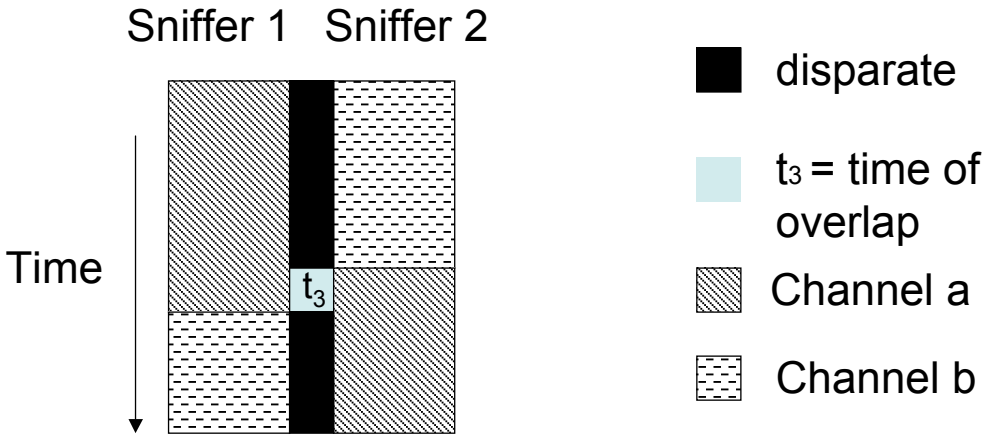


Figure 5.2: Reduced overlap between AMs.

5.1 System Architecture for Coordinated Sampling

The *amsniffer* component of dingo is flexible and is capable of running a static schedule provided to it by the *amcontroller*. We use this facility to run efficient schedules on the AMs (that have little overlap between neighboring AMs), that have been constructed on the *amcontroller* as opposed to schedules that have been constructed by hand and sent, via the controller, to the AMs (as shown in Figure 5.3).

One or more instances of *amsniffer* may be invoked when an AM is rebooted, or on demand via a network connection to the AM. Command-line options to *amsniffer* indicate which wireless interface should be employed, a default sniffing strategy to be followed, and to where the frame-capture information should be sent. We have dual-radio sniffers; our experiments have determined that it is better to invoke two distinct instances of *amsniffer* per AM, each listening on a different interface, than it is for a single process to monitor two interfaces in an interleaved manner.

A sampling *schedule* specifies a sequence of channel numbers and the duration, in milliseconds, for which the interface should listen on each channel. Our experiments with our wireless routers have demonstrated that there can be a significant delay when changing from one channel to another, and that this delay is minimized by visiting all available channels in ascending order. A typical *cycle* involves visiting each channel, capturing frames, transmitting a summary of each frame to the merger, collating and forwarding simple statistics about the traffic to the *amcontroller*. Each instance of *amsniffer* executes its current schedule for an indicated number of cycles or until directed by *amcontroller* to commence execution of a new schedule.

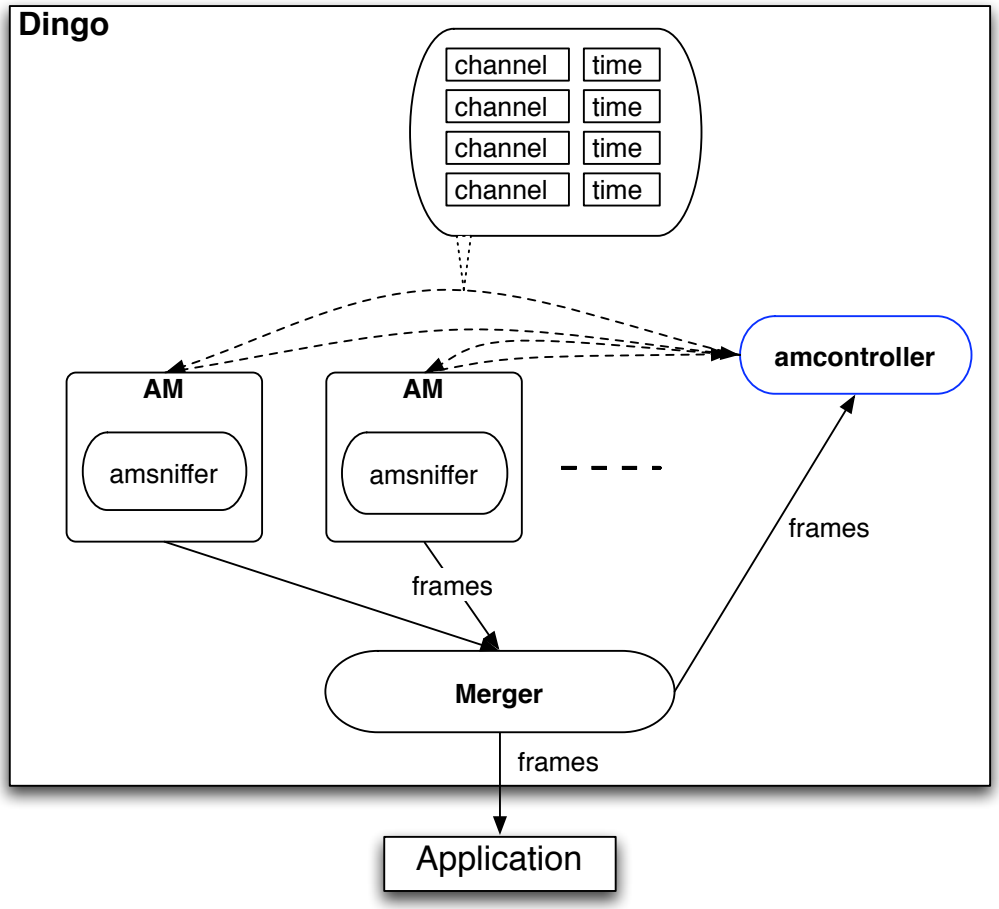


Figure 5.3: The *amcontroller* generates schedules that are sent to the AMs. These tell the AM what time to switch channels and for how long.

Each *amsniffer* maintains a set of simple counters including the number and the total length of frames captured on each channel during a scheduling cycle. At the end of each cycle, each counter's value is sent to the *amcontroller* for future scheduling decisions, and each counter is cleared. Our earlier strategies (Chapters 3 and 4) are based on these simple counts gathered at the AMs. For example, the *proportional* strategy spends time on each channel proportional to the recently observed frame rate on that channel.

The *amcontroller* is a subscriber of merged traffic from the merger. Using the merger's stream of individual frame information, the *amcontroller* maintains a neighbor graph recording which pairs of AMs recently saw the same frames. A edge $e_{i,j}$ exists in the graph if AM_i and AM_j have recently captured at least one common frame. This graph is recomputed periodically to reflect the changing nature of the captured traffic. For example, the combination of AMs capturing common frames from a particular mobile client will vary over time as that client moves. Over a longer period of time, the same sets of AMs are anticipated to form long-lived overlapping clusters, reflecting their physical proximity to each other and thus, their likelihood of seeing common frames. With reference to our goal of maximizing the number of unique frames captured, we need to ensure that neighboring AMs are not simultaneously listening on the same channels, so as to reduce the likelihood of them capturing common frames.

Periodically, every 20 seconds in our implementation, the *amcontroller* constructs the neighbor graph and uses recent frame counts to build a new coordinated schedule, as described in the previous section. The full schedule consists of a new schedule for each AM. Each schedule ensures that its AM listens on each available channel for a specified period,

avoiding (as much as possible) overlap with its neighbors' schedules. Each AM's new schedule is transmitted to the AM for execution until the arrival of a new schedule.

5.2 The coordination algorithm

We developed the following approach to reduce the amount of overlap among neighboring AMs, based on statistics of recently captured traffic.

The output of the coordinated sampling strategy is a channel sampling schedule for each AM, identifying the order and duration of visits to each channel. Consider the set of AMs $AM_1, AM_2, ..., AM_n$. Let these be vertices in a graph. Let there be an edge $e_{i,j,c}$ in the graph if the two AMs AM_i and AM_j are neighbors and are also on the same channel c; as time progresses, edge $e_{i,j,c}$ comes and goes in the graph as AM_i and AM_j follow their respective schedules. Let $t_{i,j,c}$ be the amount of time that the edge $e_{i,j,c}$ exists in the graph. The objective function is as follows:

$$Z = min \sum_{i=1..n} \sum_{j=1..n} \sum_{c \in C} t_{i,j,c}$$

where Z is the minimum overlap across all schedules. The minimum value of Z is unknown. The search space of all possible schedules is very large, C^n, where C (the set of available channels) is on the order of 11-25, and n (the number of AMs) could be in the tens or hundreds. For example in our experiments, the search space of schedules among all AMs would be 11^{20}. Clearly, a brute-force search of all possible schedules to find the value for Z is infeasible. Instead we use an approach for minimizing the overlap that is inspired by the method of simulated annealing. Our method generates a series of schedules

by perturbing each schedule a little. If a new schedule has lower overlap, we keep it; if not, we keep it anyway with probability p. This method allows our algorithm to jump out of local minima. Each time we reduce p by a factor of δ and terminate the algorithm when $p \leq P$, where $\delta > 1$ and $0 < P < 1$ are tunable parameters.

Our algorithm is portrayed in Figure 5.4.

This algorithm takes a randomized, greedy approach: if we reduce the overlap between two neighboring AMs, the overall overlap is likely to reduce as well. However, even if it does not, we accept the permutation of a schedule with a (decreasing) probability p. We rotate the channel ordering of one AM in a pair of neighboring AMs with the maximum pairwise overlap so that we can take larger steps towards a more efficient solution. This choice seems to work well in our experiments. In our coordinated sampling experiments (described later), we start off all the AMs using a schedule that spends equal time on every channel, and run the above algorithm every 20 seconds.

5.2.1 Simulation Results

We conducted simulations using 20 AMs, randomly generated proportional schedules and a random network topology: we assigned twenty AMs IDs 1 to 20 and a pair of AMs (with IDs *AMID1* and *AMID2*) were neighbors if $|AMID1 - AMID2| > 3$ (so that the number of neighbors of every AM is always constant). In these simulations, we used the coordinated sampling algorithm described above and compared it to a purely greedy approach where no schedule with a greater overlap is ever picked. Our probabilistic approach enabled the schedule to jump out of local minima and consistently produced schedules that had 5 to 7 percent lower overlap than the pure greedy approach. The pure greedy approach termi-

1. Identify the neighbor relationships among all AMs.

2. Create a new schedule S in which each AM spends time on each channel in proportion to traffic measurements provided by the AMs, and a minimum amount of time on quiet channels. Each AM's schedule starts with a randomly selected channel but progresses in order around all channels.

3. Calculate pairwise overlap t_{ij} between every pair of neighbors i, j, and the total overlap $T = \sum t_{ij}$.

4. Set $p = 1$.

5. **do** *// permute AM schedules to find a better schedule*

   (a) Among all pairs of AMs not yet considered in this loop, choose the pair with maximum overlap and rotate the channel order of one of the two AMs to create a new schedule S'; compute its overlap T'.

   (b) If the new schedule has less overlap ($T' < T$), retain it (set $S = S'$ and $T = T'$); otherwise, with probability p, retain it anyway.

   (c) Set $p = p/\delta$.

   while $p > P$ and there are other pairs to consider.

6. Accept schedule S and send each AM its new schedule.

Figure 5.4: The coordinated sampling algorithm

nated in approximately 11ms, whereas the randomized approach took between 35-45ms to terminate. We ran the experiment 100 times with different random starting schedules each time. We also used the standard "simulated annealing" algorithm [43, 48]. This approach resulted in schedules that have about 20-30 percent greater overlap than our approach. The simulated annealing approach also took >200ms to terminate. We ran this simulation 100 times as well, with a different random starting schedule every time.

Thus, we see that our randomized and greedy approach not only results in more optimized schedules, but also does so in less time.

5.3 Coordinated Sampling Experiments

We ran our experiments on the Aruba AP70 testbed described in Section 3.6. Figure 3.7 shows the floor plan and location of the AMs. We conducted our experiments on 802.11b. The production APs for the wireless network operate on 802.11b/g channels 1, 4, 8 and 11. Several experimental networks (such as a mesh network and some experimental APs) operate on channel 11.

5.3.1 Experimental setup

We used both radios of each AP70 by running two copies of *amsniffer*. One instance of *amsniffer* ran an independent proportional strategy while the other ran the coordinated strategy described above. We used $\delta = 1.08$ and our P value was 0. (We used integer values in our implementation and p ran down from RAND_MAX to P(=0). We generated a random number between 0 and RAND_MAX using a uniform random number generator

to compare with p in every iteration.) We used the cycle time of 2.2 seconds for both strategies. The independent sampling strategy however, re-calculates the proportions every 2.2 seconds and the coordinated sampling strategy re-calculates the proportions every 20 seconds. As a result, the coordinated strategy is less responsive to the short term dynamics of the traffic.

The sniffers running the independent strategy forwarded the frames they captured to one merger, and the sniffers running the coordinated strategy forwarded the frames they captured to another merger. The *amcontroller* received the output of the latter merger and used this information to create and update the neighbor graphs. As we were only merging traffic from 19 AMs, we could use the same server for both mergers and the amcontroller. In larger networks, we would run multiple mergers on different servers.

The output from each of the mergers was saved to disk for later analysis of the performance of the respective schemes.

5.3.2 Results

We conducted two experiments over a period of one hour each, and recorded the number of unique frames captured by both the proportional and coordinated approaches in 20-second intervals. On every invocation, the coordinated scheduling algorithm described earlier required a total of between 33 and 35 milliseconds to determine new schedules for all of the 19 AMs; we consider this cost insignificant relative to the frequency of scheduling. Initial random channel assignments resulted in a total channel overlap time of between 128 and 167 milliseconds. After an average of 253 iterations, this total channel overlap time was reduced to between 23 and 32 milliseconds (indicating an 80% reduction in overlap from

the random schedule to the coordinated schedule), meaning that neighboring AMs were successfully scheduled for different channels most of the time. The randomized portion of the algorithm entailed that every time the algorithm was invoked, the number of iterations that reduced the overlap ranged between 34 and 38, the number of iterations that increased the overlap but were not chosen were approximately 200 and the number of iterations that increased the overlap and were chosen anyway ranged between 2 and 6.

Our logs indicated that the number of active neighboring AM pairs observed every time the neighbor graph was recomputed (every 20 seconds during the experiment) was between 10 and 65. The mean number of neighbor pairs was 38.3 and the variance was 42.9. This indicates that the AM neighbor relationship was highly variable. This observation further supports our assumption that there is overlap in coverage of different AMs and that this overlap varies with time, indicating that we may need to recompute the neighbor graphs and the schedules often.

The data collected shows that coordinated sampling was successful for increasing unique frame capture. Figure 5.5 shows that, over a one-hour period, coordinated sampling captured over 10% more unique frames in nearly all 20-second intervals than did independent proportional sampling. The Student's t-test applied to these two sets of capture indicates that the means of the two sets of data were significantly different (t = 2.85, p-value = 0.005567). This result supports our hypothesis that coordinated sampling will, by using global information, decrease overlap and capture frames more efficiently.

Figure 5.6 is the histogram of the number of frames that were seen by one AM, two AMs and so on. The histogram of the coordinated-sampling experiment is markedly more skewed towards the left than the histogram of the independent-sampling experiment. We

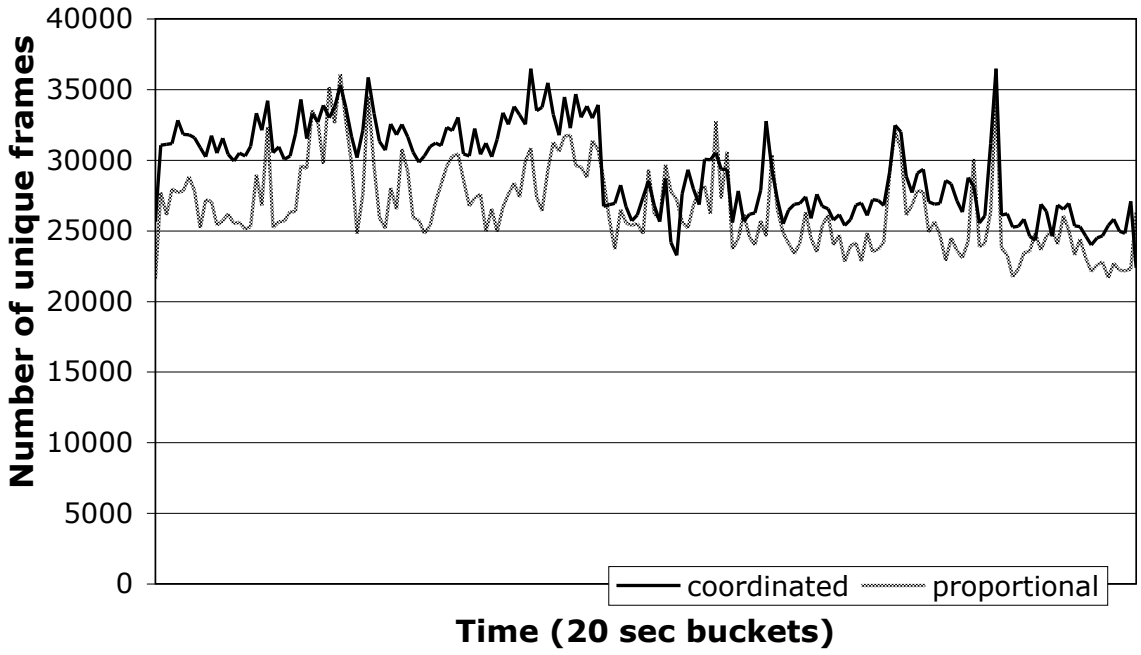


Figure 5.5: Unique frame capture over one hour.

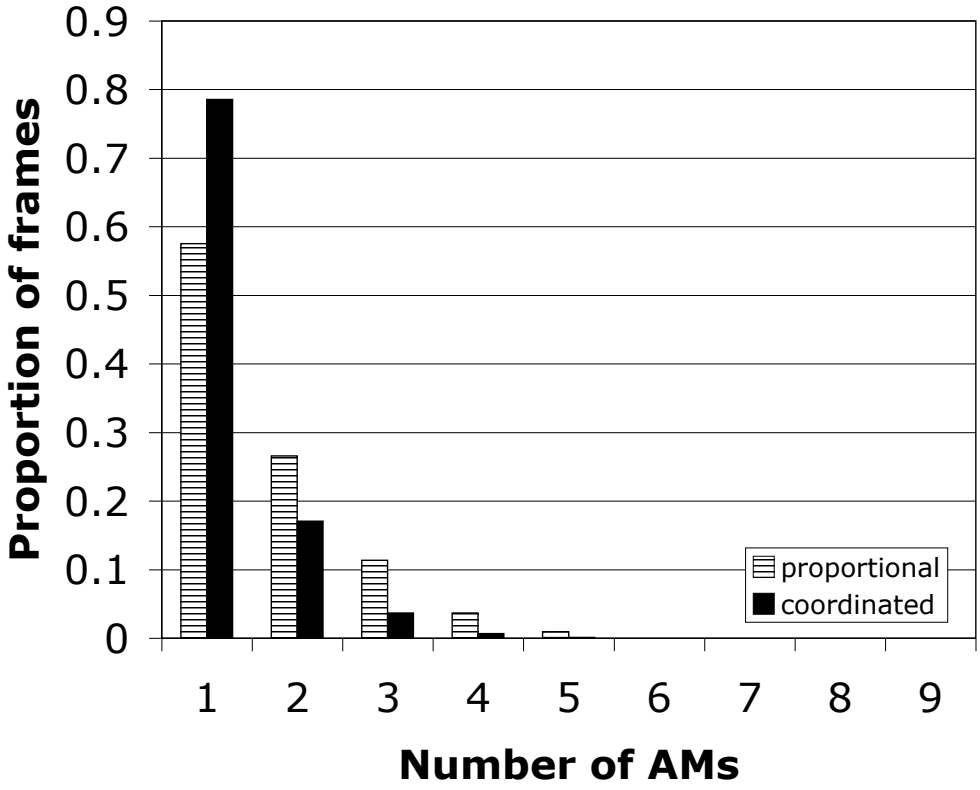


Figure 5.6: Number of AMs capturing each frame over one hour.

see that the number of frames exclusively captured by one AM under coordinated sampling was about 35% more than the number of frames exclusively captured by a single AM under independent-sampling, thus achieving our goal of reducing redundant capture.

This result is logical because coordination reduces the amount of time that neighboring AMs overlap in their channel. Neighboring AMs are more likely to hear the same frames than distant AMs. If neighboring AMs spend less time on the same channel simultaneously, there will be fewer cases in which two or more AMs hear the same frame.

5.4 Mobility and Coordinated Sampling

Mobility can affect the behavior of coordinated sampling. As clients move through the monitoring area, there will be churn in the neighbor relationships between AMs, and therefore the time spent on different channels in different schedules will change.

Number of mobiles. Client mobility will affect the way in which the neighbor graph is computed in coordinated sampling. If only one client is situated in the coverage area of two AMs, then if those two AMs capture the same frame from the client, they are considered to be neighbors in the neighbor graph. Later, if that client moves away, then those two AMs are no longer considered to be neighbors.

If there are a large number of mobile clients in the coverage area of AMs the neighbor relationships may remain stable. However, if there are few fast moving mobiles in the coverage area of AMs with overlapping coverage, then the AM neighbor relation may frequently change. This state of flux may not be desirable as it may cause inefficiency in the capture, as decisions will be for coordination based on information that becomes stale

very quickly. The aging of the neighbor graph needs to be tuned according to the mobility observed in the monitoring area. If there are a few mobiles with very high mobility in the network, the graph needs to be recomputed more often than if there were a large number of mobiles or very few, slow-moving ones. The reason is that, if there are quick moving mobile clients, the neighbor relationships are likely likely to change quickly. Note that, if two AMs both have the potential of capturing the same frame from one location, if there is no client in that location, there is no need to "coordinate" these AMs and we can reduce the constraint on the schedule. Then, we need to quickly react when a client arrives and re-impose that constraint.

We had few mobile clients in our network and therefore, our aging took place at a relatively slow 20 seconds (i.e., if the last frame seen in common by a pair of AMs was more than 20 seconds ago, that pair of AMs are no longer neighbors). This space, however, can be explored further with different parameters used in the mobility model.

5.5 Conclusion

We found that, as per our hypothesis, neighboring AMs indeed often have overlapping coverage and this overlap varies with time. We also found that our coordinated scheduling algorithm reduces overlap up to 80% over that of randomly generated schedules.

We found that using these efficient schedules, which minimize the time that neighboring AMs spent on the same channel, resulted in a greater number of unique frames captured. We achieved our goal of maximizing the number of unique frames captured and reducing the number of redundant frames.

It should be noted that even though we used the frame-rate proportional strategy when constructing the schedules for our experiments, any strategy may be coordinated. The strategy only determines the amount of time to be spent on every channel. Once this is known, the coordinated scheduling algorithm can create schedules with lower overlap. Therefore the coordination is independent of the strategy used to determine the channel sample duration. An equal strategy can be coordinated just as easily as a proportional one, and coordination can be used in conjunction with refocusing where the strategy used is one that calculates time spent on every channel proportional to the number of frames matching some predicate.

CHAPTER 6

TRACE COMPARISON METRICS

A full trace of traffic from a network will have the most information about the network. It is difficult to capture a full trace. A sampled trace with the greatest amount of information is the next-best option. We describe several different sampling mechanisms in earlier chapters that attempt to achieve this goal.

The cost for full capture is much higher than the cost for sampled capture. The full trace will require many more radios to capture the trace for the same length of time. If a monitoring system has one AM per location, full capture would require at least in n times the number of radios as sampled capture would, where there are n channels to be monitored. If, using sampling, we can gather information similar to that of full capture, the "cost" of monitoring is much reduced and the goals of monitoring are better satisfied.

It is natural to question how well a captured trace using sampling represents the hypothetical full trace. In this chapter we describe *trace comparison* methods and metrics we have developed to evaluate the ability of our network-sampling methods to capture representative traces. Furthermore, such similarity metrics can be used to compare traces in other contexts.

In the following sections we describe and evaluate our methods and metrics for comparing network traces, and we describe the setup and results of our experimental evaluation.

6.1 Trace Comparison

Our goal is to develop methods to measure *how closely* a sampled trace represents a full trace. This measure of closeness (or similarity) is one way to estimate the "quality" of our sampling techniques.

Furthermore, if we can characterize a trace with a compact representation, we can quickly compare it to a large set of other traces. Given a large collection of traces, such as those in CRAWDAD,[1] a researcher may be able to quickly determine which traces from the collection are "similar" to a trace in question, allowing the researcher to choose only those that are similar— or dissimilar— for download. The definition of similarity depends on the planned uses of the trace, so we need to develop generalizable methods. For example, one researcher may be interested in mobility patterns and another may be interested in the traffic mix at the MAC layer.

To achieve this goal, then, we must be able to characterize the trace by extracting representative attributes as a vector of numbers. A method to compare two or more vectors is then required so that we can determine the similarity, or the lack thereof, in pairs or sets of traces.

6.1.1 Representative attributes of 802.11 traces

802.11 traces consist of individual 802.11 frames. Each frame consists of multiple fields and a capture timestamp, which help distinguish it from other frames. Frames of different types and subtypes, from various sources to destinations, can be counted and vectors of these counts are one way to represent the entire trace.

[1] `http://www.crawdad.org`

Fine Granularity: 10 Second Buckets

A trace can be represented by a series of counts of frames (or counts of particular frame types) over the entire duration of the trace. We created hour-long summaries using the counts of the total number of frames in 10-second intervals and the counts of the number of beacons in 10-second intervals. These take the form $<x_1, x_2, x_3, x_4, ..., x_{360}>$ where x_i's are the number of frames or beacons observed in the i^{th} 10-second interval. We used a short interval because we wanted to capture quick bursts introduced by humans periodically using the network.

We create a frequency distribution from this data and then we use the Kullback-Leibler divergence metric (also known as the relative entropy) to determine the divergence between pairs of traces represented in the above form.

KL divergence indicates the number of extra bits required in encoding samples from P by using the distribution of Q [46] where P and Q are probability distributions (in our case, each corresponding to one trace).

The divergence of Q from P is defined to be

$$D_{KL}(P||Q) = \sum_i P(i) \times log \frac{P(i)}{Q(i)}$$

For two distributions, P and Q, the smaller the value of D_{KL} the *closer* the distributions are. Usually, Q is a model that represents P. In our case, Q is extracted from the sampled trace and P is extracted from the full trace.

Coarse Granularity: Hour-long buckets

A representative vector can be a list of counts of various types and subtypes of frames observed in a trace. For example,

<# of control frames, # of management frames, # of data frames>

or it may be a higher-level observed statistic of the trace. For example,

<# of new clients per minute, # of frames per minute, # of frames per minute per client>

may be representative of a trace if the application in mind is in some way related to observed mobility within a trace. The choice of this tuple is important as it must capture the specific attribute of the trace in which the user is interested.

The 802.11 frame format defines a 2-bit frametype field, to characterize a frame as a management, control, or data frame, and a further 4-bit subtype field to further characterize the frame within its type. Of the 64 possible frametypes, many are reserved for future use, leaving only 25 frametypes defined. Of these, many are rarely seen in actual traces, either because of their specialized role or because the network being monitored may not be used in a way that requires that frame type. For example, a network may employ no ad-hoc networking and, thus, only carry frames between mobile clients and access points. The capture of unexpected frametypes, even reserved frametypes, may identify the presence of an intruder transmitting a specific attack sequence. We did not see any of the reserved frametypes in our traces, however.

In our process of choosing the fields to use for the tuple, we went through the following steps:

1. Summarize each trace into a vector based on frame attributes; in our case, we listed the number of each valid frame subtype in the trace as members of a 25-element vector.

2. Discard those subtypes that have a very low occurrence. We encountered several subtypes that occurred only once or never in every hour-long trace.

3. Calculate the Coefficient of Variation (COV) for the counts of each remaining frametype. Sort them from the highest to lowest. This step tells us which of the frame subtype counts have the greatest variation.

4. Choose the three frame subtypes with the highest COV such that they are not correlated to each other (for example, Deauthentication and Disassociation frames are usually sent by APs in pairs, and RTS and CTS occur in pairs as well). Use these *triples* as data points that represent each hour-long trace. Our analysis, of traces we collected in Sudikoff, revealed these frametypes to be the Deauthentication management frametype, the Request-to-Send (RTS) control frametype, and the standard Data frametype. We had no initial intention to employ one of each of the three frametype categories; the result is just a coincidence. The results are shown in Figure 6.1.

5. Normalize the set of these data points such that the length of each vector is 1.

6. We cluster the resulting data points as described in the following section.

Similarity metrics can be computed at different levels of granularity in wireless traces. For example, an hour-long trace may summarized as a 3–tuple of counts of most variable subtypes (as in our analysis), if the gross traffic patterns in that trace vary little. However, if

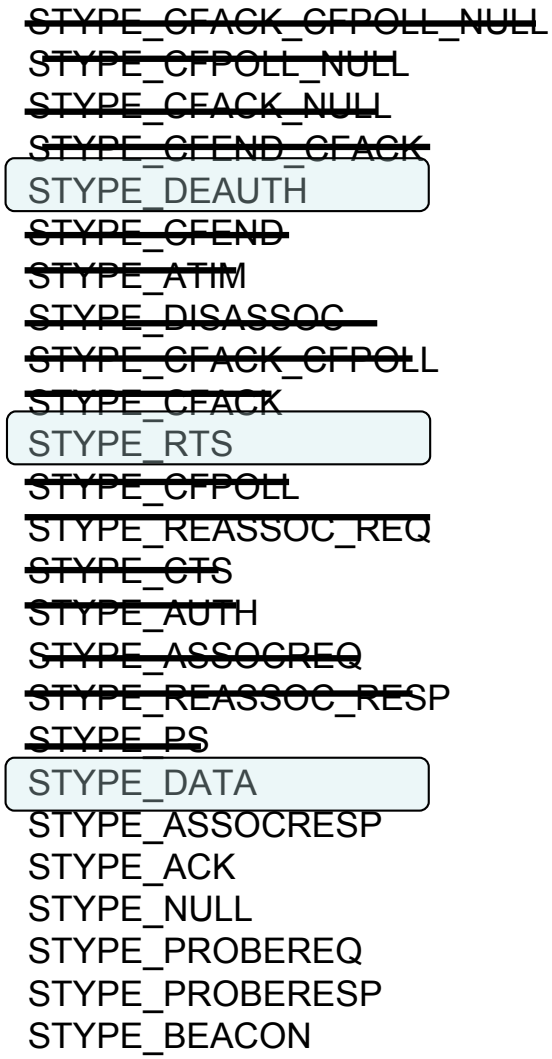


Figure 6.1: The final three chosen frame types. The ordering of frametypes is descending from top to bottom according to COV. The circled frametypes represent the ones chosen for the tuple. The crossed-out frametypes represent the ones that have a higher COV but are discarded because of low means or because of correlation with other, chosen, frametypes.

a trace is collected across a whole week, a mere summary of the frametype counts may hide most of the high-level variation in the trace. The degree of summarization depends on the granularity of the traffic that the user or application needs. Indeed, the user may not wish to hide the variation that may take place within an hour, making the hour-long summaries too coarse. We determined that for our purposes, there was sufficient variation in the week-long trace so that summaries of hour-long slices of this trace provided sufficient granularity. (The hour-long trace would be sufficiently large to include gross group behavior, like the length of a class, which is either 50 minutes or 1 hour 50 minutes long.)

We gathered all the single-hour summaries of the week-long traces collected in our experiments after the degree of granularity was determined. We employed the standard "K-means" clustering technique to group *similar* hour-long traces into clusters. K-means clustering algorithms partition data points so that the resulting clusters minimize the total intra-cluster sum-of-square distance of all points in each cluster. Our selected K-means algorithm requires, as input, the number of desired clusters and the number of required iterations. We employed the "R" statistical package [58] and the clustering packages available in that framework (Rcmdr [28]). To plot the 3-dimensional clusters, we employed the package "Scatterplot3d" [49].

The K-means algorithm needs a distance metric so that it can compute the nearness of different points in the dataset. We use Euclidean distance as the distance between points in the 3-space that we have constructed.

We represent each trace as a set of clusters. Each cluster is a set of tuples from our summaries of hour-long slices of the full week-long trace. Clustering enables us to gather together multiple "similar" (close in Euclidean space) into a much smaller and manageable

set of points (centroids of the clusters). Once the clusters for a trace are constructed, we need to be able to compare one "clustering" (set of clusters) that represents a week-long trace to the clustering from another trace.

We know that each cluster has a cluster "center" to which the points in the cluster are nearer than centers of other clusters. As the K-means algorithm is parameterized with the number of centers, we can ensure that every clustering has the same number of clusters. If we have two such clusterings C1 and C2, we attempt to match the each cluster in C1 to a cluster from C2 such that the sum of the distances between centers of matched clusters is the minimum as shown in Figure 6.2. If we consider these centers to be vertices of a graph, and the distances between them are the weights of edges between the vertices, then, we can see that the problem is the same as the *Assignment problem* [45] or the *Stable Marriage Problem* [30] where the cost of the matching is either to be minimized or maximized. (In Figure 6.3 the two sets represent the clusterings and the members of the set are cluster centers. The arrows represent a bijection between the sets and the weights are distances between cluster centers.)

This minimum sum of distances is our *Dissimilarity metric.* If the minimum sum obtained is higher in one pair of clusterings A and B than another pair C and D, then A and B are more dissimilar than C and D. The following equation shows how to calculate the dissimilarity between two set of points in space:

$$D = \min_{\forall S \in A \leftrightarrow B} \left\{ \sum d(a_i, b_j) \mid (a_i, b_j) \in S \right\}$$

where $d(a, b)$ is the Euclidean distance between two points a and b and $\leftrightarrow$ is a bijection between two sets, and A and B are two clusterings.

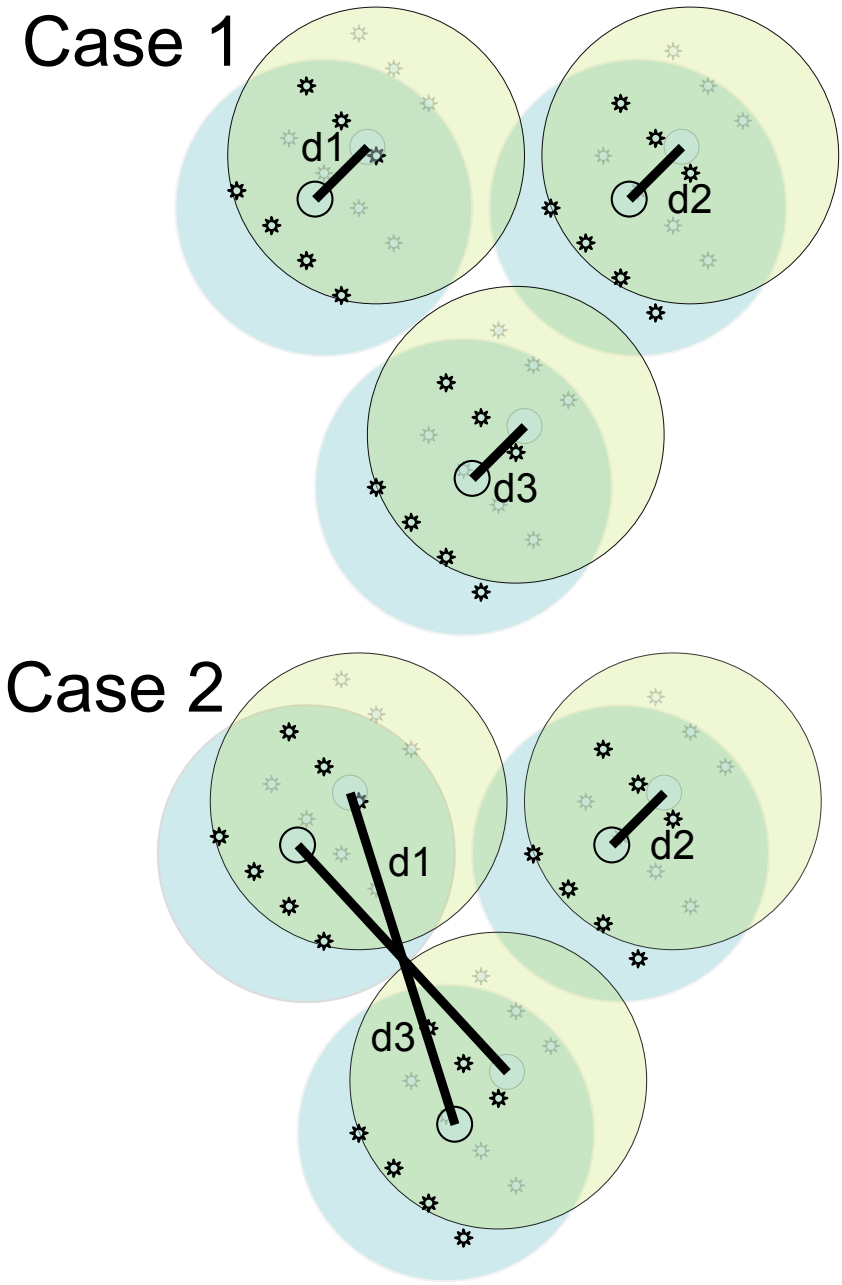


Figure 6.2: The sum of distances between cluster centers in the matching used in Case 1 is less than the sum of distances between cluster centers in the matching shown in Case 2.

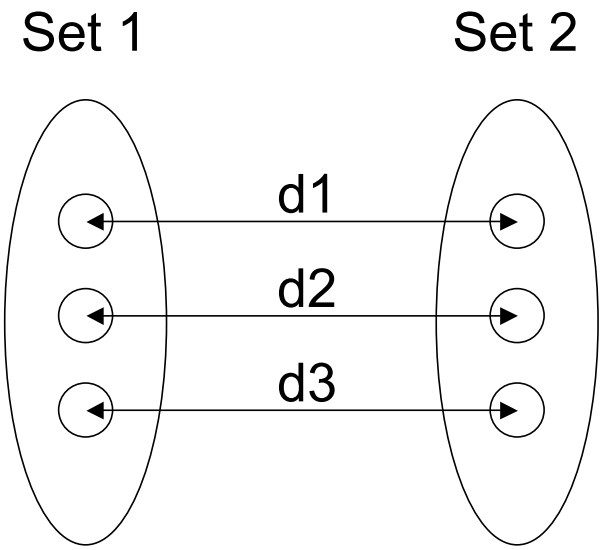


Figure 6.3: The Sets represent clusterings and the elements of the sets are cluster centers. The arrows are a matching between the sets and the weights are the distances or "cost" of the individual pairing.

6.2 Related Work

The evaluation section draws on work done in classification, information retrieval and document retrieval. We used the Euclidean similarity metric after also exploring the cosine metric described by in the area of text retrieval [61, 15, 38]. We use the "R" statistical package for all our computations [58].

Petrovic et al. [55] use cluster evaluation metrics in an IDS by detecting compact clusters. Their hypothesis is that attack traces form tight clusters because the frames are all similar to each other. They do not use clustering to compare sampled traces to full capture traces.

We do not know of any previous work that uses clustering techniques to compare sampled network traces to full traces.

Table 6.1: First week

Hour	AM Set	Radio Used	Strategy
1	1	wifi0	Full Capture on channel 1
1	1	wifi1	Static/Equal/Time
1	2	wifi0	Full Capture on channel 1
1	2	wifi1	Frames/Proportional/Time
2	1	wifi0	Full Capture on channel 1
2	1	wifi1	Frames/Proportional/Time
2	2	wifi0	Full Capture on channel 1
2	2	wifi1	Static/Equal/Time
3	1	wifi0	Full Capture on channel 1
3	1	wifi1	Static/Equal/Time
3	2	wifi0	Full Capture on channel 1
3	2	wifi1	Frames/Proportional/Time
.	.	.	
.	.	.	

6.3 Experimental Results

We used the same monitoring testbed described in Chapter 3. We divided the 20 AMs (Figure 3.7) into two groups of 10 AMs each, and we used both radio interfaces on each AM. We ran two week-long experiments so that we could compare the traces from three different sampling strategies with each other and with a full (unsampled) trace from channel 1. The parameters of the two week-long experiments are described in Tables 6.1 and 6.2. Note that interface "wifi0" remained on channel 1, to obtain "full capture" while interface "wifi1" alternated two sampling strategies.

In this section we use the Kullback-Leibler divergence metric to determine the divergence between the sampled trace and the full capture.

Table 6.2: Second week

Hour	AM Set	Radio Used	Strategy
1	1	wifi0	Full Capture on channel 1
1	1	wifi1	Coordinated Sampling
1	2	wifi0	Full Capture on channel 1
1	2	wifi1	Frames/Proportional/Time
2	1	wifi0	Full Capture on channel 1
2	1	wifi1	Frames/Proportional/Time
2	2	wifi0	Full Capture on channel 1
2	2	wifi1	Coordinated Sampling
3	1	wifi0	Full Capture on channel 1
3	1	wifi1	Coordinated Sampling
3	2	wifi0	Full Capture on channel 1
3	2	wifi1	Frames/Proportional/Time
.	.	.	
.	.	.	

Our hypothesis is that our new the clustering distance metric will demonstrate that the sampled traffic that we captured in our experiments is "closer" to the full trace captured simultaneously on another radio of the same sniffer than to traces from a different week.

6.3.1 Similarity in Capture Volume

As mentioned above, we counted the number of beacons in every 10-second interval on channel 1 of an hour-long trace. As described above, for every hour of sampling on any AM, we have its corresponding full trace captured on a different interface of the same AM. Clearly, the absolute number of beacons intercepted by the sampling technique on channel 1 will be far smaller than the absolute number of beacons intercepted by the full capture. (Beacons are very regular and are sent at a lower transmission rate. Therefore the number of beacons observed in a sampled interval should closely track the number of

beacons observed in a full trace.) We know, however, the fraction of time of the 10-second bucket that the sampling technique spends on channel 1. We can estimate the number of beacons the sampling interface *would* have observed had it been capturing fully on channel 1. We do this by simply scaling the number of beacons observed by the sampling interface up by the inverse of the fraction of time it spent on channel 1 (Figures 6.4–6.6). This method gives us numbers for the 10-second intervals that are in the same ballpark as the full capture (if indeed the sampling strategy was capturing traffic representative in terms of the relative volumes of frames in every 10-second interval). We repeated this process for estimating all the frames (not only beacons) and plotted the graphs with 10-second intervals.

Figures 6.7–6.9 show the estimated number of frames captured by the Equal, Proportional and Coordinated sampling strategies respectively. Each of these are compared to the corresponding full capture on channel 1 for the same hour.

In general, it is evident that the sampling strategies follow the trends of the baseline full-capture graph.

There are two artifacts in the graphs that need explanation. First, every 60 seconds, in all the graphs, there is a significant reduction in the frame capture during one 10-second bucket. In our experiments, we had a watchdog process that opened an SSH connection into the AM to check if the sniffer was still running. Because our AMs are resource-limited devices, and also because they were constantly running near 97% capacity, the task of creating the SSH connection caused the AMs to drop frames.

Second, we notice that the sampled interface seems to have collected between 5% and 25% more frames than the full-capture interface. In our experiments, we found that one

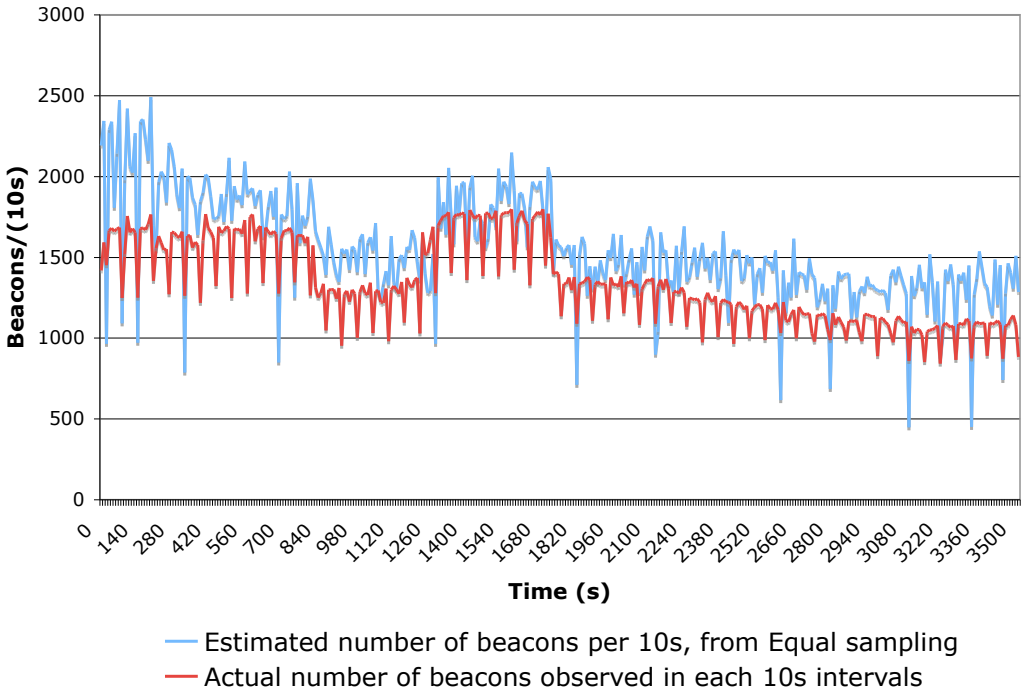


Figure 6.4: The estimated number of beacons on channel 1 captured by equal sampling every 10 seconds track the number of beacons captured by the simultaneous full trace on channel 1.

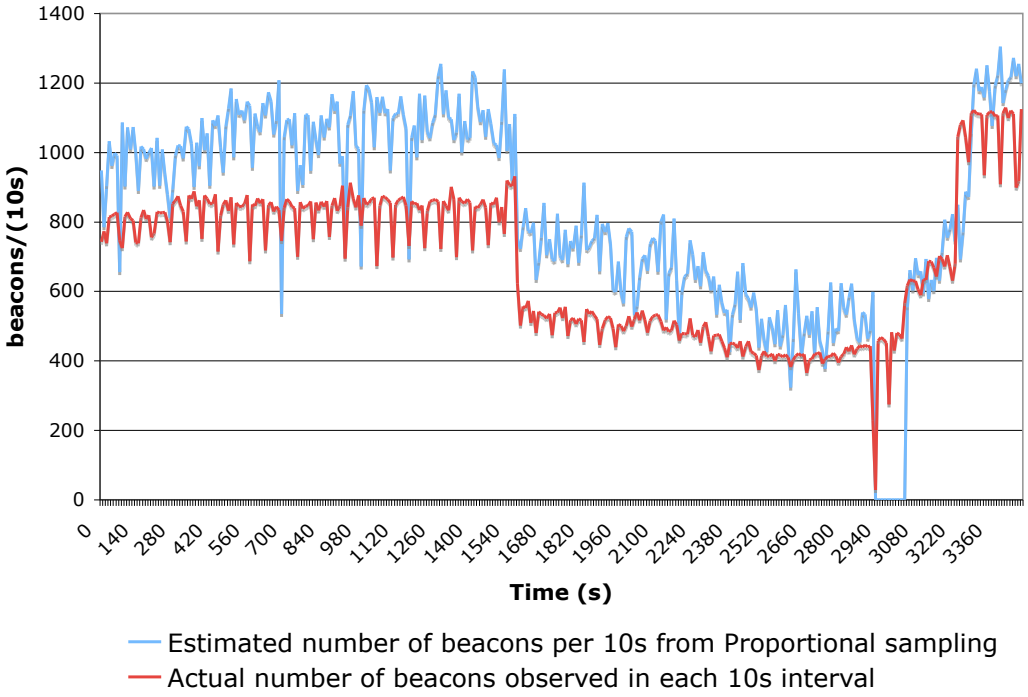


Figure 6.5: The estimated number of beacons on channel 1 captured by proportional sampling every 10 seconds track the number of beacons captured by the simultaneous full trace on channel 1.

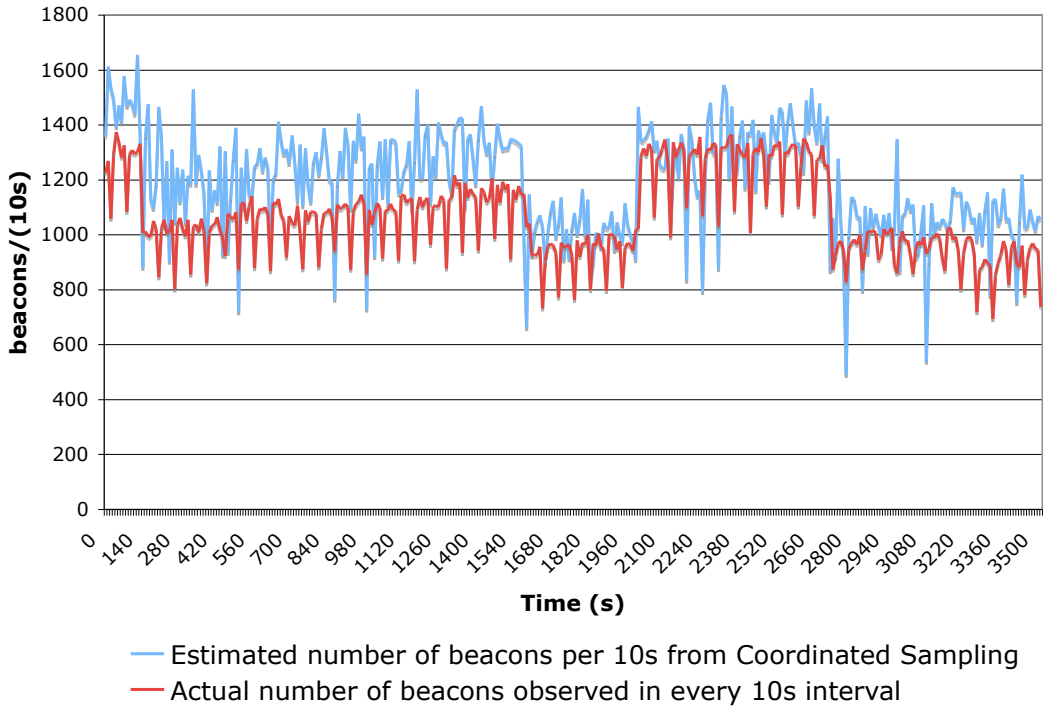


Figure 6.6: The estimated number of beacons on channel 1 captured by coordinated sampling every 10 seconds track the number of beacons captured by the simultaneous full trace on channel 1.

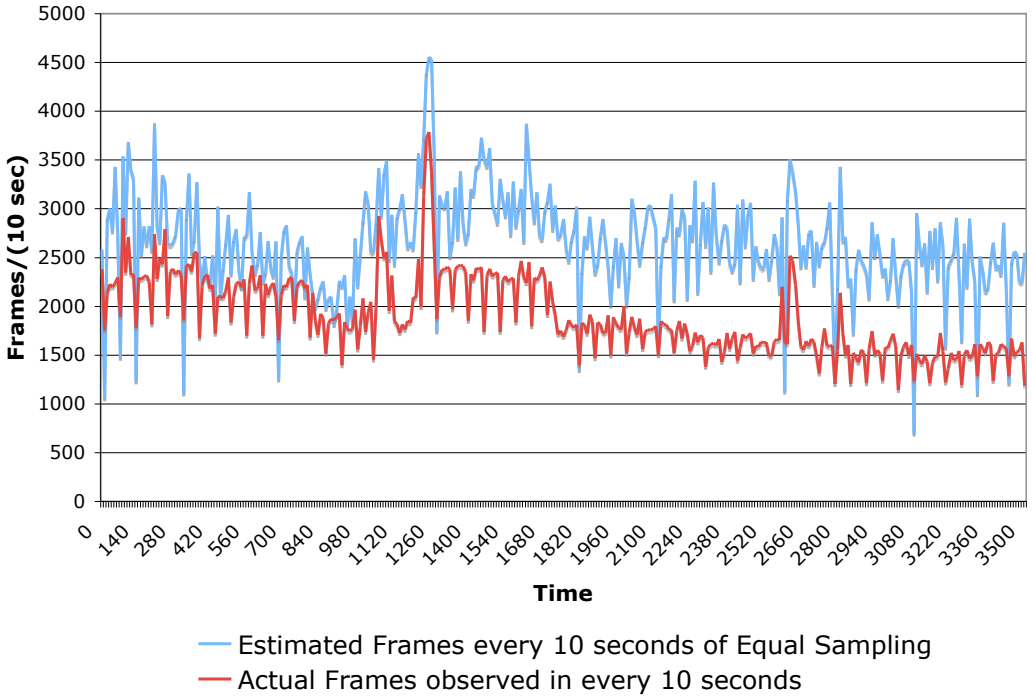


Figure 6.7: The estimated number of frames on channel 1 captured by equal sampling every 10 seconds track the number of frames captured by the simultaneous full trace on channel 1.

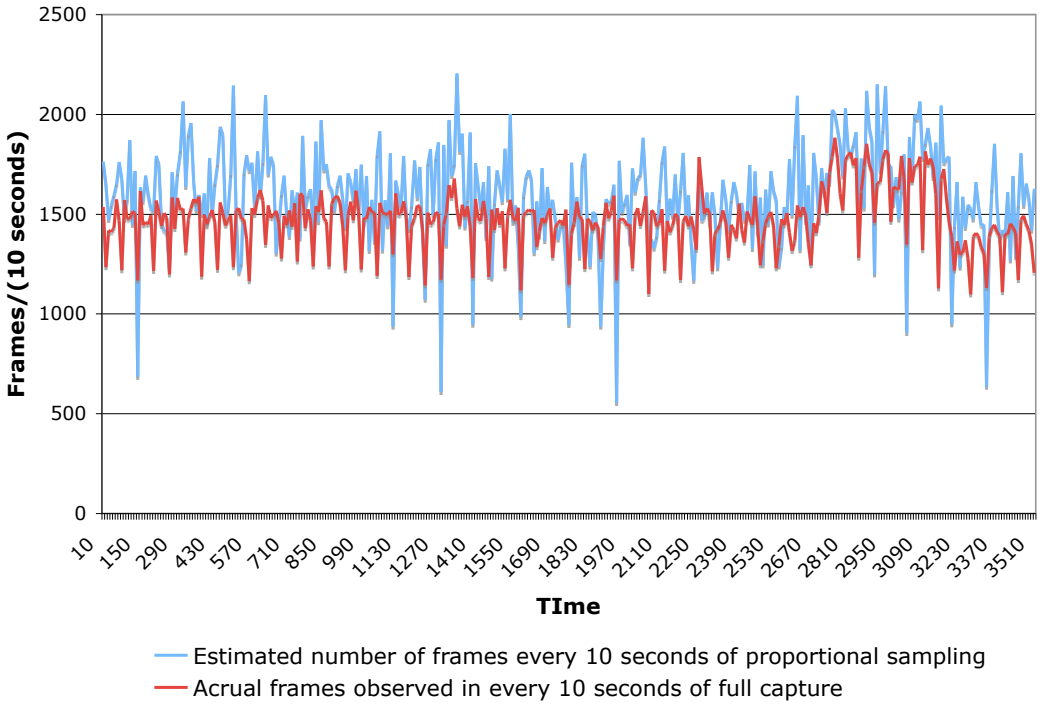


Figure 6.8: The estimated number of frames on channel 1 captured by proportional sampling every 10 seconds track the number of frames captured by the simultaneous full trace on channel 1.

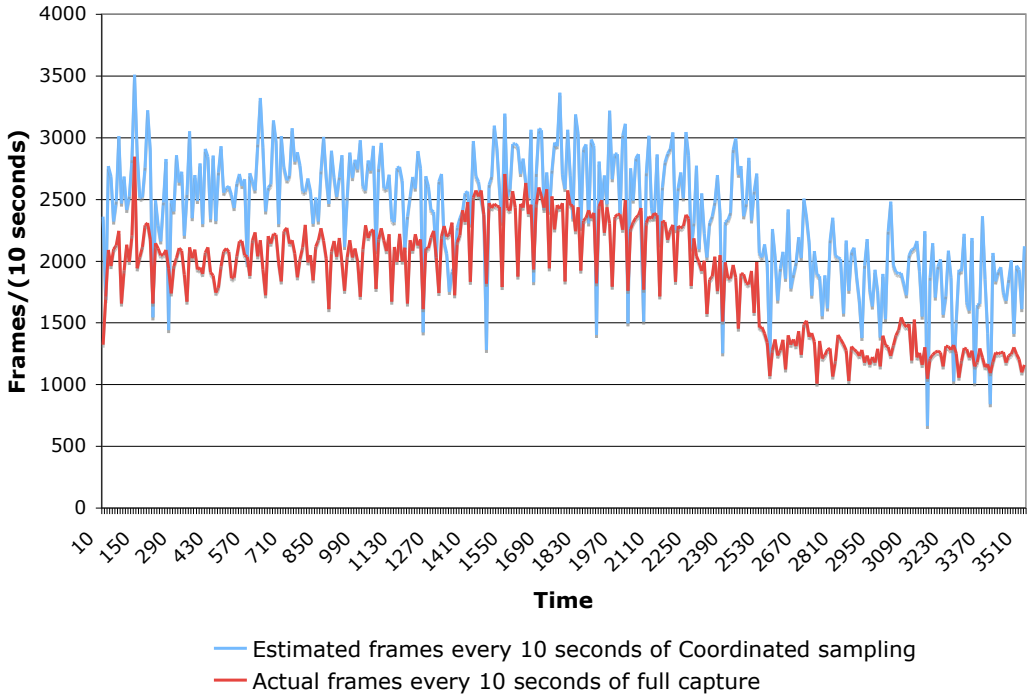


Figure 6.9: The estimated number of frames on channel 1 captured by coordinated sampling every 10 seconds track the number of frames captured by the simultaneous full trace on channel 1.

interface (wifi1) on our AMs consistently captured more traffic than the other (wifi0). The reason for difference this might be in software (priority on the bus) or it might be that one radio is more sensitive than the other. However, this phenomenon was consistent across all our AMs and in all our experiments. In Figure 6.10 we show the result of an experiment where we performed full capture on both interfaces of one AM on channel 1. We can see that there was a consistently higher capture using wifi1 than wifi0.

We then calculated the Kullback-Leibler (KL) divergence. Table 6.3 also shows that the value of the KL divergence was the highest for Equal sampling against full capture. This means that the probability distribution constructed using the hour-long equal sampling data is the furthest from the distribution of the corresponding hour's full capture data. You can see in the table that the KL divergence metric was better for Proportional sampling and was best for Coordinated sampling. In contrast, the distribution of the Equal sampling data was different from the distribution of the Coordinated sampling data and the Full capture from the first week was different from the Full Capture from the second week. We believe that because Proportional collects a larger sample than Equal and Coordinated collects a larger sample (both by spending a greater amount of time on the channel under consideration) than Proportional sampling, the respective estimates of frames scaled up to 10 seconds tend to match the full captures more closely.

6.3.2 Inter-Clustering Distance

In this section, we present results from the experiments described in Section 6.1.1. We can see that the axes in Figures 6.11–6.18 are labeled "Deauthentication", "RTS", "Data": the triple of common frame subtypes that had the highest COV. The different colors (or shades

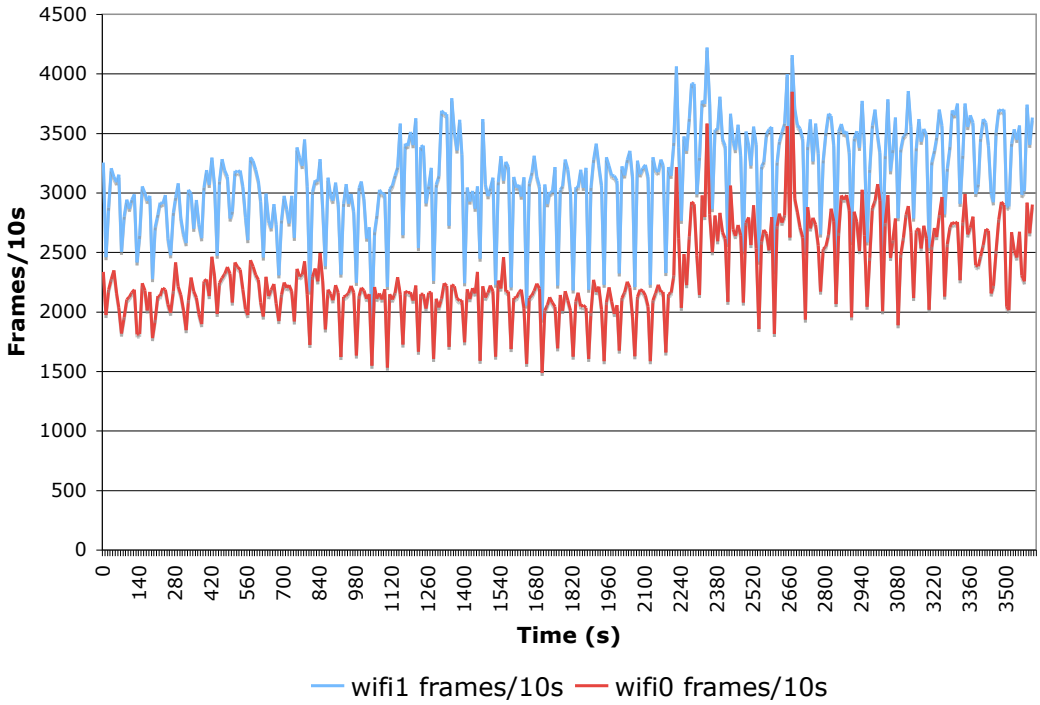


Figure 6.10: The number of frames captured by the interface "wifi0" in every 10 seconds interval on the AP70 is consistently less than the number of frames captured by the interface "wifi1" when both interfaces capture traffic continuously on channel 1.

Table 6.3: Kullback-Leibler Divergence (Section 6.1.1)

Strategies	How different?
Equal Sampling v. Full Capture (Figure 6.4)	0.531
Proportional Sampling v. Full Capture (Figure 6.5)	0.387
Coordinated Sampling v. Full Capture (Figure 6.6)	0.221
Equal Sampling (Week 1) v. Coordinated Sampling (Week 2)	1.395
Full Capture (Week 1) v. Full Capture (Week 2)	0.490

of grey if printed using a black and white printer) for the points in the graphs represent the clusters that the K-means clustering algorithm assigned to them. The points are all normalized to be unit vectors.

Figures 6.11– 6.14 show that there is an outlying data point that has high "Deauthentication" value. On the other hand, in Figures 6.15–6.18 the highest values for the Deauthentication coordinate were much lower. We looked deeper at the data and we noticed that there was a period of a few hours early in the first week that there were a high number of deauthentication frames being transmitted. (It is possible that there was a Denial of Service attack in progress.)

Interestingly, the outlier validates our process of selecting the Deauthentication feature from the vector of counts of subtypes, to some extent. The sudden spike in Deauthentication frames (two hours long in the whole week) caused the increase in the variance in the Deauthentication feature and, since we use the COV for picking frames, we chose this feature. As we see in Table 6.4 (Table 6.4 shows the results of the distance metric described in Section 6.1.1; the lower the value of the distance metric, the "closer" the clusterings are to each other. The value is the lowest for the clustering of coordinated sampling vs. the clustering of corresponding full capture.) our distance metric indicated that the data from the two weeks was "different". The fact that there was a spike in the Deauthentication coordinate for two of the points in the first week would contribute substantially to the difference.

Our hypothesis is that the reason for a lower distance between the clusterings of coordinated sampling and full capture is that coordinated sampling captures a greater number of unique frames and therefore a *better* sample.

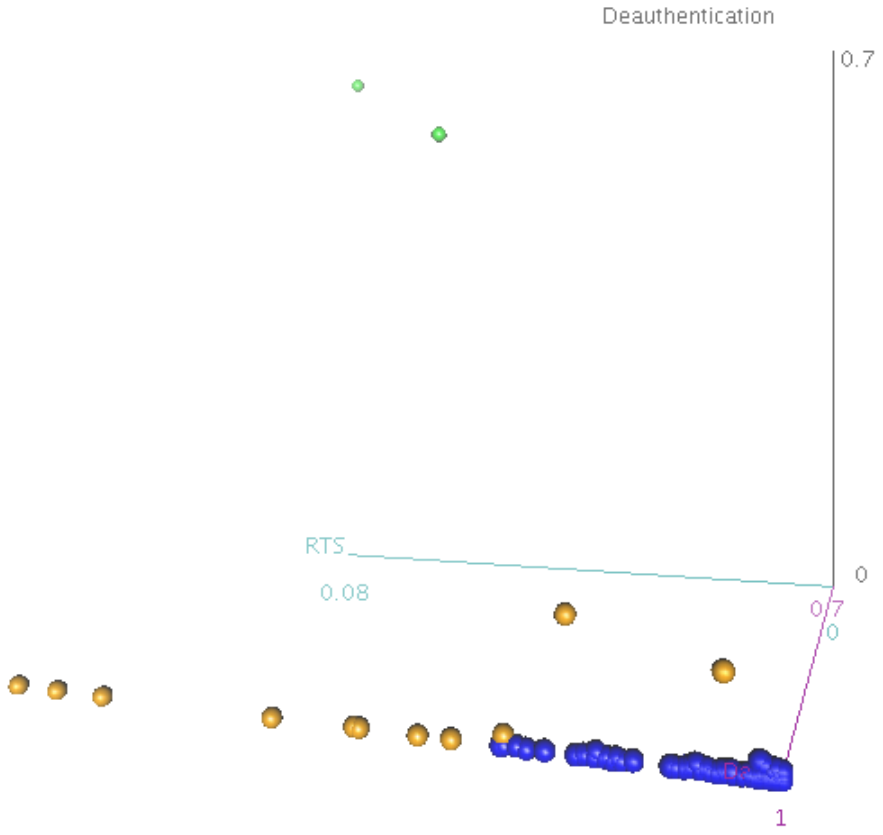


Figure 6.11: K-means clustering of hour-long traces of a week of traffic captured using Equal sampling on channel 1 (Week 1).

Table 6.4: Inter-cluster distance as described in section 6.1.1. In the table, the left-hand distribution corresponds to "P" and the right hand distribution corresponds to "Q" from the equation described in Section 6.1.1.

Strategies	How different?
Equal Sampling v. Full Capture (both Week 1)	0.223
Proportional Sampling v. Full Capture (both Week 1)	0.376
Proportional Sampling v. Full Capture (both Week 2)	0.204
Coordinated Sampling v. Full Capture (both Week 2)	0.038
Proportional Sampling (Week 1) v. Proportional Sampling (Week 2)	0.606
Full Capture (Week 1) v. Full Capture (Week 2)	0.921

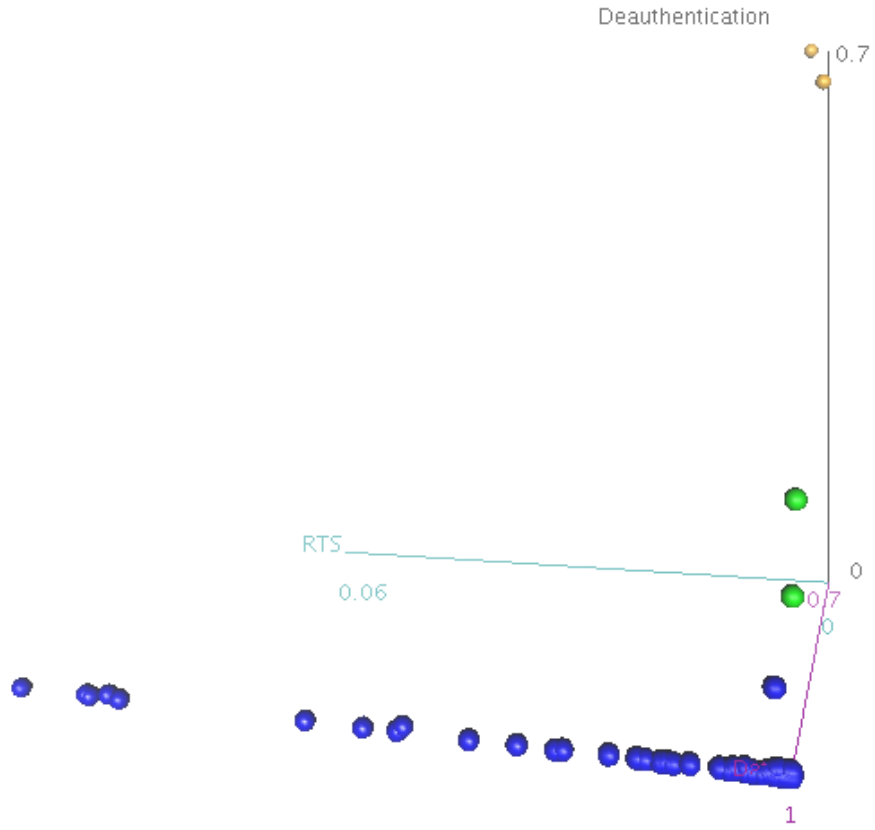


Figure 6.12: K-means clustering of hour-long traces of a week of traffic captured using full capture on channel 1 on the same AMs as the Equal sampling (Week 1).

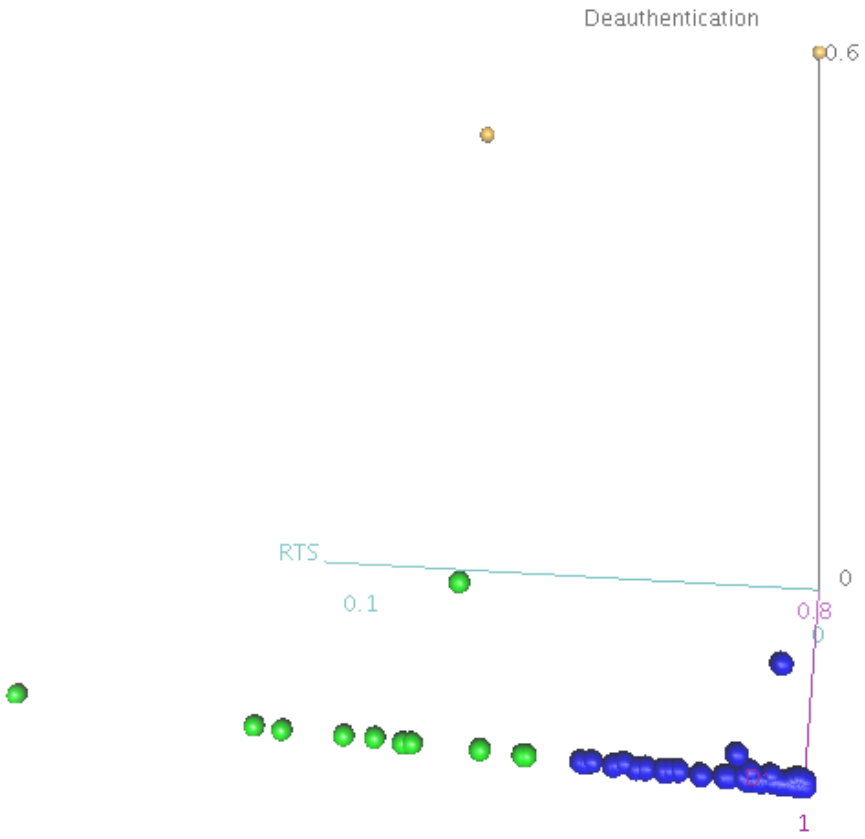


Figure 6.13: K-means clustering of hour-long traces of a week of traffic captured using proportional sampling on channel 1 (Week 1).

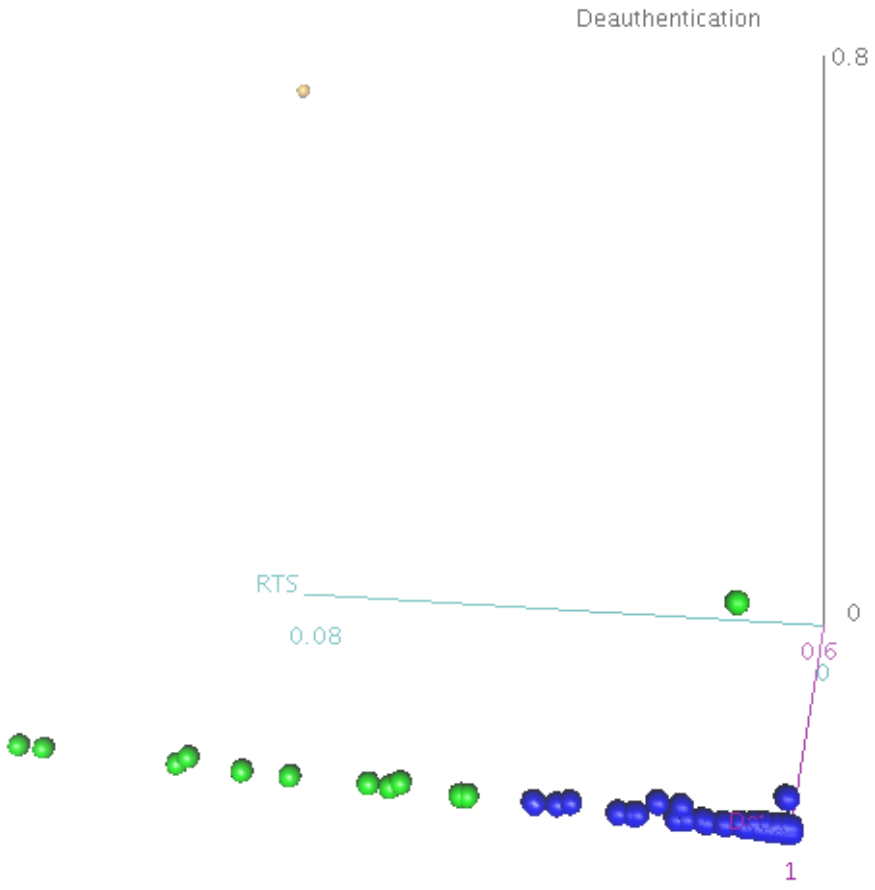


Figure 6.14: K-means clustering of hour-long traces of a week of traffic captured using full capture on channel 1 on the same AMs as the proportional sampling (Week 1).

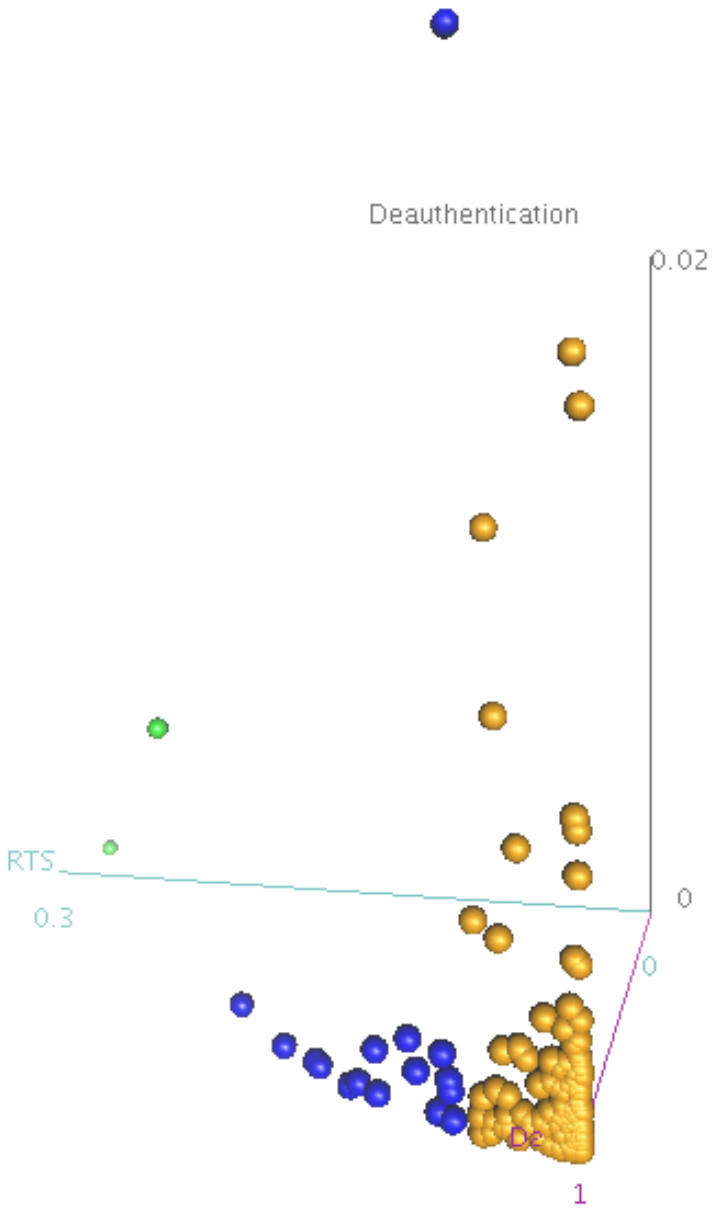


Figure 6.15: K-means clustering of hour-long traces of a week of traffic captured using proportional sampling on channel 1 (Week 2).

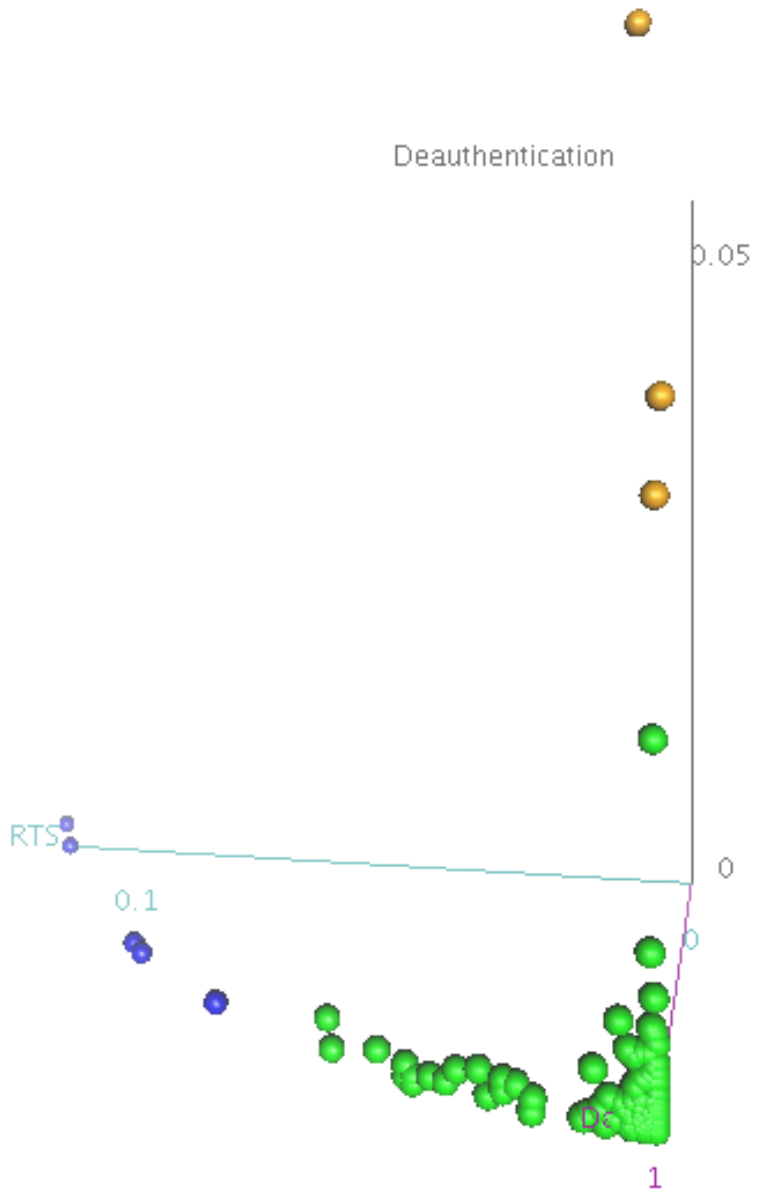


Figure 6.16: K-means clustering of hour-long traces of a week of traffic captured using full capture on channel 1 on the same AMs as the proportional sampling (Week 2).

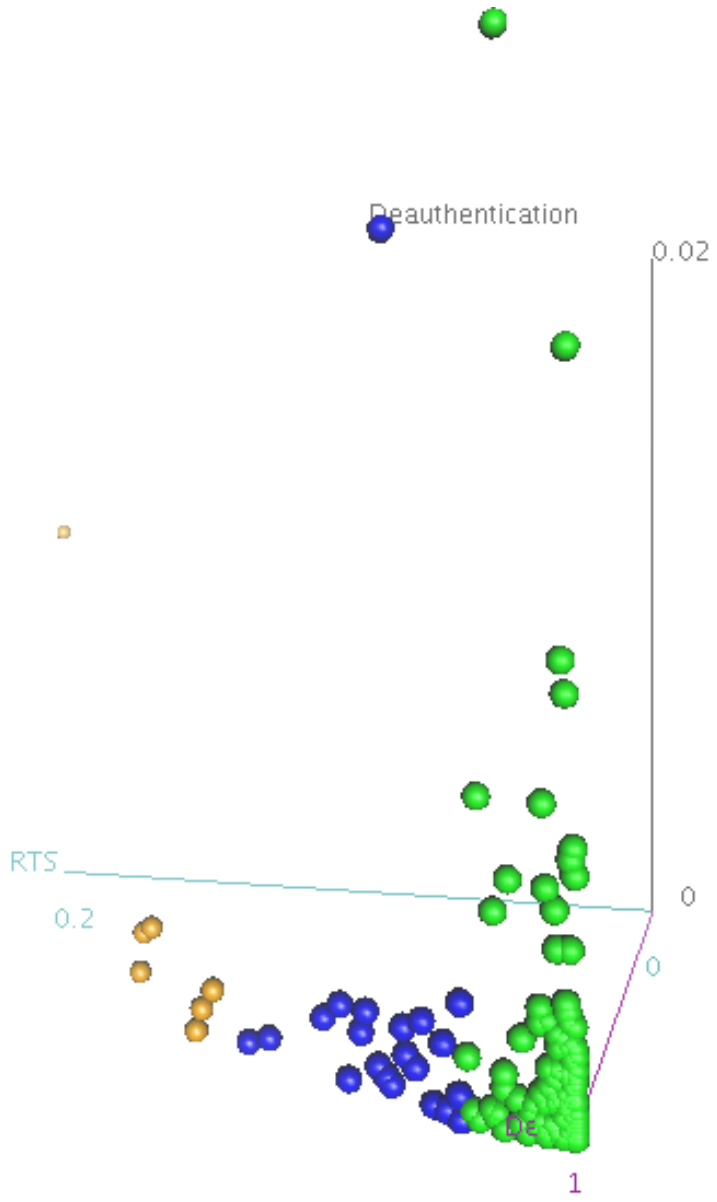


Figure 6.17: K-means clustering of hour-long traces of a week of traffic captured using coordinated sampling on channel 1 (Week 2).

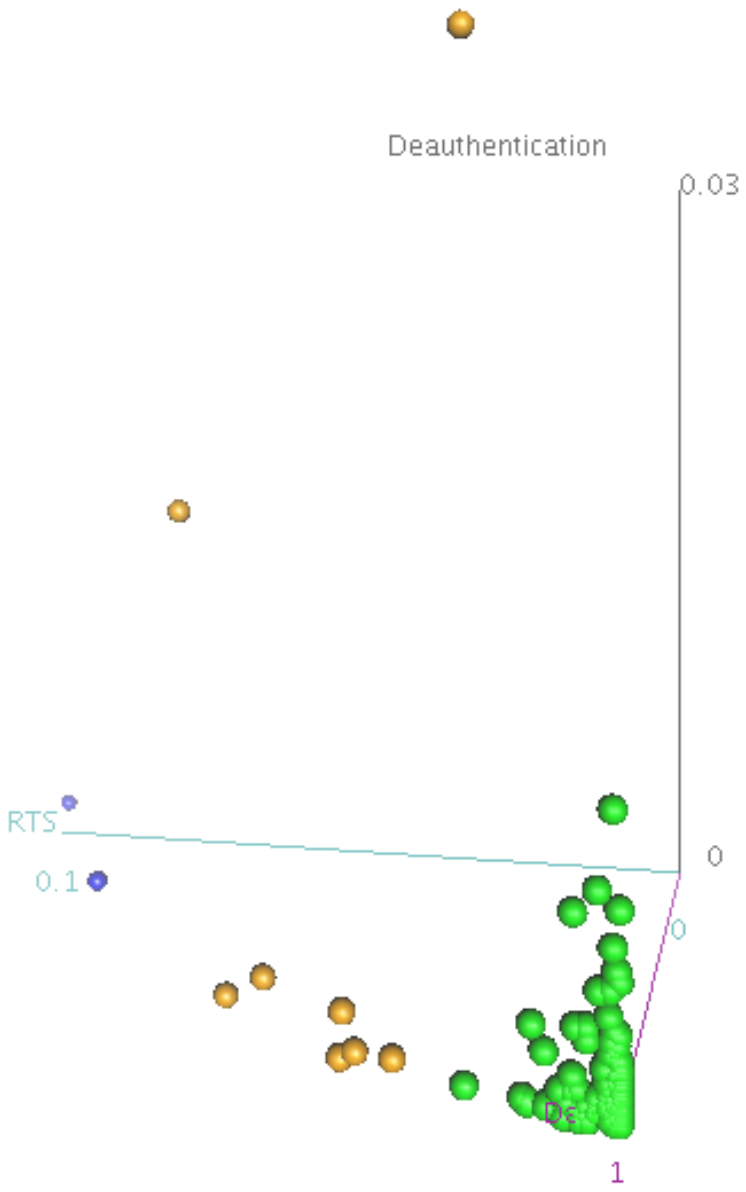


Figure 6.18: K-means clustering of hour-long traces of a week of traffic captured using full capture on channel 1 on the same AMs as the coordinated sampling (Week 2).

This hypothesis is not borne out, however, as we can see that the value of the distance metric between proportional sampling and full capture in the first week is greater than the value of the distance metric between equal sampling and full capture in the first week. This would indicate that either the sample captured using equal sampling was better than the sample captured using proportional sampling *or* that the distance metric is not a good measure of sample representativeness. We looked deeper into the data and counted the number of unique frames captured using each of equal sampling, proportional sampling (from both weeks) and coordinated sampling. However, the proportional sampling strategy captured more frames on channel 1 than equal sampling did. Therefore it seems likely that the distance metric is somehow lacking.

To try to shed some light on this, we further investigated by finding the sum of the distances between the individual points (the point representing each hour from the proportional sampling and the full capture *and* the same for the equal sampling and the full capture). In this distance metric as well, the proportional sampled trace was further away from the full capture than the equal trace was from full capture.

In contrast, the distance between the clustering of the first week's full capture and the clustering of the second week's full capture is high. This difference implies that the data from the first week was substantially different from the data of the second week. This is intuitive as one would expect the traffic patterns to be different in two different weeks.

6.4 Conclusion

Any monitoring system implicitly or explicitly samples the set of frames being transmitted, across time and space. We applied the Kullback-Leibler divergence computation for computing the divergence between fine granularity summaries of traces. The results for the experiments indicate that this metric was applicable and indicates that sampling performs well in capturing a representative subset (Equal, Proportional and Coordinated sampling in increasing order of representativeness).

Our "cluster matching" distance metric also indicates that using clusters formed using the K-means clustering algorithm on the coarser granularity triples have potential to be useful in representing traces (the distance between the clusterings of sampled traffic and full capture on channel 1 was less than the distance between the clusterings of traffic captured in two separate weeks). However, the counter-intuitive result in the table indicates that further work needs to be done to improve the metric in some way so that it distinguishes between traces more reliably.

CHAPTER 7

SUMMARY

Wi-Fi networks are ubiquitous and are widely accepted in the marketplace. On the other hand, there are several vulnerabilities inherent in the 802.11 protocol. The wide acceptance of 802.11 and the host of technologies and applications that it enables makes increasing numbers of devices and their data vulnerable to the problems of the protocol. Therefore, it is also widely accepted that wireless monitoring is needed.

In this thesis we describe the possible architectures of systems for monitoring 802.11 networks. There are various methods for capturing traffic to and from the wireless network. Wired sniffers can capture traffic at the APs, or on the switches. Servers can receive periodic SNMP, syslog or other event or summary data from the APs. Or, we can use radios to capture traffic directly off the air. The latter is the most challenging task, but it is necessary as some data can only be gathered using wireless sniffers, also known as Air Monitors or "AMs".

In this thesis, we explored 802.11 monitoring using AMs. In the first two chapters, we make the case for sampling. We discuss the large number of channels available in 802.11. With the diversity of protocols and channels available, it is difficult to capture all the transmissions in the air, at every location, on every channel, all the time. If it is possible, it is surely infeasible for most enterprises. Therefore, we recognize that sampling is an integral part of 802.11 monitoring. We designed, implemented and evaluated several

channel sampling techniques in the context of a building-wide deployment of AMs that are being used by a large number of users. The number of AMs in the deployment are far fewer than what would be needed for full coverage of the network.

We describe "coverage" of wireless sniffers, and the challenges that are involved in capturing traffic off the air. We point out that the coverage of AMs is not only geographic location, but also time and frequency that the AM monitors.

The wireless medium is dynamic and as the environment changes the capture strategies need to adapt to these changes. In the third chapter, we introduce two sampling strategies. One is static (Equal) and the other is dynamic (Proportional). Both strategies scan through the channels periodically. The proportional strategy spends time on the channels that is proportional to the frame rate observed on those channels in the previous sampling cycle. The equal strategy does not change its behavior from cycle to cycle. The proportional strategy is used when the user of the monitoring system believes that the channels with a higher frame rate are more "important". The equal strategy is used when the user of the system believes that all channels are equally important. We introduce "dingo" — our channel-sampling software that implements these two strategies. The "*amsniffer*" component of dingo is installed on the AMs and it captures traffic from the air. We show that the proportional strategy captured more traffic and enables the detection of more attacks than the equal strategy. We also discuss how we can implement other such strategies that are proportional to the observed traffic. For example, we mention the Client/Proportional and the BSSID/Proportional strategies in which the channels with either the highest number of Clients or the highest number of BSSIDs are the most important. One possible extension of the proportional strategy that we have not tried is to use first order or second order

derivatives to estimate the proportions. Also, we could use a weighted moving average of proportions to estimate the "importance."

In the fourth chapter, we introduce the concept of *focus* in the context of a monitoring system. The focus of the monitoring system is the subset of traffic in the air that the consumers of the traffic are most interested in. We provide a mechanism for consumers to request a particular focus and (if needed) to *refocus* the monitoring system. The "*amcontroller*" receives these requests and forwards them to the AMs. These requests take the form of predicates. An example predicate is "src==aa:bb:cc:dd:ee:ff", which is matched on the AM against every frame seen by the AM. The *amsniffer* software maintains an array of counts of matching frames, one element of the array for each channel being monitored. The count is reset before the beginning of every cycle. The sniffer then spends time on the various channels in proportion to the counts observed in the previous cycle. We deployed this scheme in our testbed and conducted experiments to determine its usefulness. The scheme was successful in capturing far more frames matching the predicate supplied than the "non-refocused" scheme.

In the fifth chapter, we introduce schemes to make the capture of traffic more efficient. In a densely deployed network, there are areas where the coverage of AMs necessarily overlaps with neighboring AMs. The proportional strategies exacerbate this problem because AMs usually hear similar traffic proportions on the channels as their neighbors and therefore tend to spend more time on the same channels as their neighbors. We define two AMs that have observed the same frame in the recent past as *neighbors*. As all the AMs send traffic to the merger, so we can extract the neighbor relationships using the merger. We used this neighbor information to coordinate the schedules of the AMs in the system

to minimize the time that neighboring AMs spend on the same channel. We developed and implemented a "coordinated scheduling" algorithm for this purpose. This algorithm is implemented in the *amcontroller*. The *amcontroller*, using traffic from the merger, extracts the neighbor relationships and creates schedules for the AMs to follow. We ran experiments to compare the traffic capture using the independent proportional strategy to the coordinated scheme. Our results show that the coordinated strategy was successful in reducing the number of frames that were captured redundantly at multiple AMs and therefore in increasing the volume of captured traffic.

All the schemes mentioned above sample wireless traffic. The only reason that we sample traffic is because we cannot capture all the frames transmitted into the air. The question, therefore, is "How representative is the sample being collected of the full trace?" In the sixth chapter, we introduce two metrics that we use to compare traces. Using the first of these metrics, we demonstrated that the sampled traces were indeed close to the full-capture trace. This result supports our position that our sampling schemes can be used in place of expensive deployments that attempt to cover every channel all the time at every location, especially when cost is an issue. There was a counter-intuitive result when we used the second metric and we believe that more work needs to be done to improve the usefulness of that metric. We also propose that these metrics can be used in comparing traces in other situations, such as when choosing from a database of traces.

Thus, we have introduced various channel-sampling schemes, and we have validated their efficacy in capturing traffic that is cognizant of the demands of the consumers of the traffic and also representative of the characteristics of the traffic in the air.

Therefore, our contributions in this thesis are as follows:

- We developed and studied the two basic techniques of *Equal* and *Proportional* sampling.
- We introduced and the concept of *Refocusing* and developed software for it that allows consumers of monitored traffic to request a *focus* that is most suitable to their needs.
- We introduced and implemented smart *coordination* among AMs so that the monitoring system can effectively monitor the space of $\{\text{frequency} \times \text{space} \times \text{time}\}$ with limited hardware.
- We used two metrics to compare 802.11 traces captured using various sampling techniques to traces captured using full capture on a single channel.

7.1 Future Work

There are some extensions of our work that we believe would be useful to the community.

Large-scale deployment. The next step for our project is to deploy the sniffing infrastructure in multiple buildings across the campus of the college. There are many scalability questions that could be asked in this stage of the project. The various components and schemes in the infrastructure would be stress-tested in this exercise.

- Can dingo's *amcontroller* keep up with controlling the increasing number of AMs? The coordination algorithm will need to coordinate a greater number of schedules.
- Can the merger scale as greater number of streams of traffic will flow into it?

It may be necessary to extend the architecture, possibly involving a hierarchy of mergers, to deal with these issues of scale.

Spatial sampling. As mentioned above, a given AM placement may not always capture the best (in terms of volume) sample of traffic over a long period due to the dynamic nature of the wireless environment. Placing monitors in every square inch of space is not feasible. We can get nowhere near perfect coverage in terms of space.

In this thesis, we describe "channel sampling" as a technique for dynamic allocation of channels. Similarly, we can conceive of "spatial sampling" for dynamic positioning of an AM. An AM will be placed in a location to collect frames for some period, and then it will be moved to another location. We could cycle through the predetermined locations available to the AP, while pausing for periods to be determined dynamically according to the observed metrics. As space is not discrete (unlike channels), an AM cannot pause at every possible location. The pause could be longer for locations observed to be more *busy* and shorter for *idle* locations, resulting in proportional spatial sampling.

Trace Comparison. In Chapter 6 we note that the clustering based distance metric needs deeper study. The counter-intuitive nature of the result indicates that the parameterization may be incompletely explored or that the distance metric itself is flawed or incomplete. The space of various features of the trace is large and so are the possible variations of parameters of the clustering techniques used. Further, there are several choices in the clustering technique used.

Trace-comparison techniques have applications beyond merely estimating the quality of sampling. More precise trace-comparison techniques can enable a host of applications.

For example, if a researcher wishes to pick traces from a database that are similar to traces or different from traces that he already has, a trace comparison technique would be handy. Even further, we can imagine that precise representation of traces can enable trace *generation* according to a set of parameters that can be specified to match the type of trace that is needed.

Mobility. As we have mentioned earlier, mobility will affect sampling and its performance. Our campus is non-mobile [36], therefore, our experiments did not test the interaction of mobility and sampling to any great extent. Further study would be useful to examine this interaction.

The dynamic sampling strategies depend on the observations in one cycle to predict "importance" of channels in future cycles. A low cycle time would be useful in reacting quickly to mobile clients. However, there is a down-side to this approach. The quicker the AMs switch channels, the higher the overhead from changing channels. There is, in the end a tradeoff between short cycle times and overhead from changing channels. With improving sniffer hardware and software, this tradeoff will become easier to manage as the channel switching time is likely to reduce.

APPENDIX

I am grateful to Bennet Vance for carefully designing and implementing the Merger component of the system.

Merging

The role of the merger is to combine the packet streams from multiple sniffers to produce a single coherent picture of the traffic in the air.

Figure 1 shows the motivation for merging. Five 802.11 frames (labeled *x* to *x+4*) are transmitted. Although there exist two sniffers in the vicinity of the transmitter of these frames, suppose that neither sniffer is able to capture all five frames (due to reflections, noise and so forth). Each sniffer only hears a subset of the frames. The purpose of the merger is to reconstruct the actual traffic based on these incomplete packet captures. Note that our merger cannot reconstruct all traffic: if a frame is not heard at any sniffer, the merger does not attempt to insert an artificial frame.

To merge captured frame streams we use frame timestamps. When a frame is captured at a sniffer it is timestamped with the sniffer's clock. But as the sniffers' clocks are not perfectly synchronized, frames received at multiple sniffers typically receive different timestamps at each sniffer. One of the merger's main tasks is to resolve these discrepancies by applying corrections to the sniffers' timestamps. In the following we discuss the merger's overall architecture, its timestamp synchronization and clock correction mechanism, and efficiency considerations in its implementation.

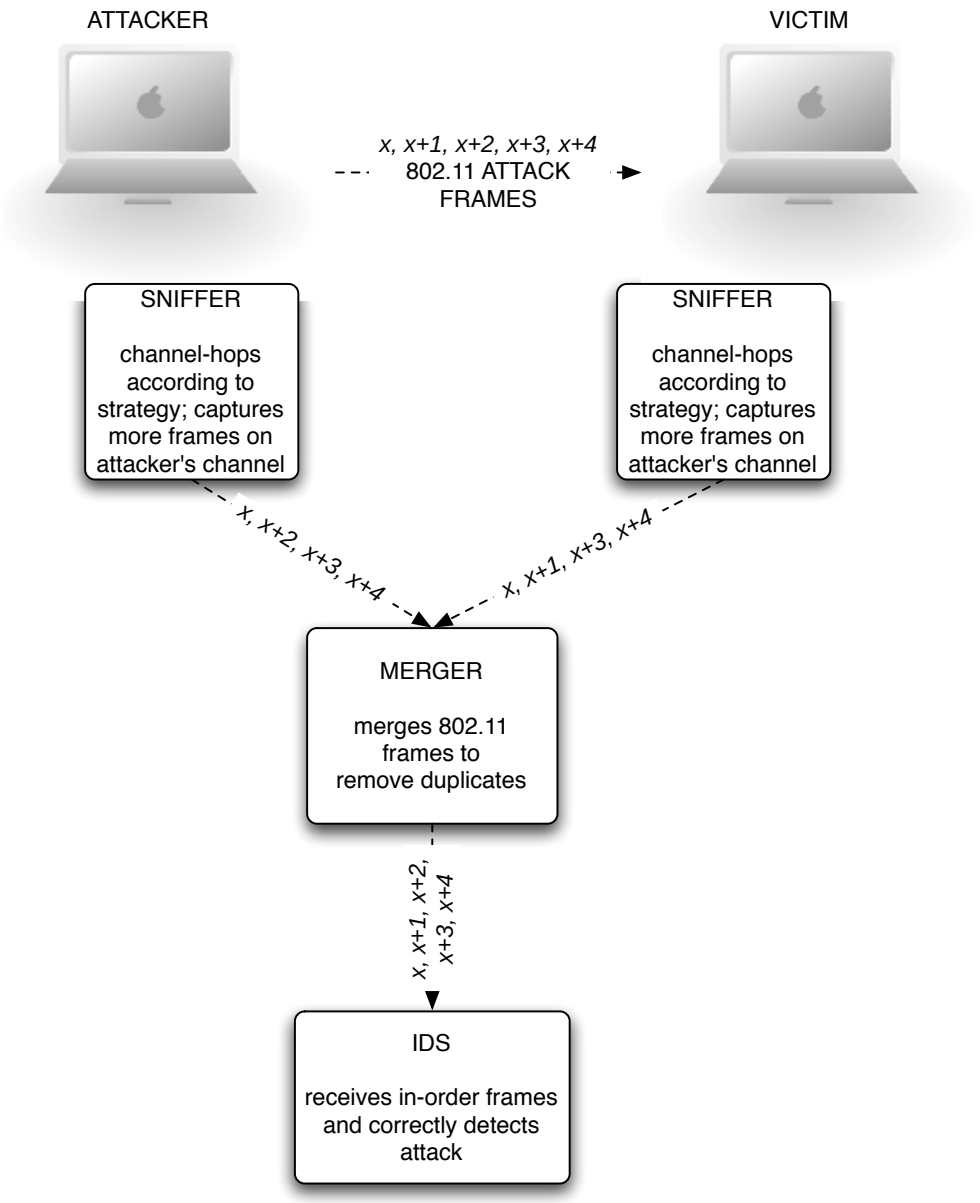


Figure 1: Our system uses sampling and careful merging.

We note that the standard network time protocol (NTP) is insufficient for synchronizing the sniffer's clocks, as it is unable to reduce the asynchrony to the necessary level (microseconds) (see below). So, although NTP may be used to initially synchronize sniffer clocks, we still need to compensate for ongoing drift when we merge the frame streams.

Merger Architecture

Figure 2 diagrams the merger's principal data structures and the manner in which data flows through them. The left-hand side of the diagram is the *input* side, while the right-hand side is the *output* side. The input side includes buffer space to hold incoming frames. The output side maintains one queue for each sniffer; the entries in these queues are not frames but *frame references*. Their order in the queue for a given sniffer reflects the order in which the corresponding frames were received at that sniffer: the oldest references appear near the lower-right corner of the diagram, and the most recent ones near the upper-right corner.

When the merger receives a frame from one of the sniffers, it proceeds to update its data structures as follows. First, it uses the frame's FCS (Frame Check Sequence) as a key into a hash table. The entries in the hash table are list anchors for lists of frames that share the same FCS. (Different frames rarely share the same FCS value.) The order of the frames within each list is determined by the frames' corrected timestamps, so the most recent frames can be found near the heads of the lists, and the most aged frames towards their tails. Once the list for the incoming frame's FCS has been found or created, the merger searches through the list for the frame (if any) whose corrected timestamp most closely agrees with that of the incoming frame. If the timestamps are in close enough agreement

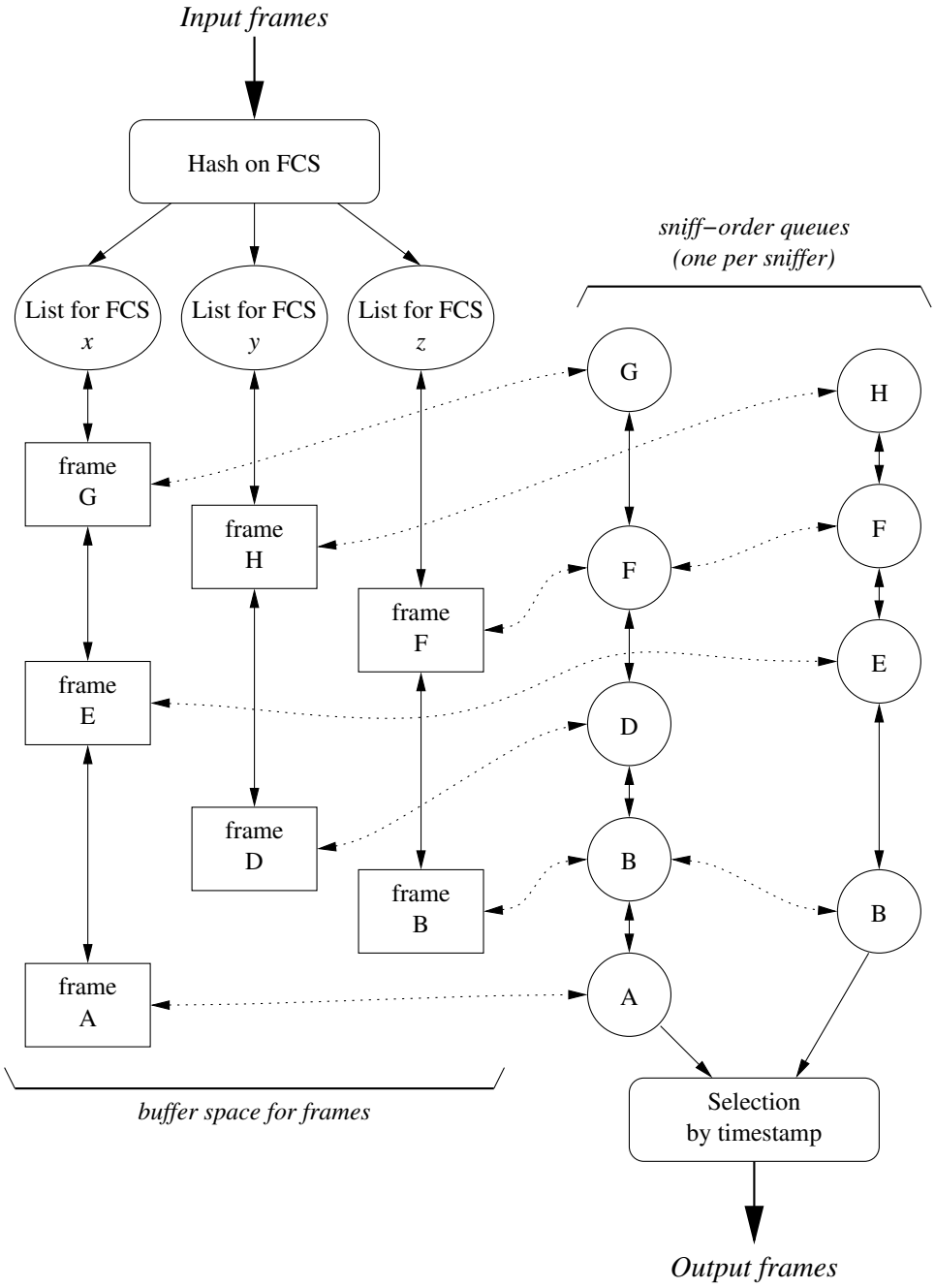


Figure 2: Merger data structures

(according to a defined degree of proximity), the incoming frame is considered to have already have been heard by another sniffer and is deleted.[1] Otherwise the incoming frame is inserted into the list at the appropriate position. In either case, the merger appends a frame reference to the appropriate queue as illustrated on the output side of the diagram; this reference ensures that the frame's position in the output stream will be consistent with the other frames heard by the same sniffer(s). Bidirectional horizontal links connect the entries in the output-side queues with the frames they reference.

The merger's output control is illustrated in the lower-right corner of the diagram. The heads of the sniff-order queues are funneled into a *selection* component that is responsible for identifying the oldest entry at any of the n queues. The merger periodically polls the selected entry to see whether it is ready for output. A frame is ready for output if its age exceeds a defined threshold, that is, if it has been queued long enough to be confident that no preceding frame will arrive from another sniffer.

Synchronization and Clock Correction

For the merger to assess correctly the ordering and spacing of frames received at one sniffer but not another, the merger depends on accurate relative timing information from the different sniffers. Even when they are synchronized by NTP, however, we have found that clocks at different sniffers can vary substantially (often by milliseconds) and require correction. Moreover, the differences between clocks drift over time, often by many microseconds per second, so the clock corrections need to be updated frequently to maintain consistency among the different sniffers' timestamps.

[1] We do not compare the frame contents; there is only a tiny chance that two different frames arrive at the same time and have the same FCS.

To synchronize sniffer clocks, we associate a *clock correction* variable c_i with each sniffer S_i. When we receive a frame from sniffer S_i with *raw* (uncorrected) timestamp r_i, we compute a corrected timestamp $t_i = r_i + c_i$ that for most purposes we use in place of the raw timestamp. If a frame is received at multiple sniffers, its imputed timestamp—the timestamp attached to the outputted frame—is the average of the corrected timestamps t_i from all the sniffers.

Following Yeo et al. [72] we use beacon frames as synchronization events. Suppose a given beacon is heard by both S_i and S_j, and that S_i gives it raw timestamp r_i while S_j gives it raw timestamp r_j. The corrected timestamps t_i and t_j are computed as described above. Since wireless frames are transmitted over the air at the speed of light, the actual times at which the beacon is received by S_i and S_j must be nearly identical, and so t_i and t_j should be nearly identical as well. To align the corrected times we augment c_i by $(t_j - t_i)/2$ and c_j by $(t_i - t_j)/2$. The revised clock corrections, when added to r_i and r_j, respectively, result in new (equal) values for t_i and t_j.

This simple clock-correction adjustment strategy is adequate for keeping up with gradual drift among sniffer clocks on the order of microseconds per second. It may be inadequate if sniffer clocks are subject to sudden jumps, as might occur if they are periodically synchronized by a central switch, for example. While we have considered various mechanisms for dealing with large clock jumps, we do not describe these here.

AMEX

To reduce the volume of forwarded traffic while retaining the relevant information from each individual frame, AMs forward information in a compressed frame format that we call *AMEX* (for *AM EXtractor*). This format includes only the *interesting features* of each frame and packs the features for several frames into each UDP datagram sent from the AM to the merger.

Our AMEX encoding scheme allows us to redefine the set of interesting features dynamically to adapt to changing conditions. Thus, each AMEX frame includes a header (with a bitmap that indicates which AMEX features are present), data specific to the AM (e.g., the timestamp and signal strength of the received 802.11 frame), and the selected features from the 802.11 header (such as the source MAC address) or PHY-layer information provided by the 802.11 driver (such as the rate).

I am grateful to Joshua Wright for implementing the AM extraction modules.

BIBLIOGRAPHY

[1] Draft Joint Wireless Administrator Checklist, Defense Information Systems Agency, Department of Defense. http://iase.disa.mil/stigs/checklist/draft-joint-wireless-administrator-checklist-dec05.doc, Visited May 25, 2008.

[2] Guidelines for the Development and Evaluation of IEEE 802.11 Intrusion Detection Systems (IDS). http://www.nsa.gov/snac/wireless/I332-005R-2005.pdf, Visited May 25, 2008.

[3] IEEE 802.11g/D8.2, Draft Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band, (April 2003), IEEE 802.11 WG.

[4] IEEE Std. 802.11b, Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed Physical Layer Extension in the 2.4 GHz Band, IEEE Std. 802.11b-1999, (1999), IEEE.

[5] Kismet 802.11 network detector. http://www.kismetwireless.net/, Visited 25 May, 2008.

[6] Porcupine, Advanced Network Management Lab, Indiana University. http://www.porcupine.iu.edu/, Visited May 25, 2008.

[7] Snort-wireless 802.11 intrusion detection system. http://snort-wireless.org/, Visited May 25, 2008.

[8] Wireless Vulnerabilities & Exploits. http://www.wve.org/, Visited May 25, 2008.

[9] Atul Adya, Paramvir Bahl, Ranveer Chandra, and Lili Qiu. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proceedings of ACM MobiCom 2004*, pages 30–44, Philadelphia, PA, September 2004.

[10] Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill. Enhancing the security of corporate Wi-Fi ntworks using DAIR. In *Proceedings of MobiSys 2006*, pages 1–14, New York, NY, USA, June 2006. ACM Press.

[11] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of InfoCom 2006*, pages 775–784, March 2000.

[12] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of the 2002 ACM SIGMETRICS Conference*, pages 195–205, Marina Del Rey, CA, USA, June 2002. ACM Press.

[13] Magdalena Balazinska and Paul Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 303–316, San Francisco, CA, USA, May 2003. USENIX Association.

[14] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of the Twelfth USENIX Security Symposium*, pages 15–28, Washington, DC, USA, August 2003. USENIX Association.

[15] Michael W. Berry. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer, 2004.

[16] R. Beyah, S. Kangude, G. Yu, B. Strickland, and J. Copeland. Rogue access point detection using temporal traffic characteristics. *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 4:2271–2275 Vol.4, 29 Nov.-3 Dec. 2004.

[17] Sergey Bratus, Cory Cornelius, David Kotz, and Dan Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the First ACM Conference on Wireless Network Security (WiSec)*. ACM Press, March 2008.

[18] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security flaws in 802.11 data link protocols. *Communications of the ACM*, 46(5):35–39, 2003.

[19] Ranveer Chandra, Jitendra Padhye, Alec Wolman, and Brian Zill. A Location-Based Management System for Enterprise Wireless LANs. In *USENIX Symposium on Networked Systems Design and Impelementation*. USENIX, 2007.

[20] Ranveer Chandra, Venkata N. Padmanabhan, and Ming Zhang. WiFiProfiler: cooperative diagnosis in wireless LANs. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 205–219, New York, NY, USA, June 2006. ACM.

[21] Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Péter Benkö, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. pages 25–36, 2007.

[22] Yu-Chung Cheng, John Bellaro, Peter Benko, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proceedings of SIGCOMM 2006*, pages 39–50, Pisa, Italy, September 2006.

[23] Francisco Chinchilla, Mark Lindsey, and Maria Papadopouli. Analysis of wireless information locality and association patterns in a campus. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 906–917, Hong Kong, China, March 2004. IEEE press.

[24] John Cox. Researchers crafting intelligent, scaleable WLAN defense. *Network World*, December 2006. `http://www.networkworld.com/news/2006/120706-intelligent-scaleable-wlan-defense-darpa.html`, Visited May 25, 2008.

[25] D. Duchamp and N. Reynolds. Measured performance of a wireless LAN. In *Proceedings of the 17th Conference on Local Computer Networks*, pages 494–499. IEEE press, September 1992.

[26] John Edney and William A. Arbaugh. *Real 802.11 Security*. Addison Wesley, Reading, Massachusetts, July 2003.

[27] Jon Edney and William A. Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[28] John Fox, with contributions from Michael Ash, Theophilius Boye, Stefano Calza, Andy Chang, Philippe Grosjean, Richard Heiberger, G. Jay Kerns, Renaud Lancelot, Matthieu Lesnoff, Samir Messad, Martin Maechler, Duncan Murdoch, Erich Neuwirth, Dan Putler, Brian Ripley, Miroslav Ristic, , and Peter Wolf. *Rcmdr: R Commander*, 2008. R package version 1.3-14.

[29] Roy Furchgott. Phones for that other system. *The New York Times*, 26 October 2006. `http://www.nytimes.com/2006/10/26/technology/26basics.html`, Visited May 25, 2008.

[30] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[31] S. Garg and M. Kappes. Can i add a voip call? *Communications, 2003. ICC '03. IEEE International Conference on*, 2:779–783 vol.2, 11-15 May 2003.

[32] Fanglu Guo and Tzi cker Chiueh. Sequence number-based MAC address spoof detection. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, USA, September 2005.

[33] Matt Hamblen. Southwest and american test in-flight wi-fi. *The New York Times*, 23 January 2008. http://www.nytimes.com/idg/IDG_002570DE00740E18002573D9007495A3.html, Visited May 25, 200.

[34] Changhua He and John C. Mitchell. Security analysis and improvements for IEEE 802.11i. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, pages 90–110, San Diego, CA, USA, February 2005.

[35] Tristan Henderson and David Kotz. Measuring wireless LANs. In Rajeev Shorey et al., editor, *Mobile, Wireless and Sensor Networks: Technology, Applications Future Directions*, chapter 1, pages 5–27. John Wiley & Sons, New York, NY, 2006.

[36] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 187–201, Philadelphia, PA, USA, September 2004. ACM Press.

[37] Camden C. Ho, Krishna N. Ramachandran, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. A scalable framework for wireless network monitoring. In *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots (WMASH)*, pages 93–101. ACM Press, 2004.

[38] Peter Jackson and Isabelle Moulinier. *Natural Language Processing for Online Applications. Text Retrieval Extraction and Catagorization.* John Benjamins Publishing, 2004.

[39] Amit Jardosh, Krishna Ramachandran, Kevin Almeroth, and Elizabeth Belding-Royer. Understanding Link-Layer behavior in highly congested IEEE 802.11b wireless networks. In *Proceedings of the ACM SIGCOMM 2005 Workshop on experimental approaches to wireless network design and analysis (E-WIND-05)*, Philadelphia, PA, USA, August 2005.

[40] Amit P. Jardosh, Krishna N. Ramachandran, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. Understanding congestion in IEEE 802.11b wireless networks. In *Proceedings of the 2005 Internet Measurement Conference*, pages 279–292, Berkeley, CA, USA, October 2005.

[41] Mayank Kabra and Joseph A. Sarlo. Microwave Oven Interference with 802.11, UCSD CSE222A Term Project. March 2006.

[42] Minkyong Kim, David Kotz, and Songkuk Kim. Extracting a mobility model from real user traces. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, April 2006. IEEE Computer Society Press.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[44] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 107–118, September 2002. Revised and corrected as Dartmouth CS Technical Report TR2002-432.

[45] Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[46] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[47] Pradeep Kyasanur and Nitin H. Vaidya. Detection and handling of MAC layer misbehavior in wireless networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, San Francisco, CA, USA, June 2003.

[48] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[49] Uwe Ligges and Martin Mächler. Scatterplot3d - an R Package for Visualizing Multivariate Data. *Journal of Statistical Software*, 8(11):1–20, 2003.

[50] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the MAC-level behaviour of wireless networks in the wild. In *Proceedings of SIGCOMM 2006*, pages 75–86, Pisa, Italy, September 2006.

[51] B.S. Manoj and Alexandra Hubenko Baker. Communication challenges in emergency response. *Commun. ACM*, 50(3):51–53, 2007.

[52] E. Masala, C. F. Chiasserini, M. Meo, and J. C. De Martin. Real-time transmission of h.264 video over 802.11-based wireless ad hoc networks.

[53] A. Mishra, M. Shin, and W. Arbaugh. Context caching using neighbor graphs for fast handoffs in a wireless network, CS tech report number CS-TR-4477, University of Maryland, College Park, 2003.

[54] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *Proceedings of SIGCOMM, 2003*, 33(2):93–102, 2003.

[55] Slobodan Petrovic, Gonzalo Alvarez, Agustin Orfila, and Javier Carbo. Labelling clusters in an intrusion detection system using a combination of clustering evaluation techniques. In *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, page 129b, Washington, DC, USA, 2006. IEEE Computer Society.

[56] Marc Portoles-Comeras, Manuel Requena-Esteso, and Josep Mangues-Balfalluy. Multi-radio based active and passive wireless network measurements. In *Proceedings of the Second International Workshop on Wireless Network Measurement (WiNMee)*, Boston, Ma, April 2006. IEEE Computer Society Press.

[57] Marc Portoles-Comeras, Manuel Requena-Esteso, Josep Mangues-Balfalluy, and Marc Cardenete-Suriol. Monitoring wireless networks: performance assessment of sniffer architectures. In *Proceedings of the 2006 IEEE International Conference on Communications (ICC)*, pages 646–651, Istanbul, Turkey, June 2006.

[58] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[59] Ishwar Ramani and Stefan Savage. Syncscan: Practical fast handoff for 802.11 infrastructure networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE*

Computer and Communications Societies (INFOCOM'05), pages 675–684, Miami, FL, March 2005. IEEE.

[60] Maya Rodrig, Charles Reis, Ratul Mahajan, David Wetherall, and John Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *Proceeding of the 2005 ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis(E-WIND)*, pages 5–10, New York, NY, USA, 2005. ACM Press.

[61] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[62] Yong Sheng, Guanling Chen, Keren Tan, Udayan Deshpande, Bennet Vance, Hongda Yin, Chris McDonald, Tristan Henderson, David Kotz, Andrew Campbell, and Joshua Wright. MAP: A scalable monitoring system for dependable 802.11 wireless networks. Accepted to the IEEE Wireless Communications, December 2007.

[63] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs. In *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 191–204. ACM, 2006.

[64] Libo Song, Udayan Deshpande, Ulaş C. Kozat, David Kotz, and Ravi Jain. Predictability of WLAN mobility and its effects on bandwidth provisioning. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, April 2006. IEEE Computer Society Press.

[65] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1414–1424, March 2004.

[66] Diane Tang and Mary Baker. Analysis of a local-area wireless network. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–10, Boston, MA, USA, August 2000. ACM Press.

[67] Bob Tedeschi. Cutting the cord. *The New York Times*, 7 December 2003. `http://query.nytimes.com/gst/fullpage.html?res=9404E0DB1F3AF934A35751C1A9659C8B63`, Visited 25 May, 2008.

[68] Wei Wei, Kyoungwon Suh, Bing Wang, Yu Gu, Jim Kurose, and Don Towsley. Passive online rogue access point detection using sequential hypothesis testing with tcp ack-pairs. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 365–378, New York, NY, USA, 2007. ACM.

[69] J. Yeo, S. Banerjee, and A. Agrawala. Measuring traffic on the wireless medium: Experience and Pitfalls, 2002.

[70] Jihwang Yeo, Moustafa Youssef, and Ashok Agrawala. A framework for wireless LAN monitoring and its applications. In *Proceedings of the Third ACM Workshop on Wireless Security (WiSe'04)*, pages 70–79, Philadelphia, PA, USA, October 2004.

[71] Jihwang Yeo, Moustafa Youssef, Tristan Henderson, and Ashok Agrawala. An accurate technique for measuring the wireless side of wireless networks. In *Papers pre-*

sented at the 2005 Workshop on Wireless Traffic Measurements and Modeling (WiT-MeMo), pages 13–18, Berkeley, CA, USA, 2005. USENIX Association.

[72] Jihwang Yeo, Moustafa Youssef, Tristan Henderson, and Ashok Agrawala. An accurate technique for measuring the wireless side of wireless networks. In *Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling*, pages 13–18, Seattle, WA, USA, June 2005.

[73] Hongda Yin, Guanling Chen, and Jie Wang. Detecting protected layer-3 rogue APs. In *Proceedings of the IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, Raleigh, NC, September 2007.