

Refocusing in 802.11 Wireless Measurement

Udayan Deshpande¹, Chris McDonald², and David Kotz¹

¹ Institute for Security Technology Studies,

Department of Computer Science, Dartmouth College

² School of Computer Science and Software Engineering,

The University of Western Australia

Abstract. The edge of the Internet is increasingly wireless. To understand the Internet, one must understand the edge, and yet the measurement of wireless networks poses many new challenges. IEEE 802.11 networks support multiple wireless channels and any monitoring technique involves capturing traffic on each of these channels to gather a representative sample of frames from the network. We call this procedure *channel sampling*, in which each sniffer visits each channel periodically, resulting in a sample of the traffic on each of the channels.

This sampling approach may be sufficient, for example, for a system administrator or anomaly detection module to observe some unusual behavior in the network. Once an anomaly is detected, however, the administrator may require a more extensive traffic sample, or need to identify the location of an offending device.

We propose a method to allow measurement applications to dynamically modify the sampling strategy, *refocusing* the monitoring system to pay more attention to certain types of traffic than others. In this paper we show that refocusing is a necessary and promising new technique for wireless measurement.

1 Introduction

The new edge of the Internet is wireless. At Dartmouth College, all undergraduate students own wireless laptops, and take advantage of ubiquitous 802.11 coverage on campus. Most large enterprises provide wireless coverage for employee use. Many cities are deploying or considering large-scale municipal wireless-access networks. Although these networks were originally intended as a convenience, an overlay for the wired network, for many users and in many places the wireless network is of fundamental importance. To understand the edge of the Internet, we need effective wireless measurement.

Consequently, there are many motivations for monitoring wireless networks, including network management, security, and research. We consider situations where the network of wireless access points (APs) is augmented with interspersed wireless air monitors (AMs). These dedicated sniffers can provide real-time capture of wireless traffic and measurement of MAC-layer conditions for network analysis and management [6], and intrusion-detection systems (IDS) can analyze live streams of traffic from these AMs to monitor the network for attacks [2,9].

Wireless (802.11) networks allow traffic on multiple parallel channels, and yet all practical monitoring systems can listen to only one or two channels at a time. This approach is limited because there may be a need to monitor all the channels (e.g. to locate the presence of ad-hoc networks or rogue APs.) If there is only one channel to be monitored, the radio can simply monitor that channel continuously [6]. If no specific channel is desired, most scanning systems simply capture traffic on all channels with a predefined time spent on each channel [11]. Earlier work [2,6] acknowledges the need for smart channel-sampling strategies in security and management applications.

Our earlier work [9] demonstrates how to improve the capture by dynamically scheduling AMs to spend more time on channels with higher frame rates. In this work we extend our sampling philosophy by demonstrating a technique and framework that allows external applications— such as an administrator’s console, or an IDS— to dynamically instruct the AMs to put more effort into capturing traffic that meets a given condition.

We describe the traffic trace that a monitoring application requires at any time by its *focus*. An application can, of course, filter the stream of captured frames to suit its interests, but we want to allow the application to *refocus* the measurement system to skew its ongoing traffic capture towards this new focus, capturing more of the desired frames. We recognize that many important scenarios require the capture of a baseline sample, suitable for basic monitoring by multiple applications, and simultaneously a more focused sample(s) required by one or more applications.

For example, an application may be content with a traffic trace that consists of equal samples from each of the channels being monitored in the network. After observing some event, it may wish to *refocus* most of the sampling effort on the channels where a specific MAC address was observed. This application could be a WLAN intrusion-detection system, an application that displays locations of 802.11 devices around an office, or a system that monitors the quality of voice-over-wireless calls.

We claim that dynamic refocusing helps the wireless-network measurement system be more responsive to the needs of the subscribers of measured data. We describe a method and a tool that enables refocusing.

2 Related Work

We draw on related work in wireless measurement and 802.11 security. Few large-scale 802.11 measurement studies have attempted to capture wireless frames from the air. Although a few papers characterize traffic at meetings and conferences [10,13], none consider channel sampling or refocusing. To our knowledge, no commercial products provide refocusing, although some do allow channel sampling; for example, Aruba Networks [1], and Kismet [11]. Our own earlier work [9] focused on the challenge of sampling traffic from many channels, and merging frames from many AMs; in this paper we look at the problem of

refocusing through a large-scale experimental deployment. We compare one of the strategies used in the previous paper with our refocusing mechanism.

Security in 802.11 remains a challenge, because there are many vulnerabilities in the protocol and its implementations [2,4,14, for example]. We expect refocusing to help in capturing more information about an ongoing attack that is first detected during baseline sampling of a network.

A few recent papers describe offline tools to capture and merge wireless frames from multiple AMs located around a building [8,12,15]. 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. One, *Jigsaw* [8], requires four radios per location, clearly a more 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* [8], the authors place 39 monitoring “pods” around the building with four radios each. Each radio (AM) monitors a separate channel (Channels 1, 6, 11 and another “center” frequency). 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 measures the coverage in the scenario where only the AP or the client is in the range of a transmitting radio but not the nearby AM. 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, of course, important in a security scenario. 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 *Jigsaw* paper [7] that monitoring platforms from DAIR [2] and *Jigsaw* provide “the ability to observe every link-layer network transmission across location, frequency and time” is overly optimistic.

The DAIR system [2] uses USB NICs to turn an enterprise’s desktop computers into AMs, and could benefit from our sampling techniques for collection of traffic from production networks. The newer DAIR-based network management system [6] simply assigns the USB NICs to the channels of the nearby access points, missing important security-related traffic on non-production channels.

3 Dingo: A Coordinated Sniffer

We developed a set of software components, named *dingo*,¹ that collectively enable a variety of packet sampling policies to be defined and controlled, and

¹ A dingo is an Australian native dog renowned for its ability to track prey in bleak conditions.

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 1 shows the principal components of this software and the communication paths between them.

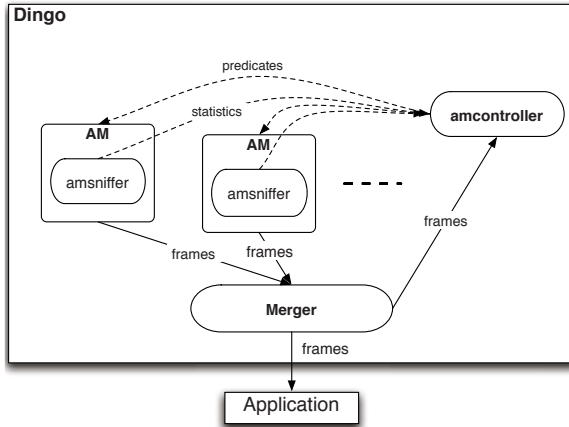


Fig. 1. The Sampling Architecture

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 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 (approx. 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.

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, amsniffer sends its counters to the amcontroller for consideration in future scheduling decisions. The range of policies described in our earlier work [9] 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 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.

4 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.

Filtering alone may not achieve the results needed by the application, because the necessary frames may not be available. Therefore, there is a need for online dynamic refocusing of the monitoring hardware.

We consider three classes of application.

Localization. If a WIFI 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 5.2.

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 $\geq x$ 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 [5]. If indeed they are vulnerable, we can refocus our sampling to more closely monitor them.

5 Results

We set out to investigate whether refocusing can be a valuable tool in wireless measurement systems. In this short paper, we do not have the space for a complete evaluation, but we seek to demonstrate the potential value of this approach.

5.1 Improved Volume of Capture

In our CS department, we deployed 19 Aruba AP70 AMs throughout the three floors of the building. The building also has 20 802.11a/b/g access points. The AP70 has a MIPS IDT32434 CPU running at 266MHz, 32MB DRAM, two Atheros AR5212 802.11a/b/g NICs (network interface controllers), two Ethernet NICs and one USB port. We installed OpenWRT Linux (Kamikaze branch, r5494) and Madwifi (v0.9.2) on each, and a copy of amsniffer on each. In this experiment, we only used one of the two wireless NICs on each AM.

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. In each experiment, the laptop was carried around a fixed path in the CS department building for a period of 10 minutes.

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".

Figure 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.

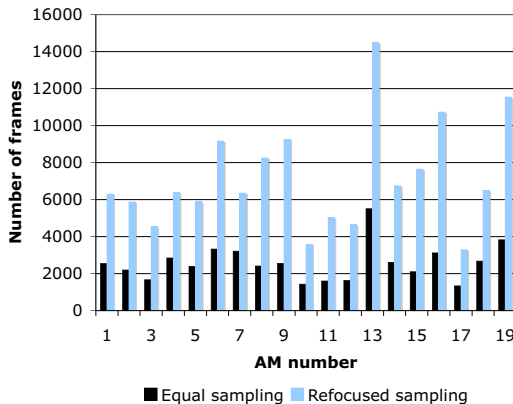


Fig. 2. Number of frames captured that matched predicate

In Figure 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.

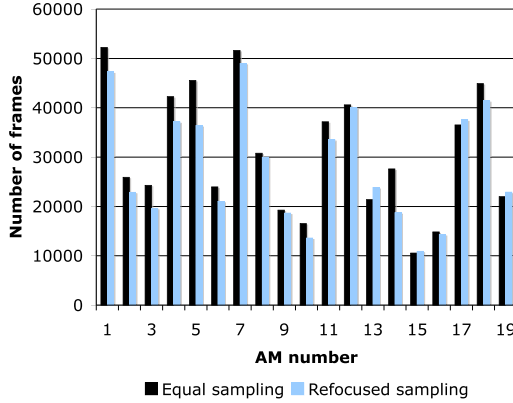


Fig. 3. Number of frames captured that did not match predicate

5.2 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. [3]. 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 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 are, on average, 1.95 feet more accurate than without refocusing.

6 Discussion

Given that the Internet edge is increasingly wireless, and the increasing number of channels in Wi-Fi (802.11n includes far more channels than 802.11b, for example), any researcher or network manager seeking to measure their network traffic must find efficient mechanisms to sample the network traffic. We propose a mechanism for applications to dynamically refocus the attention of the wireless-measurement infrastructure to capture more of a desired kind of network traffic.

Our experimental results indicate that refocusing was successful in capturing a greater number of frames matching the supplied predicate. Simultaneously, the capture of non-matching frames was not degraded substantially.

These preliminary results demonstrate the potential for our framework to be used in scenarios where stateless predicate matching is sufficient. There may, however, be more complex scenarios that require more state to be maintained. For example, our framework cannot currently express the desire to refocus on *newly arrived* clients, on those channels with an *increase* in some metric, or on channels with high jitter (inter-arrival times between frames) in a voice flow. Our next step is to extend the framework to be able to refocus on the basis of temporal changes like those.

Acknowledgments

We gratefully acknowledge the input and support of colleagues on the MAP team, and Dartmouth's network administrators, including Wayne Cripps and Tim Tregubov. The staff at Aruba Networks has been invaluable, including Nick DePetrillo and Mike Baker.

This research program is a part 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 authors and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Science and Technology Directorate. This project was also supported by the Cisco Systems University Research Program, the Center for Mobile Computing at Dartmouth College, and NSF Infrastructure Award EIA-9802068.

References

1. Aruba Networks Air Monitors, <http://www.arubanetworks.com/technology/air-monitors/>
2. Bahl, P., Chandra, R., Padhye, J., Ravindranath, L., Singh, M., Wolman, A., Zill, B.: Enhancing the security of corporate Wi-Fi networks using DAIR. In: Proceedings of MobiSys 2006, Uppsala, Sweden, June 2006, pp. 1–14 (2006)
3. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. In: Proceedings of InfoCom 2006, pp. 775–784 (2000)
4. Bellardo, J., Savage, S.: 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In: Proceedings of the Twelfth USENIX Security Symposium, Washington, DC, USA, August 2003, (USENIX Association), pp. 15–28 (2003)
5. Bratus, S., Cornelius, C., Kotz, D., Peebles, D.: Active behavioral fingerprinting of wireless devices. In: Proceedings of the First ACM Conference on Wireless Network Security (WiSec), March 2008, ACM Press, New York (accepted for publication, 2008)
6. Chandra, R., Padhye, J., Wolman, A., Zill, B.: A location-based management system for enterprise wireless LANs. In: Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2007), Cambridge, MA, USA (2007)

7. Cheng, Y.-C., Afanasyev, M., Verkaik, P., Benkö, P., Chiang, J., Snoeren, A.C., Savage, S., Voelker, G.M.: Automating cross-layer diagnosis of enterprise wireless networks. *SIGCOMM Comput. Commun. Rev.* 37(4), 25–36 (2007)
8. Cheng, Y.-C., Bellaro, J., Benko, P., Snoeren, A.C., Voelker, G.M., Savage, S.: Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In: *Proceedings of SIGCOMM 2006*, Pisa, Italy, September 2006, pp. 39–50 (2006)
9. Deshpande, U., Henderson, T., Kotz, D.: Channel sampling strategies for monitoring wireless networks. In: *Proceedings of the Second Workshop on Wireless Network Measurements, USA*, April 2006, IEEE Computer Society Press, Boston (2006)
10. Jardosh, A.P., Ramachandran, K.N., Almeroth, K.C., Belding-Royer, E.M.: Understanding congestion in IEEE 802.11b wireless networks. In: *Proceedings of the 2005 Internet Measurement Conference*, Berkeley, CA, USA, October 2005, pp. 279–292 (2005)
11. Kismet wireless sniffer, <http://www.kismetwireless.net>
12. Mahajan, R., Rodrig, M., Wetherall, D., Zahorjan, J.: Analyzing the MAC-level behavior of wireless networks in the wild. In: *Proceedings of SIGCOMM 2006*, Pisa, Italy, September 2006, pp. 75–86 (2006)
13. Rodrig, M., Reis, C., Mahajan, R., Wetherall, D., Zahorian, J.: Measurement-based characterization of 802.11 in a hotspot setting. In: *Proceedings of the ACM SIGCOMM 2005 Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND-2005)*, Philadelphia, PA, USA (August 2005)
14. Wireless vulnerabilities & exploits database, <http://wirelessve.org>
15. Yeo, J., Youssef, M., Henderson, T., Agrawala, A.: An accurate technique for measuring the wireless side of wireless networks. In: *Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling*, Seattle, WA, USA, June 2005, pp. 13–18 (2005)