Coordinated Sampling to Improve the Efficiency of Wireless Network Monitoring

Udayan Deshpande and David Kotz
Institute for Security Technology Studies
Dartmouth College
Hanover, New Hampshire, 03755, USA

Chris McDonald
Computer Science and Software Engineering
The University of Western Australia
Nedlands, 6009, Western Australia

***Abstract*—Wireless networks are deployed in home, university, business, military and hospital environments, and are increasingly used for mission-critical applications like VoIP or financial applications. Monitoring the health of these networks, whether it is for failure, coverage or attacks, is important in terms of security, connectivity, cost, and performance.**

Effective monitoring of wireless network traffic, using commodity hardware, is a challenging task due to the limitations of the hardware. IEEE 802.11 networks support multiple channels, and a wireless interface can monitor only a single channel at one time. Thus, capturing all frames passing an interface on all channels is an impossible task, and we need strategies to capture the most representative sample.

When a large geographic area is to be monitored, several monitoring stations must be deployed, and these will typically overlap in their area of coverage. The competing goals of effective wireless monitoring are to capture as many frames as possible, while minimizing the number of those frames that are captured redundantly by more than one monitoring station. Both goals may be addressed with a sampling strategy that directs neighboring monitoring stations to different channels during any period. To be effective, such a strategy requires timely access to the nature of all recent traffic.

We propose a coordinated sampling strategy that meets these goals. Our implemented solution involves a central controller considering traffic characteristics from many monitoring stations to periodically develop specific sampling policies for each station. We demonstrate the effectiveness of our coordinated sampling strategy by comparing it with existing independent strategies. Our coordinated strategy enabled more distinct frames to be captured, providing a solid foundation for focused sampling and intrusion detection.

I. Introduction

Wireless networks are increasingly being used by enterprises for mission-critical applications, including Voice over IP and backhaul infrastructure for sensor devices. Any interruption to service may cause loss of revenue or other bad consequences. It is, therefore, necessary to monitor wireless networks to detect anomalies or attacks in the network. The task of monitoring all channels everywhere in the coverage area is non-trivial because of the diversity of channels and the ubiquity of wireless access. Wireless sniffers, which we call Air Monitors (AMs), can be used to capture traffic on the wireless medium.

Each AM has the responsibility of capturing traffic in its own vicinity. There are several channels available in Wi-Fi networks, however, and many may be active simultaneously; it is challenging to simultaneously monitor multiple channels at the same location. One could attach multiple radios to one device so that each radio can monitor one channel, or place multiple single-radio devices at one location. Both methods are infeasible because the hardware required is bulky and prohibitively expensive. An example of a multi-radio device is the "porcupine."[1] A compromise solution is to monitor multiple channels using one radio, periodically changing the channel on which the radio is capturing traffic. Figure 1 shows how several channels at different locations can be monitored by using single radios at each location.

To monitor large areas, one must deploy several AMs. 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 shifts the radio sequentially through each channel in the wireless network, in a predetermined order, and spends equal amounts of time on each. If each radio is autonomous in its pattern of channel sampling, we call it *independent* sampling; if a single mechanism coordinates the sampling on multiple radios, we call it *coordinated* sampling.

In this paper, we describe the sampling problem in detail, the design of our sampling software in regard to its flexibility and simplicity, and our algorithms for coordinating multiple AMs and the metrics that drive them. We demonstrate by experiment that intelligent coordinated sampling is superior to independent sampling.

II. Related work

Few large-scale 802.11 measurement studies have attempted to capture wireless frames from the air; most prior work captures packet traces from the wired side of the access point [6], [9]. Although a few papers characterize traffic at meetings and conferences [8], [15], or describe tools for capturing and analyzing wireless frames [7], [10], [12], none of these papers consider channel sampling, merging, or security.

[1] http://www.porcupine.iu.edu/

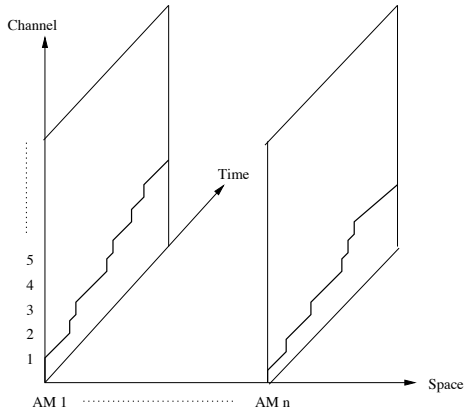

Fig. 1. Dynamic multichannel monitoring

A few recent papers describe offline tools to capture and merge wireless frames from AMs located around a building. Yeo et al. [16] use beacon frames that are heard at multiple sniffers to create a linear equation to synchronize sniffer clocks and subsequently remove duplicate frames. This system is only tested on offline traces and it is unclear whether it will work in an online measurement environment, which is necessary for quick detection of network problems. In addition, the sniffers in this system are all configured to capture packets on the same channel, which leads to a large percentage of frames being heard at multiple sniffers. The *Wit* [11] tool includes a similar merging mechanism to combine multiple traces and an inference engine to determine missing frames. The *Jigsaw* [4], [3] system uses multiple AMs to collect and merge traces for analysis of 802.11 networks. The focus of Jigsaw is to detect performance artifacts and network inefficiencies by reconstructing transport and link-layer conversations. The authors use common frames captured by different AMs for time synchronization. These methods do not sample the channels. The AMs are fixed on specific channels.

Our own earlier work [5] focused on the challenge of sampling traffic from many channels, and merging frames from many AMs. Our online merging software works with channel-hopping sniffers, which receive fewer frames in common. In that paper we reported more deeply on a large-scale experimental deployment, considered several independent sampling strategies, and considered the challenges of combining the streams of frames from different mergers.

A few publications examine captured frames in an attempt to detect attacks. For example, the DOMINO system [14] detects several MAC-layer cheating attacks, in which clients attempt to obtain greater wireless access by subverting the protocol's standard execution. Several commercial products [13] employ multiple sniffers although they do not provide the ability to merge streams from these sniffers. Typically the sniffers are used to send high-level summary statistics, rather than packet traces.

The DAIR system [1] uses USB NICs to turn desktops in an enterprise network into AMs, and could benefit from our techniques for traffic collection. In a follow-on publication [2] the authors use the DAIR system to localize clients to better understand network performance. DAIR assigns each sniffer to a specific channel whereas we support channel sampling to capture a broader range of traffic.

III. Sampling and efficient capture

The goal of a wireless monitoring system is to capture a representative subset of the traffic being transmitted through the network. Ideally, we wish to capture as much traffic as possible. In our attempt to achieve this goal, we try to maximize the total number of unique frames captured by maximizing the time spent by AMs 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*.

We consider a family of 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 this paper, we consider the *proportional* sampling strategy, which observes the frame rate on each channel, and uses the proportion of traffic on each channel to determine the proportion of the next scanning cycle to spend on each channel. Production infrastructure channels are typically the busiest, so this strategy partially achieves our goal of maximizing unique frame capture [5].

Consider a monitoring system with AMs deployed densely enough to be able to hear any client in the monitored area; there will necessarily be some areas covered by more than one AM. We say that two AMs are *neighbors* if they have recently captured a redundant frame. When employing proportional sampling, neighboring AMs will observe the same channel to be busy 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 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, as shown in Figures 2 and 3, such that the coverage includes minimal overlap, should result in even greater unique frame capture.

In this paper 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.
Our approach has three goals:

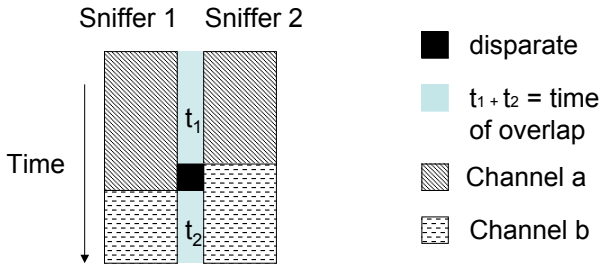

Fig. 2. Poor overlap between AMs.

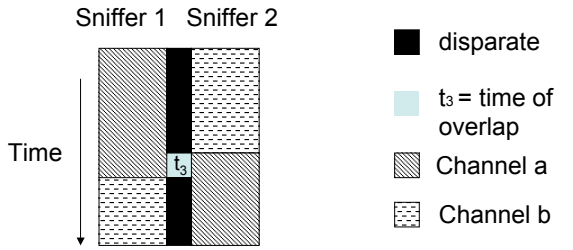

Fig. 3. Reduced overlap between AMs.

- 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 one time,
- deploying a sniffer costs money and space, hence limits deployment, and
- no frames are captured during channel changes, which take time.

IV. The coordination algorithm

We developed the following approach to reduce the amount of overlap among neighboring AMs. We assume that a central controller determines a sampling schedule for all 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 $a_1, a_2, ..., a_n$. Let these be vertices in a graph. Let there be an edge e_{ij} in the graph if the two AMs a_i and a_j are neighbors and are also on the same channel; as time progresses, edge e_{ij} comes and goes the graph as a_i and a_j follow their respective schedules. Let t_{ij} be the amount of time that the edge e_{ij} exists in the graph. We want to minimize the total *overlap* value $T = \sum t_{ij}$ over the entire schedule. Clearly a brute-force search of all possible schedules to find the minimum value for T 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 works as follows.

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 [5]. 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.

The above algorithm takes a greedy approach: if we reduce the overlap between two neighboring AMs, the overall overlap is likely to reduce as well. 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, we start off all the AMs using a schedule that spends equal time on every channel, and run the above algorithm every 20 seconds.

V. Dingo: a coordinated sniffer

We have developed a small collection of software components, named *dingo*, that collectively enable a variety of packet-sniffing strategies 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. Figure 4 shows the principal components of our software and the communication paths between them.

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 a variety of frame capture

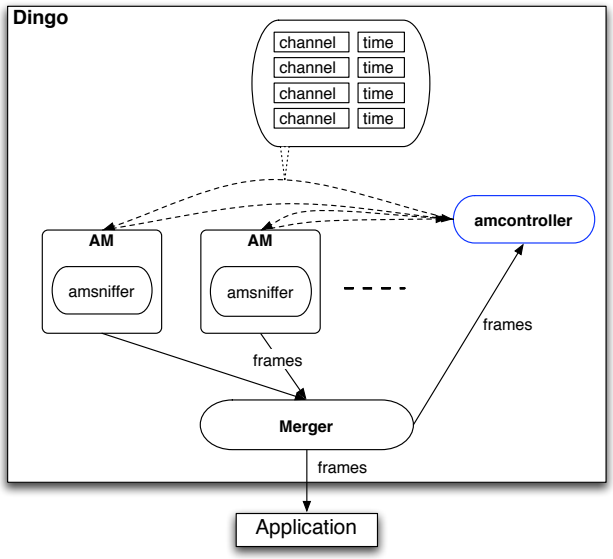

Fig. 4. Our sampling architecture

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.

The MAC-layer header of every frame captured and a summary of the wireless channel characteristics that existed during that capture are packaged into a single UDP/IP frame and transmitted, over a wired Ethernet infrastructure, to another important software component, the frame *merger* (described below). Of note, the payload of each captured frame is neither examined nor provided to any other software components in our work. Thus, our method works even on encrypted networks.

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.

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 [5] 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 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 and their frequencies are collated and forwarded as a UDP/IP frame stream to any number of subscribers, such as our amcontroller.

Periodically, every 20 seconds in our implementation, the amcontroller considers the current neighbor graph and 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.

VI. Sampling Experiments

We ran sampling experiments on a testbed in our department's three floor, 1,600 square meter building. The building has 19 Aruba AP52 access points that provide 802.11a/b/g service to over 80 resident faculty, students and staff members, so we deployed an additional 19 Aruba AP70s as AMs; Figure 5 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.

A. Experimental setup

We chose the Aruba AP70 as our sniffer boxes; each has a MIPS IDT32434 CPU running at 266MHz, 32MB DRAM, two Atheros AR5212 802.11a/b/g network interface controllers (NICs), 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.

We captured frames on both the 802.11 NICs 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 connected all the AMs to the merger through our wired building network, a switched 100Mbps Ethernet without any internal routers. Our merger and amcontroller ran as user-level processes on a Linux (Fedora core 4) server. This server has four 3.0GHz Intel Xeon CPUs, 4GB of RAM, and a 3.2TB RAID storage system.

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

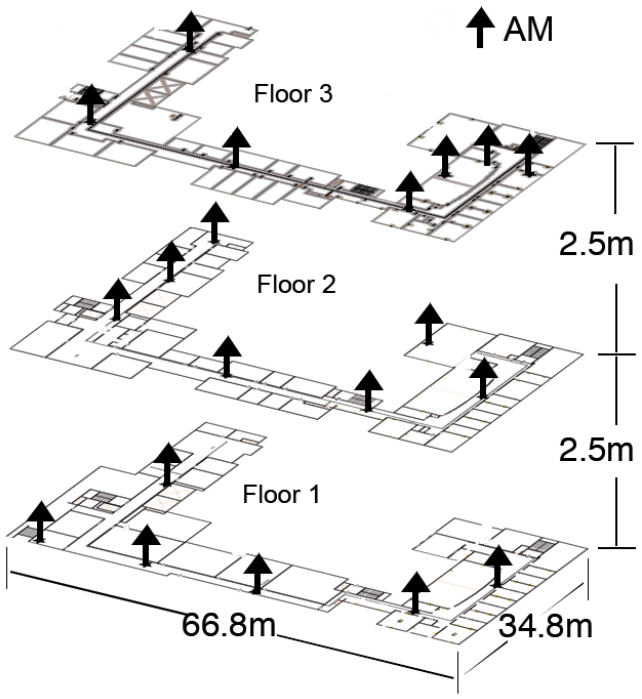

Fig. 5. AM Deployment

latter merger and used this information to create and update the neighbor graphs.

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

B. Results

Experiments were conducted over a period of one hour, and the number of unique frames captured by both the proportional and coordinated approaches collected and compared in 20 second intervals. The coordinated scheduling algorithm 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 milliseonds, meaning that neighboring AMs were successfully scheduled for different channels most of the time.

The data collected shows that coordinated sampling was successful for increasing unique frame capture. Figure 6 shows that, over a one hour period, coordinated sampling captured over 10% more unique frames in nearly all 20 second intervals than did 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.

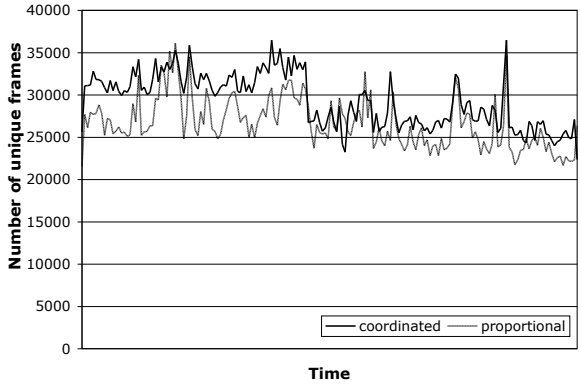

Fig. 6. Unique frame capture over one hour.

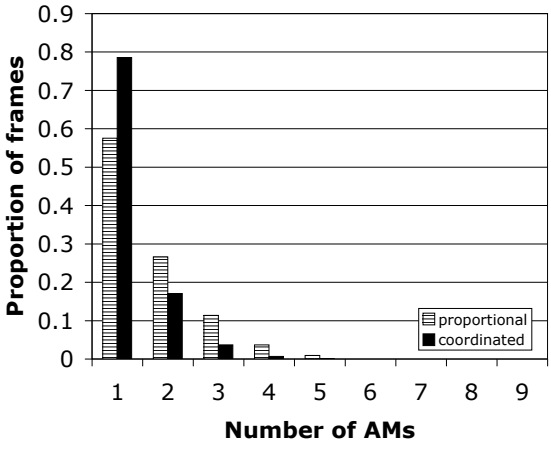

Fig. 7. Number of AMs capturing each frame over one hour.

Figure 7 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 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.

VII. Conclusion and Future Work

We found that using global information to schedule our monitoring system to 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.

Some frames may be more *relevant* to an application than others. For example, a security application may need frames from a particular client that acts suspiciously. Another application may require frames of a particular type. Monitoring systems need to be cognizant of the needs of applications. Our future work focuses on providing flexibility to applications so that the most relevant data can be made available by tuning the monitoring system to better meet the needs of the applications.

REFERENCES

[1] Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill. Enhancing the security of corporate Wi-Fi networks using DAIR. In *Proceedings of MobiSys 2006*, pages 1–14, Uppsala, Sweden, June 2006.

[2] Ranveer Chandra, Jitendra Padhye, Alec Wolman, and Brian Zill. A location-based management system for enterprise wireless LANs. In *Proceedings of USENIX NSDI*, April 2007.

[3] Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Peter Benko, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *Proceedings of SIGCOMM 2007*, Kyoto, Japan, August 2007.

[4] 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.

[5] Udayan Deshpande, Tristan Henderson, and David Kotz. Channel sampling strategies for monitoring wireless networks. In *Proceedings of the Second Workshop on Wireless Network Measurements*, Boston, MA, USA, April 2006. IEEE Computer Society Press.

[6] 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.

[7] Kyle Jamieson, Bret Hull, Allen Miu, and Hari Balakrishnan. Understanding the Real-World performance of carrier sense. 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.

[8] 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.

[9] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11:115–133, 2005. An earlier version appeared in ACM MobiCom 2002, and as Dartmouth College Technical Report TR2002-432.

[10] Feng Li, Mingzhe Li, Rui Lu, Huahui Wu, Mark Claypool, and Robert Kinicki. Tools and techniques for measurement of IEEE 802.11 wireless networks. In *Proceedings of the Second Workshop on Wireless Network Measurements*, Boston, MA, USA, April 2006.

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

[12] Arunesh Mishra, Eric Rozner, Suman Banerjee, and William Arbaugh. Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In *Proceedings of the 2005 Internet Measurement Conference*, pages 311–316, Berkeley, CA, USA, October 2005.

[13] Wireless IDS/IPS systems, Network World, June 2006. http://www.networkcomputing.com/showArticle.jhtml?articleID=189500017.

[14] Maxim Raya, Jean-Pierre Hubaux, and Imad Aad. DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots. *IEEE Transactions on Mobile Computing*, 5(12):1691–1705, 2006.

[15] Maya Rodrig, Charles Reis, Ratul Mahajan, David Wetherall, and John Zahorjan. 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-05)*, Philadelphia, PA, USA, August 2005.

[16] 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.