Tools and Techniques for Measurement of IEEE 802.11 Wireless Networks

Feng Li, Mingzhe Li, Rui Lu, Huahui Wu, Mark Claypool, Robert Kinicki
{lif,lmz,kkboy,flashine,claypool,rek}@cs.wpi.edu
CS Department at Worcester Polytechnic Institute
Worcester, MA, 01609, USA

Abstract—With the growing popularity of wireless local area networks (WLANs) has come an increased need for effective measurements of real-world WLANs and their applications. This paper presents tools and techniques for measuring IEEE 802.11 WLANs. The techniques include details on building a host access point and setting up a wireless sniffer while the tools include programs for measuring link, network and application layer traffic. The tools are all open-source software available for download and the techniques all use open-source software and off-the-shelf hardware components. Together, these tools and techniques facilitate WLAN performance analysis across network layers in a flexible, accurate and cost-effective manner. To illustrate the usefulness of these tools and techniques for gathering WLAN measurements, three case studies are presented: a streaming video session showing cross-layer performance; network characteristics of a wireless hand-held game; and measurements of access point queue size. Research employing these tools can yield more accurate WLAN models and more realistic evaluation of proposed WLAN changes in a network testbed.

I. INTRODUCTION

The combination of the decrease in price of wireless local area networks (WLANs) and the increase in wireless link capacities has significantly increased the number of WLANs in homes, corporate enterprise networks, academic campus networks, and entire cities. Initially, much of the WLAN research was conducted primarily through the use of analytic models in [2], [3] and simulation techniques [18], [6], [4]. Only recently have researchers tackled the task of measuring WLANs [12], [17], [20] to understand performance anomalies and the implications of installation choices.

However, accurate WLAN measurements have proven more elusive than measurements for wired LANs due to the unique characteristics of the wireless medium. For example, measurements over a single wireless hop, such as in an 802.11 infrastructure network, can vary depending upon the hop distance, cross and contending traffic, the building structure and even the human motion within a measurement testbed. Generally, capturing aspects of WLAN performance requires more than collecting measurement data at any one layer in the protocol stack. Instead, measurement tools and techniques that enable the researcher to observe the intertwined effects between network layers are important for a comprehensive assessment of WLAN performance [10].

Building network testbeds, a common approach used to run controlled experiments, provides special challenges within WLANs where accurate multi-layer measurements require custom hardware and software solutions and realistic measurements tend to come from commercial, often proprietary, black-box software and hardware. For example, wireless sniffers, while effective for trouble shooting and other diagnostic functions, are typically expensive and closed-source devices that offer less flexibility with respect to capturing specific performance metrics compared to employing open-source solutions. While inexpensive, today’s commercial wireless access points (APs) remain as black-box components in the WLAN in that their exact internal configurations and protocol implementations are not normally known.

This paper presents useful tools and techniques for measuring IEEE 802.11 WLANs in real-world environments. One technique includes the details on building a host AP using off-the-shelf hardware and open-source software. Building your own host AP means being able to gather measurements concurrently from several network layers and providing the ability to experiment with customized changes within the AP. Another technique provides details on building a wireless sniffer to capture 802.11 and higher network layers, permitting non-intrusive network measurements and evaluation of proprietary 802.11 devices. Tools at the end-host allow measurement of wireless characteristics obtained through the wireless interface card and round-trip time and loss rate at the IP layer. Application-layer tools, previously developed for wired-network testbeds, are mentioned to provide information for complete wireless network experimentation. All the tools and techniques discussed in this paper are open-source software available for download at http://perform.wpi.edu/tools/.

This paper is organized as follows: Section II presents tools and techniques for IEEE 802.11 network measurement; Section III illustrates the use of the tools and techniques with three performance evaluation examples; and Section IV provided conclusions and future work.

II. TECHNIQUES AND TOOLS

Figure II depicts a basic testbed deploying the tools and techniques presented in this paper, with the underlined links being specific contributions of our work.

A. Techniques

A.1 Build a Wireless Sniffer

While sniffers have been widely used to monitor network traffic at the data-link layer and above, most commercial wireless sniffers are costly and are not a flexible open source solution. However, passive sniffing does not interfere with the hosts under test and does not require access to the hosts themselves. Thus sniffers can be used to measure black-box devices such as handheld game consoles (see Section III-B).

Our first wireless measurement technique is turn a PC into an IEEE 802.11 wireless sniffer from open source software and
off-the-shelf wireless networking hardware.\(^1\)

1. Install the Linux operating system. The Linux distributions tested include SUSE (Novell) Linux releases 9.0, 9.1, 9.2 and 10.0 and Red Hat Fedora Core 3, with Linux kernel versions 2.4 and 2.6.
2. Integrate a prism GT-based wireless network interface card. The wireless network interface cards tested include Netgear WG 511 version 1 PCMCIA card and Allnet ALL0271 54 Mbit Wireless PCI adapter.
3. Update the driver (a prism54 kernel module) to the latest version.\(^2\)
4. Create an interface configuration file that can brings the wireless interface up in monitor (sniffing) mode.
5. Use network sniffing tools to capture frames. Popular network tools to capture and analyze capture data include tcpdump,\(^3\) Ethereal,\(^4\) or Kismet.\(^5\)

The wireless sniffer captures the IEEE 802.11 MAC layer header, which provides data-link frame information, such as the number of Medium Access Control (MAC) layer retries, power management functionality and WEP encryption information. In addition, the wireless sniffer provides extra physical and MAC layer information provide by the Absolute Value Systems (AVS) header,\(^6\) a “extra” header which is created by the wireless driver while working in monitor mode. With the AVS header, the wireless driver collects wireless layer metadata, including the received signal strength, channel number and capacity and more, and passes the metadata to the sniffer as an emulated frame header. In addition to performance monitoring, the sniffer can capture wireless management frames, such as RTS/CTS, (De)authentication, and (Dis)association, allowing it to be used as a wireless network diagnostic tool.

A.2 Build a Host Access Point

Results from prior WLAN measurement studies [1], [10], [14], [18] indicate that the internal allocation of buffers within a wireless AP can significantly impacts the performance of WLAN applications. Moreover these studies suggest that providing AP tuning mechanisms could yield performance gains. However, commercial APs are mostly “black-box” commodities that all promise equivalent performance regardless of internal design differences. This leaves academic researchers with little ability to evaluate and understand the internal workings of APs. Hence, the goal is an open source AP that runs on a PC - the Host AP.

The first four steps in building a Host AP are the same as the steps for building a wireless sniffer (see Section II-A.1). The additional steps are:\(^7\)

5. Create an interface configuration file to bring up the wired interface.
6. Configure DHCP to provide the addressing services common to most APs. The DHCP\(^8\) daemon can be most easily configured by modifying its configuration file based on a sample at http://perform.wpi.edu/tools/.
7. Share the connection to the wired and wireless adapters. One of many possible solutions is to use iptables\(^9\) using a sample configuration file from http://perform.wpi.edu/tools/.

Once set up, an end-host wireless client using a commercial AP should be able to transparently associate and use the Host AP, instead. The control of the internal workings of the Host AP allows exploration and understanding of the ramifications of internal AP resource allocation decisions on overall WLAN performance.

B. Tools

B.1 Wireless Layer

For monitoring WLAN performance on end hosts, a real-time monitor tool named WRAP\(^1\) was developed to query information from the end hosts’ IEEE 802.11b/g network device. WRAP\(^1\) was built upon the freely available C++ library named WRAP\(^10\) that provides function calls to gather wireless statistics in Windows XP. The WRAP website provides significant documentation on the features of WRAP, but less information available for building applications that use WRAP. For example, one initial appeal of WRAP was that claims to be “a hardware-independent tool that works with any IEEE 802.11b wireless network hardware vendor”. However, during the developing of WRAP, hardware dependent problem was a major issue as several wireless adaptors could not be fully supported by the WRAP library.

Thus, a contribution is WRAP\(^1\), a tool to monitor wireless statistics in real-time, with guidelines on the wireless adaptors supported (the PrismGT/2.2.5/3 chipsets). When run on an end-host, WRAP\(^1\) periodically (every 500 ms, by default) reports

---

\(^1\)See http://perform.wpi.edu/tools/ for details on each step.
\(^2\)http://www.prism54.org/
\(^3\)http://www.tcpdump.org/
\(^4\)http://www.ethereal.com
\(^5\)http://www.kismetwireless.net/index.shtml
\(^6\)http://www.ethereal.com/docs/dhref/w/wlancap.html
\(^7\)See http://perform.wpi.edu/tools/ for details on each step.
\(^8\)http://www.isc.org/products/DHCP/
\(^9\)http://www.netfilter.org/
\(^10\)http://sysnet.ucsd.edu/pawn/wrap/
IEEE 802.11 network interface statistics, including received signal strength, transmitted frame count, and failed frame transmissions and acknowledgments. WRAPi+ is provided with source code to maximize reuse and enable customization.

B.2 Network Layer

Preliminary experiments revealed that the system ping (using ICMP) provided by Windows XP waits for the previous ping reply or a timeout before sending out the next ping packet. Hence, a constant ping rate cannot always be maintained under poor wireless conditions where 10 second and longer round-trip times can be encountered. Thus, a customized ping tool, UDP Ping, was built using application-layer UDP packets to provide configurable ping intervals and packet sizes. UDP Ping uses a server and a client. The UDP Ping client writes a sequence number and current timestamp into a UDP packet and sends it to the server who echoes the packet data back to the client. The client then reports the round-trip time and calculates the packet loss rate.

WLAN performance is often not symmetric, with network congestion in only one direction or with the upstream flows having more contention than the downstream flows. In these cases, UDP Ping can only report the aggregate network performance for both directions. To measure performance for each direction, a variant of UDP Ping named UDP Heartbeat was developed to report one-way delay and packet loss. The UDP Heartbeat sender writes a sequence number and current timestamp in the UDP packet and sends it to the receiver. Upon receiving the packet, the receiver calculates the time difference and reports any missing packets. In order to measure absolute one-way delay, the sender and receiver clocks must be synchronized. However, often only the difference in one-way delay time, say after network congestion starts, is needed to evaluate performance, allowing UDP Heartbeat to be used even when clocks are not synchronized.

B.3 Application Layer

At the application layer, there are numerous possible tools that drive network performance workloads and provide application-centric measures of network performance. For example, application measurement tools for streaming video such as Media Tracker [15] and Real Tracker [19], can provide a realistic video workload while provide application layer data specific to streaming video including: encoding data rate, playout bitrate, time spent buffering, video frame rate, video frames lost, video frames skipped, packets lost and packets recovered.

A partial list of other application layer tools worth considering include: wget, a publicly-available HTTP/FTP download application that can be used to estimate the effective throughput of a TCP bulk transfer, httpfer, a tool for measuring web server performance, and Iperf, a tool to measure maximum TCP and UDP bitrates, with reports for bandwidth, delay and loss.

III. Case Studies

This section presents three case studies to demonstrate the use of the WLAN measurement tools discussed in Section II. The first case study shows cross-layer performance measurements for a video being streamed over a WLAN to a mobile user (Section III-A). The second case study utilizes a wireless sniffer to characterize the wireless network characteristics of two hand-held video games over a WLAN (Section III-B). The final case study employs a Host AP to assist in understanding queues within an AP (Section III-C).

A. Streaming Video over WLAN

To show the importance of gathering wireless LAN measurements from multiple layers of the protocol and to provide a feel for how the tools we developed can be used in a complementary fashion, this section presents measurements from an experiment where a video is streamed from a Windows Media server to a mobile client over a WLAN. Specifically, this section shows wireless measurement results gathered simultaneously at the application, network and data link layers of the end-host.

A three-minute video clip of a moving Coast Guard cutter was encoded with 11 bitrate layers and streamed over UDP from a Windows Media server, located on the wired part of the WPI campus, to a Media Tracker client on a laptop being carried by a graduate student volunteer. The volunteer was initially standing very close to a wireless AP. After approximately 10 seconds, the volunteer walked slowly away from the AP.

Figure III-A offers a multi-layered snapshot of the data gathered. The top three graphs show application layer performance measurements from Media tracker that include the video encoding bitrate, the received bitrate, and the frame rate, respectively. The middle two graphs show network layer performance measurements from UDP ping that represent the round-trip time and the IP packet loss rate, respectively. The bottom three graphs show wireless layer performance measurements from WRAPi+ that include the target bandwidth level, the retransmission failures, and the signal strength, respectively.

Initially, the highest quality encoding layer, 2.4 Mbps, is selected for streaming by the media server. Consistent with results in [16], the client buffers from 2 seconds until about 18 seconds at a rate much higher than the video encoding rate, peaking around 3.75 Mbps. This suggests that the high playout rate overflows the AP’s downstream buffer from the server to the client and causes significant IP packet losses. This can be seen in the middle graph labeled “Loss Rate” (4th from the bottom) where there is a spike in loss rates up to 20% in the first 20 seconds, even though the wireless signal strength is still quite good.

Near the 10 second mark, video frame playout begins as seen by the graph labeled “Frame Rate” (3rd from the top). As the volunteer moves the client laptop away from the AP, the wireless signal strength steadily degrades, shown in the graph labeled “Signal Strength” at the bottom. However, the wireless

11http://www.gnu.org/software/wget/wget.html
12http://www.hpl.hp.com/research/linux/httpfer/
13http://dast.nlanr.net/Projects/Iperf/
14Note, a mobile wireless client provides a challenging test for measurement tools over a wide range of network conditions.
15Note, the conjecture that the losses are caused by queue overflow could be confirmed by replacing the AP with a Host AP where the queue size can be monitored, as in Section III-C.
target bandwidth remains high except for a brief, abrupt and unexplained change in bandwidth to 2 Mbps at 65 seconds, seen in the graph labeled “Bandwidth” (3rd from the bottom).

At approximately 90 seconds, there is a sudden spike in round-trip time reported by UDP Ping that is accompanied by a considerable increase in IP packet losses, shown in the middle-two graphs labeled “Loss Rate”. The streaming buffer built up at the client is able to smooth over these losses, keep the same encoding rate and maintain a consistently high frame rate.

Just before 100 seconds, the streaming wireless system encounters a major event: the round-trip time spikes again; the wireless signal strength degrades further; numerous wireless frame retransmissions are recorded by WRAPI+ (shown in the “Retry Fail” graph, 2nd from the bottom) and higher IP packet loss rates are seen due either to wireless layer frame drops or to AP buffer overflow.

With a delay in reaction time to this major event, the wireless data link layer lowers its target bandwidth to 6 Mbps (see the third graph from the bottom) and the streaming application changes encoding levels to 1 Mbps (see top graph) just before 120 seconds. Notice, however, that the wireless streaming system has reached a volatile state and the target bandwidth oscillates for the remainder of the test. Even the 1 Mbps encoding rate chosen for streaming is still too high for the effective wireless capacity. This yields numerous IP packet drops and subsequently a low-quality video frame rate. One can conjecture that at 142 seconds, the streaming system mistakenly switches to the original 2.4 Mbps encoding rate possibly due to erroneous capacity estimates from a packet-pair technique.

The use of these measurement tools suggest possible improvements to the streaming system. If the media server had better information on the effective bandwidth of the wireless channel, application throughput, round-trip time and packet loss rate, the server could choose a more effective video encoding layer from among the many choices it could have picked below 1 and 2.4 Mbps. Moreover, this brief analysis demonstrates the value of being able to collect wireless measurements concurrently at multiple layers of the protocol stack.

B. Hand-Held Wireless Network Games

To demonstrate the ability to non-intrusively gather wireless layer traffic from commercial devices for traffic characterization, this section presents measurements from an experiment where video games are played on a WLAN network the newest commercial hand-held gaming platforms, that come equipped with IEEE 802.11 wireless networking for multi-player gameplay.

The traffic generated by one host on a WLAN can have dramatic impact on the performance of other hosts on the WLAN [1], [11]. Specifically, when there is a WLAN host with weak wireless connectivity, and hence low WLAN capacity, the performance of all WLAN hosts is severely degraded. Moreover, some hand-holds are specifically designed to operate at low WLAN capacities to conserve power. These results are especially important since preliminary evidence suggests some game phases for hand-held games may have adverse affects on throughput for Internet applications sharing the same 802.11 wireless channel [7]. To adequately plan WLAN network infras-

Fig. 2. Application layer, network layer, and wireless layer performance of a mobile client streaming a high-quality, multi-layer video clip.
structures, it may be important for engineers to have knowledge of the network load caused by hand-held game traffic.

Three hand-held Nintendo DSes were used to play two different three player games in separate sessions. Sniffing of the 802.11 traffic between the hand-helds was performed using a Dell Inspiron 8600 laptop, running in close proximity (10 feet) to the hand-helds, and with the hand-helds in close proximity with each other. The two games chosen were GoldenEye: Rogue Agent (a first-person shooter) and Super Mario (a third-person action game) to illustrate differences between genres.

Figure III-B graphs bitrate in Kbps versus time in seconds for the two different games. Notice there are two distinct phases of network traffic in each game session. The first phase is marked by high bitrates since the hand-held devices send data to each other as fast as possible. In this case, the data are the games themselves since the devices allow game sharing, but may also include game settings and map information. This setup phase is of different lengths for the two games, with Rogue Agent ending its setup phase after about 25 seconds, while Super Mario does not complete setup until just over 60 seconds.

The second phase is characterized by lower, more consistent data rates. During this phase, the game is actually played, with players responding to the game state and the hand-helds communicating player actions to the other hand-helds, as appropriate. The bitrates during this phase are different for the two different games, with Rogue Agent having slightly higher bitrates (104 Kbps average, 15.6 std dev) than Super Mario (47 Kbps average, 9.1 std dev).

Since the play phase is typically significantly longer than the setup phase and is of the most concern to the players, a 50 second slice of the Play phase for each game is analyzed to provide a uniform sample with which to compare the games.

The accuracy of the QFind method can be checked by instrumenting the Linux kernel in the Host AP to monitor the transmit queue attached to the wireless adaptor. This instrumentation also illustrates the flexibility of an open-source AP. The instrumentation records queue size via a simple user-level application issuing a custom system call to obtain the queue size every 200 ms.

QFind with kernel monitoring was run with the AP queue set to 1000 and then re-run with the AP queue set to 100. Figures 5
begs the question: what size should access queues be? The tools and methodology presented in this paper can be used to explore the impact of access queue size on throughput and round-trip times through wireless measurements.

Figure 5. Host AP with transmit queue length set to 1000 packets.

Figure 6. Host AP with transmit queue length set to 100 packets.

Note that the Linux kernels v2.4 default queue size is 100 packets and Linux kernels v2.6 default queue size is 1000 packets. The differences between access queue sizes for different versions of Linux and perhaps for different commercial APs begs the question: what size should access queues be? The tools and methodology presented in this paper can be used to explore the impact of access queue size on throughput and round-trip times through wireless measurements.

IV. Conclusions

This paper presents open-source software tools and techniques that facilitate exploration of the intertwined effects of WLAN performance across network layers in a flexible, accurate and cost-effective manner. Three examples are presented to illustrate the tools and techniques: 1) a streaming video session using applications, network and wireless layer tools aligns cross-layer performance; 2) wireless sniffing shows network bitrates, frame sizes and inter-frame times for a wireless, handheld game; and 3) data gathered with two queue size parameters demonstrates the use of a host AP. These tools and techniques are useful for gathering real data, both for building more accurate models of WLANs and their applications and for evaluating proposed improvements within a network testbed.

Future tool development such as incorporating new IEEE 802.11 standards (e.g., 802.11e and 802.11n), will allow exploration of state-of-the-art WLAN technology. Techniques and tools to measure and monitor 802.11 ad hoc mode, in particular ad hoc routing, and wire-area wireless technologies, can expand the research scope of the tools and techniques presented here.

REFERENCES


