AirLab: Consistency, Fidelity and Privacy in Wireless Measurements

Vinod Kone, Mariya Zheleva, Mike Wittie,
Ben Y. Zhao, Elizabeth M. Belding, Haitao Zheng, Kevin C. Almeroth
Department of Computer Science
University of California, Santa Barbara
{vinod, mariya, mwittie, ravenben, ebelding, htzheng, almeroth}@cs.ucsb.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed. The author takes full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

Accurate measurements of deployed wireless networks are vital for researchers to perform realistic evaluation of proposed systems. Unfortunately, the difficulty of performing detailed measurements limits the consistency in parameters and methodology of current datasets. Using different datasets, multiple research studies can arrive at conflicting conclusions about the performance of wireless systems. Correcting this situation requires consistent and comparable wireless traces collected from a variety of deployment environments. In this paper, we describe AirLab, a distributed wireless data collection infrastructure that uses uniformly instrumented measurement nodes at heterogeneous locations to collect consistent traces of both standardized and user-defined experiments. We identify four challenges in the AirLab platform, *consistency, fidelity, privacy, security*, and describe our approaches to address them.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Monitoring*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Measurement Techniques*

General Terms

Measurement, Performance

Keywords

Wireless measurements, Airlab

1. INTRODUCTION

Accurate measurements of existing wireless networks are vital for researchers to perform realistic evaluation of proposed systems, and to identify and address their limitations. Given the expense and effort required to develop a data collection platform, deploy it in a network, and collect wireless measurements, most researchers form conclusions by relying on observations made from only a small number of (often only one) measurement traces. As a result, observations are often inconsistent across different networks, leading different researchers to draw potentially conflicting conclusions across their studies.

Significant progress has been made in improving the availability of wireless traces. The Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD) project[1] has made notable headway in the gathering of data sets collected from functional wireless deployments. Currently, the archive contains over 60 data sets from a wide variety of networks, including university campuses, conferences, academic departments, buses, cars, hotspots and testbeds. The data contains measurements of 802.11 networks, as well as Bluetooth, 802.15.4, and CDMA 1x EVDO.

While this wealth of information has become an invaluable resource to the wireless networking community, the inherent usability of CRAWDAD data is fundamentally limited by the lack of consistency in measurement parameters and measurement methodology across traces. Should a researcher want to confirm the presence of an observation in her measurement trace, it is extremely difficult, if not impossible, to find one (or more) similar traces gathered on a comparable type of network using a desired set of parameters. Even if the researcher manages to locate such a trace, it is likely that the desired metric of interest is missing from the second trace, thereby preventing any meaningful cross-comparison of network deployments. Hence, while each of these traces is useful in their own right, it is difficult, if not impossible, to draw any conclusions about the comparative operation of the networks. For example, of the 60 CRAWDAD datasets we examined in October 2009, 10 included data on indoor 802.11x networks. Of these 10 datasets, only 2 (NIIT/biterrors and Rutgers/capture) provide data on packet loss rates. But even these two are not comparable, given significant differences in drivers used (Prism II vs. MadWifi) and measurement methodology (driver modification vs. external sniffer).

To facilitate meaningful analysis of wireless networks, we need a way to collect measurement traces across a wide variety of network deployments, all using a consistent set of measurement metrics. The metrics should encompass a broad range of measurement types to maximize their utility to wireless researchers in different subfields. It is only through collection of a broad set of metrics from a variety of network deployments that we, as researchers, can deepen our knowledge of the operation of these networks.

A widespread multi-faceted data collection will provide multiple viewpoints of similar networks, enabling deeper understanding of network characteristics such as self- and exterior- interference properties, spectrum usage, and network traffic loads. Further, data collected across a variety of heterogeneous network types and traffic loads, such as university, corporate, and home environments, will facilitate cross-comparison of observed network phenomena within each of these settings.

To address the critical need for comparable, consistent wireless measurement traces, we propose AirLab, a distributed infras-

[1] http://crawdad.cs.dartmouth.edu/

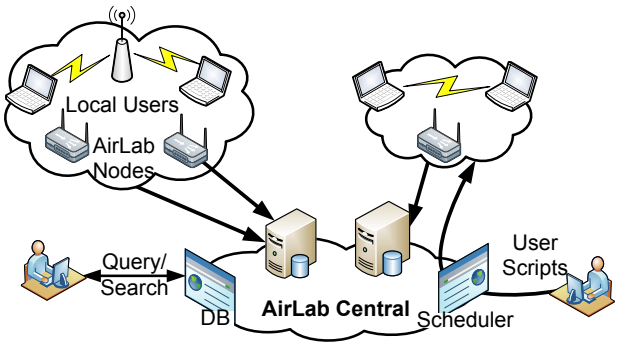

Figure 1: AirLab Architecture. AirLab includes measurement nodes located at member sites that perform local measurements and send traces back to AirLab Central. Users can query the collected data, as well as schedule customized measurement tasks written as scripts calling AirLab's measurement API.

tructure for wireless measurements. AirLab is an effort to deploy uniformly instrumented wireless measurement nodes spanning the globe at multiple educational and research institutions, as well as private homes. Each AirLab site runs one or more AirLab nodes, each a standalone wireless-enabled mini PC running Linux and a customized wireless measurement API stack. All nodes perform a core set of ongoing periodic passive measurements to study long-term wireless properties. Collected data is streamed to AirLab Central, a central data repository and management facility. In addition to standardized ongoing measurements, AirLab Central allows partners to implement, schedule, and perform customized measurement experiments across multiple sites using high level measurement scripts.

In designing the AirLab architecture, we focus on achieving four critical goals:

1. *Consistency.* To ensure comparability of data traces collected from multiple locations, all AirLab nodes run uniform software stacks from the OS to the measurement API, and use identical wireless cards.

2. *Fidelity.* We want to maximize the number of simultaneous user-scheduled experiments performed by AirLab without sacrificing measurement fidelity.

3. *Privacy.* AirLab partner sites ensure that measurement data sent to AirLab Central does not compromise the privacy of local wireless users or their data transmissions.

4. *Security.* Since AirLab supports user-written measurement scripts, it protects against accidental or intentional misuse of the platform.

We have completed the high level design of the AirLab architecture, and are currently developing and testing the software measurement stack for AirLab measurement nodes. We began local site deployment and measurement tests at UCSB in January 2010, with planned deployment to external partner sites in US and UK in fall 2010.

In the remainder of this paper, we describe the AirLab architecture, challenges, and mechanisms.

2. AIRLAB ARCHITECTURE

AirLab is designed as a distributed measurement platform with wireless measurement nodes deployed at participating institutions distributed around the globe. It aims to produce a reliable, distributed platform for gathering consistent wireless measurements from heterogeneous deployment settings. The AirLab infrastructure is organized as a loose confederation of remote measurement sites, all controlled and managed through a central site (AirLab Central) located at UC Santa Barbara. Like the PlanetLab Internet testbed [7], each participating site hosts at least one AirLab measurement node. Unlike PlanetLab which allows users to execute arbitrary programs, AirLab restricts experiments to passive and active measurements of the local wireless environment through controlled interpretation of user scripts. A high level picture of the AirLab architecture is shown in Figure 1.

Deployment. Uniformity and consistency across measurement results are ensured because all AirLab measurement nodes in our initial deployment share the same hardware and software configurations. Each node is a mini PC with three 802.11 radios and a customized software stack (Section 4). Nodes use each wireless interface to search for local 802.11 networks, and monitor different channels within each network. We initially focus on 802.11 because of its ubiquity. Once the platform has matured, however, we plan to explore integration of more advanced hardware platforms such as USRP GNU Radios. We expect to deploy AirLab nodes in a variety of wireless settings, including university departments, research labs, and private homes. Most sites will deploy nodes within buildings, while the rest might deploy nodes in open air environments (*e.g.* covered parking garages). Sites will vary significantly in AP density, interference levels, and traffic volume.

Access Control and User Management. Like PlanetLab, membership is managed on a per-site basis. Local PIs at each deployed site have the ability to create and manage a fixed number of user accounts. Unlike PlanetLab, however, AirLab is not a full experimental testbed. Users do not have remote login access to AirLab nodes. This restriction allows administrators to maintain greater control over scheduling and admission control, leading to improved stability and resource utilization.

Functionality. AirLab nodes support both continuous passive measurements of a set of baseline metrics, as well as user-defined measurement metrics. We define a set of measurements that collect baseline traces of interesting wireless channels in local environments. User-defined measurement scripts are uploaded from AirLab Central to remote measurement nodes and executed locally. Resulting data traces are uploaded to AirLab Central, imported into a central database, and processed to generate user-defined statistics. Users can perform queries on anonymized traces and statistics.

3. MEASUREMENT FRAMEWORK

AirLab provides a distributed infrastructure for consistent and comparable wireless measurement experiments. At a high level, AirLab supports both *baseline* and *user-defined* measurements.

Baseline Measurements. Baseline measurements at AirLab nodes are a key contribution to the AirLab framework. The goal of baseline measurements is to continuously and passively monitor the local wireless network for extended periods of time. Such monitoring will help researchers to analyze and understand the short-term and long-term network dynamics of a wide-variety of wireless deploymnets. Large-scale analysis of different networks and their evolution is not possible without uniformly collected traces. In addition to the collection of raw traces, we also compute and store a predefined set of baseline metrics useful to researchers. Our initial set of metrics include channel load, number of active access points and users, packet loss, control traffic overhead and average RSSI

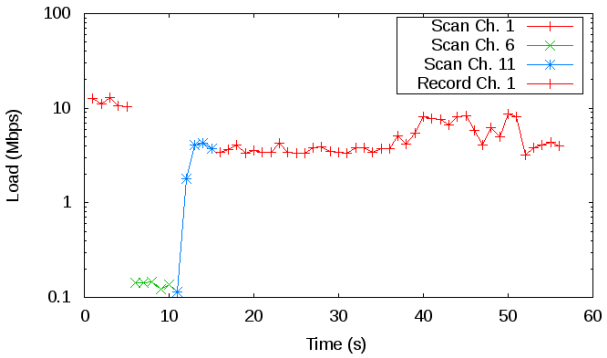

Figure 2: Baseline experiment visualization. The node scans active channels 1, 6 and 11 and picks the highest utilized channel 11 for recording baseline data.

values. These baseline metrics will continue to evolve throughout the course of the project based on the needs of our participating partners and users.

One of the challenges in our baseline measurement collection will be ensuring collection of "interesting" data sets. Configured naïvely, our AirLab measurement nodes could initialize to a default channel, and proceed to collect data on that channel regardless of its utilization. Instead, we are exploring a simple alternative where our measurement nodes scan all available channels, and collect data on the channel with the highest utilization. Figure 2 illustrates our baseline measurement process at a high level. The x-axis shows time and the y-axis shows a baseline metric, here channel load, averaged over one second buckets. The distinct lines show the radio moving across multiple channels, scanning each for several seconds to detect short term utilization. In this case, channel 11 shows the highest load, and is selected for longer observation. Thus all baseline metrics will be collected on channel 11 for the next measurement period, until the next periodic scan. To adapt to the network dynamics, AirLab nodes periodically scan the network every hour and monitor the highest loaded channel. To avoid being interrupted by user-defined measurements, one of the three radio interfaces on each AirLab measurement node is always dedicated for baseline measurements.

User-defined Measurements. In addition to the baseline passive measurements, AirLab provides users with the ability to design and execute their own passive or active measurements, which can be scheduled either for a specific time or to be triggered by specific events or wireless conditions. Instead of users logging in locally and loading their own code on each measurement node, AirLab provides a Core Measurement API (Section 4) that users can utilize to compose their measurements. We provide users the flexibility to write their measurement experiments in any scripting language[2], submit it to AirLab Central's scheduler and receive results asynchronously when they complete. Since one interface is dedicated to baseline measurements, user-defined measurements can only access two of the three interfaces.

In return for constraining user measurement tasks, AirLab gains several key advantages. First, because measurement tasks are scheduled, AirLab can use admission control to ensure that each measurement task is allocated its minimum required time slice, and to eliminate resource conflicts between multiple tasks. Second,

[2]Our current codebase supports Python, with support for other languages en route.

Figure 3: AirLab Measurement Node.

we believe most user-defined tasks only need to perform measurements on a periodic or as-needed basis. Scheduling allows AirLab to co-schedule compatible measurement tasks together on the same nodes, thus maximizing node utilization through time-based multiplexing. For example, one task performing channel sensing and another task measuring link loss on the same channel can interleave their measurements without causing conflicts. Finally, since users cannot log in or modify software, this greatly simplifies the task of securing AirLab nodes and helps to improve their stability and availability. We discuss the security and privacy issues in more detail in Section 5.

User Interface. Users interact with AirLab via a Web interface to manage experiments and the data they collect. To create an experiment, a user selects a measurement script, configuration options, and metrics. Configuration options declare whether the experiment is active or passive, specify radio interface and channel, as well as experiment start time, or trigger and experiment duration. Finally, the user selects measurement nodes, on which the experiment is to be scheduled.

Each experiment is associated with one or more metrics. AirLab offers several common metrics of interest in baseline experiments. Users can define their own metrics using either `tshark` statistics via the `-z` option, or use custom scripts that compute metrics from a pcap file.

4. IMPLEMENTATION

4.1 AirLab Central

AirLab Central is served by a DELL PowerEdge R510 with 12GB of memory running CentOS, initially configured with 12TBs of RAID 1 storage for collected traffic traces and computed metrics. Initial estimates suggest this amount of storage can support baseline experiments in the initial deployment of 20 nodes for at least 12 months. Beyond this, we have plans to expand storage capacity and to assure data persistence via off-site backups.

Collected data, calculated metrics, and all user and experiment settings are stored in a MySQL database. A Python-based Django Web application server handles user requests from the Web interface as well as remote interactions from AirLab measurement nodes. To assure security of raw packet captures, trace files are send from measurement nodes to AirLab Central over encrypted connections and are anonymized immediately upon arrival using `PktAnon` [4]. Tshark, or user-defined scripts are then used to calculate metrics from the anonymized traffic. These metrics are stored in the database then displayed on AirLab Central Web interface via Multi-Router Traffic Grapher (MRTG) plots.

4.2 Measurement Nodes

The core of an Airlab measurement node is Twitter-31270A, a custom embedded network platform from Hacom [1], with 1.6 GHz Intel Atom CPU, 1 GB RAM, and a 32 GB Compact Flash and 160

GB Hard Drive for storage. The platform also includes two Mini-PCI Express slots where we installed two 802.11 a/b/g/n cards, and one Mini-PCI with a 802.11 a/b/g card installed. All NICs are connected to dual-band rubber duck external antennae for improved communication (See Figure 3). We chose this specific platform for its small-foot print, manageability and support for multiple wireless NICs.

Hardware Limitations. We conducted multiple in-lab experiments to test the capability of our multi-radio platform. Our experiments were designed to answer four questions:

- Can the wireless NICs capture all the traffic when the wireless channel is saturated with traffic at 54 Mbps?
- Are the signal strength measurements reported by the cards stable and consistent?
- What is the impact of crosstalk on the closely spaced wireless NICs?
- What is the impact of heterogenous wireless interfaces, PCI vs PCI-Express?

In our experiments, we configured four laptops as two pairs of adhoc links communicating on orthogonal and isolated 802.11a channels (Channels 48 and 64). The two transmitters used iperf to continously send UDP traffic at a constant rate to their respective receivers. The AirLab node was placed nearby the two transmitters and had line of sight with all 4 laptops. Each of the 3 wireless NICs of the AirLab node was either monitoring channel 48 or monitoring channel 64 or powered down. We tested with several combinations of interfaces and monitoring states and analyzed the traces. The take-away results from our analysis are as follows:

First, all three interfaces can simultaneously capture wireless traffic even when their respective channels are saturated. With 54 Mbps traffic, PCI-Express cards captured 99% and PCI card captured 97% of all the traffic. We believe the slightly worse performance of PCI is due its slower bus speed compared to PCI express.

Second, RSSI values reported by the cards are not impacted when the cards are monitoring different channels. But, when two or more cards are operating on the same channel, we observed fluctuations in the RSSI, in some of our experiments. This implies that crosstalk impacts the NICs when operating on the same channel.

Third, all wireless interfaces reported stable RSSIs, though the absolute values differed from NIC to NIC. We found that this was due to the sensitivity of hardware to slight orientations of antennae and connector cables between the external antennae and the NICs.

Consequently, we made the following design decisions

- The PCI interface is dedicated for baseline measurements, and the PCI-Express interfaces are dedicated for user-defined measurements.
- No two interfaces are allowed to operate on the same channel at the same time.
- If an active measurement is running on an interface, the other two interfaces should not run any measurements for the duration of the active experiment.
- RSSI measurements need to be calibrated once the nodes are deployed at remote sites to account for hardware sensitivity.

Software Stack. All AirLab nodes run the desktop edition of Ubuntu 10.04 Linux with a custom software stack shown in Figure 4. The AirLab measurement library makes use of multiple

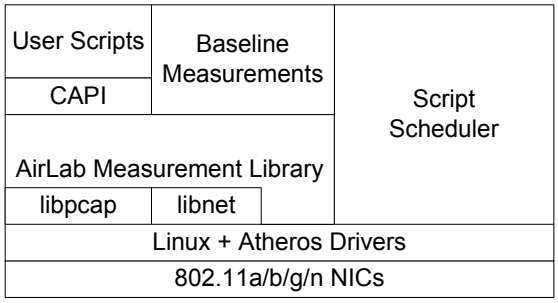

Figure 4: AirLab Software Stack. Each node has a measurement library on top of MadWifi Atheros drivers. A scheduler receives user scripts from AirLab Central, schedules measurements and uploads traces. User scripts are executed alongside AirLab's basesline measurements.

libraries such as `libpcap`, `libnet`, and the MadWifi drivers (`ath5k` and `ath9k`) for interacting with wireless devices and communicating with the local WiFi network. The Script scheduler is a core component of the AirLab node, which is responsible for scheduling measurements and communicating with AirLab Central to upload traces and receive new measurements.

Core Measurement API. AirLab includes a Core measurement API (CAPI), exposed by the measurement library, to support passive or active user-generated measurement experiments. CAPI allows users to configure and perform customized passive and active measurement tasks, using a standard API across all AirLab nodes. CAPI provides AirLab users with a high-level and easy-to-use interface, shielding them from the complexity of the underlying radio hardware. Users can design measurements on specific channels, interfaces and time periods, and use timers or conditional events to trigger measurements.

For passive measurements, AirLab nodes behave like sniffers that monitor and collect network information. CAPI makes AirLab nodes significantly different from conventional sniffers by allowing AirLab users to configure customized measurement tasks, and by allowing multiple measurement tasks to execute concurrently. For passive transmissions, AirLab produces the measurement dataset by using `libpcap` to capture packet headers from each ongoing measurement.

In addition to being able to passively monitor a network, AirLab nodes can inject test (or application) packets into a network to measure the service obtained from the network. For these active measurements, CAPI uses `libnet` to generate and send custom traffic generated by the user. Currently, AirLab supports sending TCP, UDP and Ping packets. Additionally, CAPI interfaces with popular traffic generator tools like iperf, netperf and httperf to give users the flexibility to generate bulk traffic. For active measurements, CAPI sets the corresponding wireless interface into `Station` mode and connects to one of the APs in the neighborhood. A custom network manager scans all the nearby APs, and connects to an AP for which it has access credentials.

4.3 Scheduling

With both passive and active measurements running on AirLab, we must carefully schedule their execution to a) maximize the number of experiments supported by a fixed AirLab infrastructure, b) minimize overhead on local networks imposed by measurements, and c) provide flexibility for users to specify their experiment conditions.

As noted earlier, users can schedule data collection experiments as either time-based or event-triggered. Time-based scheduling can provide information relevant to daily usage patterns. To schedule a time-based measurement task, a user specifies the start time period T and duration D for which the experiment should run. On the other hand, event-triggered collection occurs when user-specified network conditions are met. One example of such an event is: "when the channel load of the network is at least 2 Mbps, collect traces for time period D."

Scheduling in AirLab is divided between AirLab Central (Central Scheduler) and measurement nodes (Script Scheduler). The central scheduler is responsible for selecting and coordinating measurements across different measurement nodes. Once the required nodes are identified, the central scheduler pushes the measurement scripts to those nodes. Only a selected subset of nodes to which the measurement scripts are pushed might be informed to actually run the experiments. This is useful for event-triggered measurement tasks for which the feasibility of schedulability cannot be determined beforehand. For time-based experiments, the central scheduler can deny a new user-defined measurement if it cannot find a place for the experiment in its schedule. The Central Scheduler relies on proactive updates from measurement nodes to maintain a consistent schedule of all measurement nodes in its database.

The responsibility of a local script scheduler is to schedule measurements pushed to its node, execute them and upload the resulting traces back to AirLab Central. It creates and maintains a local schedule for each local wireless interface. When the scheduler receives a new time-based experiment, it schedules the experiment if doing so does not introduce conflicts with previously scheduled experiments. If conflicts arise, the measurement is rejected and a report sent back to AirLab Central. In contrast, an event-triggered experiment is put on a wait list, and periodically checked to see if its trigger conditions are satisfied. The scheduler maintains a table of statistics for all the 32 channels (11 for 2.4GHz and 21 for 5GHz bands) in the network. These statistics are updated periodically from traces of user-defined and baseline measurements. This table is checked periodically and event-triggered tasks whose prerequisites are satisfied are moved into the schedule. Any update to the local schedule triggers a synchronization exchange to update the schedule at AirLab central.

The script scheduler is also responsible for periodically (every 30 min) uploading the collected traces to AirLab Central through an encrypted SSH tunnel. Finally, since the measurement nodes are likely behind NATs, the scheduler performs NAT hole-punching and makes AirLab Central aware of its publicly routable IP address.

5. SECURITY AND PRIVACY IN AIRLAB

Providing security and data privacy on the AirLab platform is the most difficult challenge we face. In particular, we must design policies and mechanisms that do their best to deliver data requested by AirLab users, while protecting the privacy of users near AirLab deployments. This is a challenging and open research problem which we will tackle while leveraging the expertise and active research of collaborators in the community.

Security of AirLab nodes. Even with our restrictive user access model for AirLab nodes, it is still possible for active user measurements to burden or overwhelm local networks with extra traffic. Worse yet, compromised user accounts can be used to perform Denial of Service (DoS) attacks on local networks near AirLab nodes.

To prevent against accidental and intentional misuse of AirLab nodes, we use a two-pronged approach. First, we set an upper bound on the amount of outgoing traffic per time that can be sent by each AirLab node. We employ a stringent static analysis tool at AirLab central that parses user scripts at submission time, and determines whether it is possible for a script to exceed our transmit rate limits. Scripts that can potentially exceed our rate limits are returned to the submitting user for review and editing. Second, because static analysis can under-report the traffic generated by an experiment, we deploy a run-time traffic monitoring mechanism on each AirLab node that monitors outgoing traffic volume. In addition to enforcing our own rate limits, the run-time monitor also observes existing traffic volume on the same channel, and delays or terminates any experiment that imposes significant congestion or delay on the local network.

Finally, the measurement library is compiled as an executable, and language specific wrappers are used to interface with user scripts. The rationale for building an executable is that the library can be exclusively given root privileges to interact with device drivers, whereas the user script is still run under normal user privileges. This makes it hard for malicious user scripts to compromise the system integrity of AirLab nodes.

Data Collection and Privacy. User privacy of user communications is a critical component necessary for the successful adoption of AirLab. During passive measurements, an AirLab node acts as a sniffer that monitors the wireless network without interfering with local transmissions. In addition to data frames, nodes can capture management frames such as beacons, authentication and RTS/CTS/ACK frames in 802.11.

We recognize that data privacy is an extremely difficult challenge, and that perfect data anonymization may not be possible. As a starting point, we utilize the `PktAnon` package from the University of Karlsruhe, which uses flexible anonymization profiles to customize anonymization for a wide range of network protocols at varying levels of security, using techniques ranging from prefix anonymization to byte-level hashing and bit-level noise infusion. We will also consider and analyze the security properties of available anonymization techniques, including `tcpmkpub` [22], `TCPdpriv`[3], `PktAnon` [4], and the system proposed by [14]. As we begin to deploy AirLab nodes, we are beginning collaboration and discussions with researchers at the NetSANI project[4].

While anonymization can thwart many types of attacks, existing work has also shown that significant data can be inferred even from anonymized traces [10, 14, 22]. Beyond anonymization, we design the data communication and storage mechanisms in AirLab to protect against accidental information leakage. Since AirLab nodes have limited processing power, collected data is offloaded at short intervals (30 min) over a SSH tunnel to AirLab Central, where it is anonymized immediately upon arrival using a hash key unique to each experiment. Finally, anonymized data at AirLab Central is available only to the user that created the corresponding experiment and to collaborators, who are granted access to specific experiments. In time, experiment data can be made public to all AirLab users.

6. RELATED WORK

Related work on wireless measurements can be classified as follows: (i) measurement studies of indoor production 802.11 networks, which have covered a variety of environments, including university departments [12, 13, 19], corporate enterprises [21], neighborhood networks [15], and conference and professional meetings [17,

[3]TCPdpriv is written by Greg Minshall and available at `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`.
[4]NetSANI: `http://www.ists.dartmouth.edu/projects/NetSANI.html`.

20]; (ii) measurement studies of outdoor mesh and WiFi networks [2, 3, 11]; (iii) measurement studies of vehicular and mobile networks [6, 30]; and (iv) measurement studies of mesh network testbeds [18, 25]. Many of these measurement traces are available from CRAWDAD, a central wireless measurement depository.

AirLab is similar to Harvard's MoteLab project [27]. But where MoteLab is a centralized testbed with full user access to motes, AirLab focuses on distributing measurements across heterogeneous locations with stronger restrictions on user access.

The most relevant work in wired network measurements is NETI [26], a distributed approach to collecting end-to-end network performance measurements among Internet users. This work uses passive measurements and only collects measurements in a single protocol layer (i.e. UDP/TCP). Similarly, DOME [28] is another distributed approach for passive network measurement. This work assumes predetermined measurement tasks, and only supports active measurement. Other wired network testbeds include VINI [8] and PlanetLab [7]. Wireless measurements differ significantly from wired data because the wireless medium is physically dispersed, and the usage patterns are much more diverse due to mobility and the proliferation of new devices. Hence while we can learn from the methodologies employed in wired networks, new solutions are required to understand the properties of the wireless physical medium.

The analysis of collected data sets has resulted in a wide variety of network characterization studies by both the data collectors themselves as well as the general research community when the traces have been made public. A small sampling includes characterization and analysis of wireless networks and networking protocols [5, 11, 15, 18], traffic and usage patterns [16, 17, 19], detection of performance anomalies [13, 12, 24], network troubleshooting [23], analysis of the performance of deployed services [2], and design of network protocols [9, 29]. While each of these studies has contributed to the understanding of the operation and usage of wireless networks, each is an isolated study of a specific deployment. The lack of a data set collected using a homogeneous methodology from multiple network deployments has prevented the cross-correlation of observed phenomena between networks. Hence, it is difficult to discern whether observed behaviors are specific to a particular network deployment or are intrinsic to the protocols in use or the wireless medium itself.

7. CONCLUSION

The effective design of next generation wireless protocols and systems requires deep understanding of wireless properties across a wide variety of operating environments. This in turn depends on the availability of consistent and comparable wireless measurements. To address this need, we propose AirLab, a wide-area wireless measurement infrastructure that supports both a core set of ongoing passive measurements as well as user-scheduled, user-generated wireless experiments implemented using a restricted measurement programming interface.

We are currently developing and testing AirLab software through local experiments at UCSB, and will begin deployment to a small group of partner sites in the coming year. As AirLab grows and matures, we hope to expand AirLab sites to numerous locations around the globe, thereby increasing capacity for user-driven measurements while making an increasing volume of consistent and comparable datasets available to the research community at large. We welcome participation from the wireless research community, and institutions interested in joining AirLab can contact the authors for more information.

8. REFERENCES

[1] http://www.hacom.net.

[2] AFANASYEV, M., CHEN, T., VOELKER, G. M., AND SNOEREN, A. C. Analysis of a mixed-use urban wifi network: When metropolitan becomes neapolitan. In *Proc. of IMC* (October 2008).

[3] AGUAYO, D., ET AL. Link-level measurements from an 802.11b mesh network. In *Proc. of ACM SIGCOMM* (September 2004).

[4] AGUILAR, J. Packet trace anonymization with pktanon. Hack a day, July 2008. `http://hackaday.com/2008/07/11/packet-trace-anonymization-with-pktanon`.

[5] AKELLA, A., JUDD, G., SESHAN, S., AND STEENKISTE, P. Self-management in chaotic wireless deployments. In *Proc. of MobiCom* (2005).

[6] BALASUBRAMANIAN, A., ET AL. Interactive WiFi Connectivity for Moving Vehicles. In *Proc. SIGCOMM* (Aug. 2008).

[7] BAVIER, A., ET AL. Operating system support for planetary-scale network services. In *Proc. of NSDI* (March 2004).

[8] BAVIER, A., ET AL. In vini veritas: realistic and controlled network experimentation. *SIGCOMM CCR 36*, 4 (2006).

[9] BROUSTIS, I., ET AL. Mdg: Measurement-driven guidelines for 802.11 wlan design. In *Proc. of Mobicom* (Sept. 2007).

[10] BURKHART, M., ET AL. The role of network trace anonymization under attack. *SIGCOMM CCR 40*, 1 (January 2010), 5–11.

[11] CAMP, J., ET AL. A measurement study of multiplicative overhead effects in wireless networks. In *Proc. of INFOCOM* (April 2008).

[12] CHENG, Y.-C., ET AL. Jigsaw: solving the puzzle of enterprise 802.11 analysis. *SIGCOMM CCR 36*, 4 (2006).

[13] CHENG, Y.-C., ET AL. Automating cross-layer diagnosis of enterprise wireless networks. *SIGCOMM CCR 37*, 4 (2007).

[14] FAN, J., XU, J., AMMAR, M. H., AND MOON, S. B. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks 46*, 2 (October 2004), 253–272.

[15] HAN, D., ET AL. Mark-and-sweep: Getting the "inside" scoop on neighborhood networks. In *Proc. of IMC* (Oct. 2008).

[16] HENDERSON, T., KOTZ, D., AND ABYZOV, I. The changing usage of a mature campus-wide wireless network. In *ACM MobiCom* (Sept. 2004).

[17] JARDOSH, A. P., ET AL. Understanding congestion in ieee 802.11b wireless networks. In *Proc. of IMC* (Oct. 2005).

[18] JUDD, G., AND STEENKISTE, P. Using emulation to understand and improve wireless networks and applications. In *Proc. of NSDI* (2005).

[19] KOTZ, D., AND ESSIEN, K. Analysis of a campus-wide wireless network. In *ACM MobiCom* (September 2002).

[20] MAHAJAN, R., ET AL. Analyzing the mac-level behavior of wireless networks in the wild. *SIGCOMM CCR 36*, 4 (2006).

[21] MURTY, R., ET AL. Designing high performance enterprise wi-fi networks. In *Proc. of NSDI* (2008).

[22] PANG, R., ALLMAN, M., PAXSON, V., AND LEE, J. The devil and packet trace anonymization. *SIGCOMM CCR 36*, 1 (2006), 29–38.

[23] QIU, L., BAHL, P., RAO, A., AND ZHOU, L. Troubleshooting multihop wireless networks. In *Proc. of SIGMETRICS* (2005).

[24] RAGHAVENDRA, R., ET AL. Unwanted link layer traffic in large IEEE 802.11 wireless networks. In *Proc. of IMC* (Oct. 2007).

[25] RAYCHAUDHURI, D., ET AL. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *Proc. of WCNC* (2005).

[26] SIMPSON, JR., C. R., AND RILEY, G. F. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *Proc. of PAM* (April 2004).

[27] WERNER-ALLEN, G., SWIESKOWSKI, P., AND WELSH, M. Motelab: A wireless sensor network testbed. In *Proc. of IPSN* (April 2005).

[28] WOLF, T., ET AL. An architecture for distributed real-time passive network measurement. In *Proc. of MASCOTS* (2006).

[29] YANG, L., CAO, L., ZHENG, H., AND BELDING, E. Traffic-aware dynamic spectrum access. In *Proc. of WICON'08* (November 2008).

[30] ZHANG, X., ET AL. Study of a bus-based disruption tolerant network: Mobility modeling and impact on routing. In *Proc. of Mobicom* (Sept. 2007).