A Taxonomy of IEEE 802.11 Wireless Parameters and Open Source Measurement Tools

Diego Dujovne, Thierry Turletti, and Fethi Filali

***Abstract*—The analysis and evaluation of new wireless network protocols is a long process that requires mathematical analysis, simulations, and increasingly experimentations under real conditions. Measurements are essential to analyze the performance of wireless protocols such as IEEE 802.11 networks in real environments, but experimentations are complex to perform and analyze. Usually, network researchers develop their own tools, sometimes from scratch, to fit the requirements of their experimentations, and these tools are then abandoned when the paper is published. In this study, we emphasize the importance, for the network research community, to use and contribute to the development of open source measurement tools. In this regard, we propose a survey and classification of IEEE 802.11 wireless parameters and open source tools available to collect or estimate these parameters. We highlight the parameters that can be extracted from wireless traffic probes and those that are available through the driver of wireless cards. Then, we introduce and compare open source tools that can be used to make the measurements, with special attention to the flexibility of the tools and their application scope. Finally, we discuss with several case studies the combination of tools that best suit the needs of the wireless experiments and provide a list of common pitfalls to avoid.**

***Index Terms*—IEEE 802.11, open source software, probes, wireless analyzers, wireless measurements, wireless parameters.**

I. Introduction

IT HAS LONG been recognized that wireless networks play an important role in the access networks at the border of the Internet. The most deployed wireless access networks are those based on the IEEE 802.11 standard [1]. However, these networks are affected by many problems such as exposed and hidden terminals [2] and possible high packet loss due to the unreliable and time-varying nature of the wireless channel. Pre-existing applications that once used to work flawlessly in a wired environment have to be adapted to wireless and new services are rapidly emerging to take advantage of mobility and portable devices. However, the elaboration of new transmission mechanisms, and especially the validation procedure is a complex task to perform. Usually, an analytical evaluation is performed to assess the basic behavior of the protocol. Then simulations are used to study and analyze the new protocol with various conditions in a fully-controlled environment. But simulators generally can not reflect with enough accuracy the impact of composite factors on the performance of transmission protocols such as hidden terminals, capture effect, fading, scattering and interference. So, experimentation is necessary to test the new protocol under more realistic conditions, although this environment offers less controllability than simulation [3], [4].

Within the evaluation process, the wireless experimentation is regarded as the most difficult task to perform because it needs a complex set up and it requires to monitor and analyze a large number of parameters. Some commercial solutions[1] can be applied as Mahanti et al. [6] demonstrate through the analysis of 1 billion wireless frames from a platform installed on the University of Calgary campus. But most often, specific tools have to be developed to satisfy the specific needs of each experiment because of the lack of adaptable tools available for the corresponding experimentation scenario. This leads to the design of programs that can not be used for any other experiment without significant changes, and often with poor or nonexistent documentation to possibly make these changes. As a general case, when the experiment is finished and the results are published, the tool is abandoned and sometimes lost forever. We argue that the development of flexible wireless measurement and analysis open source tools will highly benefit the network research community. Several open source measurement tools for IEEE 802.11 networks already exist, and some of them can be used to run wireless experiments. Few of these tools have been originally designed for research purpose, and most of them have been developed to monitor networks, discover topologies or to identify anomalies and security breaches.

This paper aims to help the reader in choosing the best set of tools to achieve his/her specific needs by providing a survey and taxonomy of available measurement and analysis wireless tools. In particular, we focus on measurement tools for the Linux Operating System such as data acquisition tools, filtering and analysis tools, which form links of a global evaluation chain. The snapshots of tools presented in this survey is current as of November 2008. While the tools mentioned in the paper are all available in the public domain, to the best of our knowledge there is no study available that provides a deeper analysis of existing measurement tools for 802.11-based networks and that discusses their relevance according to the user needs. The objective is to avoid re-inventing the wheel for each new experiment where the desired functionalities may already be present in existing tools (or combination of tools).

Manuscript received 9 April 2008; revised 13 August 2008 and 2 December 2008.

D. Dujovne is currently with Universidad Diego Portales, Chile (email: ddujovne@mail.udp.cl). This work has been done while he was in the Planète project-team, INRIA, France.

T. Turletti is with the Planète project-team, INRIA, France (e-mail: turletti@sophia.inria.fr).

F. Filali is with the Mobile Communications department of Institut Eurécom, France (e-mail: fethi.filali@eurecom.fr).

Digital Object Identifier 10.1109/SURV.2010.021110.00020

[1] A list of commercial monitoring tools can be found also at the "Wi-Fi Planet" site [5].

We also emphasize that selecting the most efficient tools with regard to the target metrics to measure is a critical step to perform prior to evaluating network protocols.

The remainder of this paper is organized as follows. Section II introduces the 802.11 protocol basics and in particular the PHY and MAC layer characteristics. Section III identifies and classifies the most important wireless parameters of the IEEE 802.11 standard. Section IV presents the steps required to perform wireless measurements. Section V introduces open source tools through an exhaustive survey of publicly available measurement tools. Section VI presents several case studies of wireless experiments and some common pitfalls to avoid. Finally, Section VII concludes the paper.

II. Basics of IEEE 802.11

A. 802.11 PHY Layer

IEEE 802.11 gathers together several standards for wireless local area network (WLAN) computer communication, developed by the IEEE LAN/MAN Standards Committee (IEEE 802) in the 5 GHz and 2.4 GHz public spectrum bands [1]. These standards specify both physical (PHY) and medium access control (MAC) layers. In 1997, IEEE defined three kinds of options in the PHY layer, which are an infrared (IR) baseband PHY, a frequency hopping spread spectrum (FHSS) radio and a direct sequence spread spectrum (DSSS) radio. All these options support both 1 and 2Mbps PHY rates. In 1999, two high rate extensions were defined: (1) 802.11b based on DSSS technology, with data rates up to 11Mbps in the 2.4GHz band, and (2) 802.11a, based on orthogonal frequency division multiplexing (OFDM) technology, with data rates up to 54Mbps in the 5GHz band. In 2003 the 802.11g standard was proposed that extends the 802.11b PHY layer to support data rates up to 54Mbps in the 2.4GHz band. This family of wireless standards and several other amendments have been merged in a single document called IEEE 802.11-2007 [1]. IEEE 802.11 is still evolving. For example, IEEE 802.11n is an expected amendment to the IEEE 802.11-2007 wireless standard to significantly improve network throughput over previous standards. It builds on previous 802.11 standards by adding multiple-input multiple-output (MIMO) and 40 MHz operation to the PHY layer, operating on both 2.4GHz and 5GHz bands.

It is well known that 802.11 wireless channels are prone to high error rates and channel variability compared to wired Ethernet [7]. Examples of sources for channel variability include multipath propagation, mobility and time-varying multiuser interference. Indeed, 802.11b/g devices suffer interference from a large number of products operating in the unlicensed 2.4 GHz band. Devices operating in the 2.4 GHz range include microwave ovens, Bluetooth devices, baby monitors and cordless telephones. Since the 2.4 GHz band is heavily used to the point of being crowded, using the less overloaded 5 GHz band gives 802.11a a significant advantage. However, this high carrier frequency also brings problems and the effective overall range of 802.11a is slightly less than that of 802.11b/g. Indeed, 802.11a signals cannot penetrate as far as those for 802.11b because they are absorbed more easily by walls and other solid objects in their path.

B. 802.11 MAC Layer

The 802.11 MAC layer aims to provide access control functions to the wireless medium such as access coordination, addressing or frame check sequence generation. Two different classes of wireless configuration have been defined for 802.11: The infrastructure network, where many stations (STAs) can communicate with the wired backbone through an access point (AP), and the ad hoc network, where any device can communicate directly with other devices, without any connectivity to the wired backbone. In infrastructure mode, an AP works as an authentication and network association device, and can act as a bridge with other networks. A group of STAs coordinated by 802.11 MAC functions is called a basic service set (BSS) in infrastructure mode and independent BSS (IBSS) in ad hoc mode, respectively. The IEEE 802.11 MAC sub-layer defines two medium access coordination functions, the basic Distributed Coordination Function (DCF) and an optional mode called Point Coordination Function (PCF), which is unused in practice.

DCF is an asynchronous transmission mode based on Carrier Sense Multiple Access with Collision Avoidance scheme (CSMA/CA). Collision detection can not be implemented because, due to the nature of the channel, a station is not able to transmit and listen at the same time [7]. Actually, two different carrier sensing mechanisms are used: PHY carrier sensing at the air interface and virtual carrier sensing at the MAC layer. PHY carrier sensing detects the presence of other STAs by analyzing all packets received from other STAs. Virtual carrier sensing is optionally used by a station to inform all other stations in the same BSS (or IBSS) how long the channel will be reserved for its frame transmission. The sender can set a duration field in the MAC header of data frames, or in the Request-To-Send (RTS) and Clear-To-Send (CTS) control frames. Then, other stations will update their local timers of network allocation vectors (NAVs) to take into account this duration. The RTS/CTS mode is often used to reduce collisions in presence of hidden nodes [8].

IEEE 802.11e-2005 (or 802.11e) is an amendment to the IEEE 802.11 standard that defines a set of quality of service (QoS) enhancements for WLAN applications through modifications to the MAC layer, and has been incorporated into the IEEE 802.11-2007 specification. In order to provide queue based QoS support, a new MAC layer coordination function has been proposed, called hybrid coordination function (HCF). Within the HCF, two interoperable methods of channel access are defined: HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). HCCA is based on polling, while EDCA is based on a slotted and highly parametric CSMA/CA protocol. The 802.11e amendment is important for delay-sensitive applications, such as voice over wireless IP and multimedia streaming.

C. Structure of 802.11 Frames

All of the 802.11 frames share the same basic PHY level structure: a preamble to train the receiver followed by a Start of Frame (SOF) delimiter; a Physical Layer Convergence Procedure (PLCP) header and the payload called the MAC Protocol Data Unit (MPDU). The PLCP carries the signal field

TABLE I
COMMON DLT HEADER VALUES

Parameter	Description
Timestamp	Packet arrival time. At the radiotap header, there is the MAC timestamp, which corresponds to the arrival of the first bit of the packet. libpcap [9] adds another timestamp extracted from the system clock, after the packet has arrived to the kernel level.
RSSI	Received Signal Strength Indication. Measures the average receiving power of the packet. The RSSI value and bounds varies between implementations [12].
Channel	Channel number where the packet was transmitted.
Noise level	Characterizes the background noise level measured before packet reception on the channel at the receiver.
Data Rate	Physical bit rate of the packet payload.
Preamble	PLCP preamble length (short or long).

containing the payload data rate, a service field describing modulation characteristics, the length of the payload in microseconds and the Cyclic Redundancy Check (CRC) of the PLCP header. The preamble and the PLCP header are transmitted at 1Mbps regardless of the current data transmission speed. Three types of MPDU exist: Management, Control or Data.

III. A TAXONOMY OF WIRELESS PARAMETERS

Wireless parameters are essential to monitor the network, to detect possible anomalies or to analyze deeply the behavior of network protocols. A very large number of wireless parameters exist at different levels of the protocol stack, and are available either from the wireless card drivers or through packet traces. In this section, we propose a classification of most common wireless parameters that distinguishes between per-packet, per-flow, per-station and per-BSS wireless parameters.

A. Per-Packet Wireless Parameters

Per-packet wireless parameters are those included in PHY and MAC headers of each packet and are available through packet sniffing. The following two subsections detail the most important PHY and MAC information present in IEEE 802.11 frames sniffed from the wireless medium.

1) PHY data information: As we mentioned in section II-C, every 802.11 frame starts with a training sequence, a Start Of Frame (SOF) marker and a PLCP header. When a sniffer captures a frame, the three of them are removed on the hardware interface before the frame arrives to the driver, and an artificial header, called DLT (Data Link Type), is added instead. Different formats of DLT headers exist and they are defined in the pcap [9] library. DLT headers include PHY-level information captured by the network interface card. Note that if some PHY-level parameters are not supported, the user still has the possibility to add the missing features himself. However, such changes require OS kernel development skills and assume that sufficient chipset documentation is available.

The different types of DLTs specified in the pcap library include a variety of media like fiber optics, PPP serial links, ethernet and wireless networks. For 802.11 networks, the most popular ones are AVS[2] [10], PRISM[3] and radiotap [11]. These three DLT types share some common parameters shown in Table I. In the remainder of the paper, we focus on the radiotap header type because it provides more features than the other header types. The radiotap header type consists of a standard preamble followed by an extensible bitmap indicating the presence of optional capture fields packed into the header as compactly as possible. This typically includes information such as rate, channel number, signal at per-packet basis RSSI (Receive Signal Strength Indicator measured for the PLCP header by the circuitry on the wireless network interface card [12]) and timing information (a 64-bit field in microseconds indicating when the first bit of the MPDU arrives at the MAC).

2) MAC data information: On top of the physical layer, lies the MAC layer. Table II identifies the main fields included in the MAC header of IEEE 802.11 frames, see [1]. We also present a use example for each of them.

B. Per-Flow Statistical Parameters

Per-flow statistical parameters are essential to analyze the performance of applications on top of IEEE 802.11 wireless networks, or to study the fairness between the different stations. They are general industry-accepted statistical parameters obtained by processing packet traces sniffed on the wireless medium, i.e., by analyzing packet presence, packet headers and arrival time.

Table III presents a non-exhaustive list of per-flow statistical parameters with a use case example for each of them.

C. Per-Station Statistical Parameters

Each station may have several active flows in the same time. It is sometimes useful to analyze the performance of all flows that belong to the same station. Per-station statistical parameters correspond to the aggregation of per-flow statistical parameters detailed in the previous subsection, plus statistical parameters including:

- active flag: true if the station has at least one flow running,
- association state: associated or disassociated,
- association duration,
- MAC address of the current associated AP,
- Current Received Signal Strength Indication (RSSI) [15], [12],
- Current uplink physical transmission rate.

D. Per-BSS Statistical Parameters

Per-BSS statistical parameters include the aggregation of per-station wireless parameters, and other statistics such as the parameters described in Table IV. These wireless parameters are used to monitor and to analyze the characteristics of the whole channel.

[2]AVS: (AbsoluteValue Systems, Inc).

[3]Originally designed by Intersil Inc., PRISM Wireless LAN business is now part of Conexant Systems Inc.

TABLE II
MAC HEADER STRUCTURE

Parameter	Description
Type/Subtype	These two fields together identify the function of the received frame. They are useful to classify the traffic between management, control and data frames. Each of the frame types has several defined subtypes.
More Fragments field	Set to 1 when it is not the last part of the frame. This indicator is useful to reassemble a sequence of fragments and to compute fragment loss.
Retry field	Set to 1 if this frame is a retransmission of an earlier frame. This bit helps in eliminating duplicating frames and enables many measurements on frame loss, since the frame is retried a fixed number of times.
Power Management field	Indicates when the sender toggles to power save mode. The behavior of the Access Point changes when at least one of the stations is in power save mode. In particular, multicast and broadcast packets are transmitted only after fixed periods and unicast packets are buffered for the station.
More Data field	Set to 1 if there are buffered unicast frames for the destination station.
Protected Frame field	Set to 1 if the frame body field contains information that has been processed by a cryptographic encapsulation algorithm. Encryption is systemwide, for monitoring and statistics purposes, while decrypting is only required if the payload is of interest.
Duration	The contents of this field vary with frame type and subtype. For data frames, if the More Fragments bit is set to 1, and the destination address field contains an individual address, the duration value is set to the time, in microseconds, required to transmit the next fragment of this management frame, plus two ACK frames, plus three SIFS intervals. To calculate the medium idle time, the duration value must be taken into account.
Addresses	Identify the basic service set identifier (BSSID), source address, destination address, transmitting station address and receiving station address of the frame. Certain frames may not contain some of the address fields. This information is useful to identify the different stations and to classify the individual flows. Roaming stations can also be identified when starting sessions on different APs.
Sequence Control	Carries a sequence number and a fragment number. The former can be used to identify lost frames and to detect intrusions and/or failures while the latter indicates which part of the packet is being carried by the current frame.
Frame Check Sequence[4]	Trailer of the frame, it is used to verify the integrity of the frame through a CRC. Frames with errors can be logged to identify the source of lost frames.
Backoff[5]	Derived from the difference between the timestamp values of successive frames, this parameter can be used to detect MAC layer misbehavior [13].

IV. STEPS TO RUN WIRELESS MEASUREMENTS

The most important source of information to analyze wireless networks are statistics inferred from packets sent on the channel. They are obtained using packet sniffers, which are stations that passively listen all the packets on the medium. Furthermore, statistics provided by the drivers of wireless cards can be used as a complementary source of information, and the range of statistics available depends on the evolution state of the driver. The choice of open-source drivers as madwifi [21], enables the instrumentation of the driver and also the sharing of the source code with other laboratories in order to validate the experiments. More complete PHY and MAC layers capture data can be obtained with hardware specific platforms like GNU Radio [22] or WARP [23], but these fall out of the scope of this publication.

In the following, we describe the different steps involved in doing wireless measurements.

- **Sniffing:** The packet sniffing task consists in retrieving all frames transmitted on the wireless medium. It can be divided in two stages: The first one is the hardware side, which depends exclusively on the ability of the wireless cards to detect, decode, buffer and transfer the packets to the PC bus (either through PCI, PCMCIA, USB, PCI express or whichever standard communications bus is used). The second one is the software side, where the driver sends a copy of any received or transmitted packets to a part of the kernel called the packet filter. By default, all the packets are then copied from the kernel space to the user space where the sniffer is actually running. To create a sniffer, the driver of the wireless card must be configured in *monitor* (also called *promiscuous*) mode. Each sniffer generates an event log (or a packet trace) composed of all packets sniffed[6] on the wireless channel.
- **Merging:** Producing an accurate packet trace requires great care. In the wireless domain, spatial diversity prevents any single sniffer from capturing the overall traffic. Thus, many spatially dispersed sniffers are required to reconstruct all the traffic. Using too few packet sniffers, placing them poorly, or using inadequate hardware can introduce missed or reordered packets and incorrect times-

[4]The FCS parameter is not part of the MAC header. It is the last field of the MAC frame and follows the Frame Body field.

[5]The backoff parameter is not included in 802.11 MAC header, it is computed by software.

[6]For obvious privacy reasons, the payload of packets should be discarded before building up packet traces.

TABLE III
PER-FLOW STATISTICAL PARAMETERS

Parameter	Description
Goodput	Measures the packet arrival rate during a fixed period of time at the application level. It can be used to evaluate the quality perceived by the application.
Data loss rate	Measures the number of data frames lost during a period of time, it is the inverse of the data delivery ratio. The loss can be due to transmission errors (e.g. noise, interference), buffer overflow or collisions. It can be used to estimate the link quality, the quality perceived by the application or as feedback to adaptive wireless-aware protocols.
Data loss burstiness	Measures the number of packets which are lost consecutively. This parameter can be used to measure the quality of a link or to tune the error correction algorithms [14].
Delay	Measures the latency at the application layer for a frame from departure at the transmitter to the arrival at the receiver. The delay is a consequence of queuing, packet retransmissions and packet transmission on the medium. It can be used to estimate the channel load or to evaluate performance of real time applications.
Jitter	Estimate of the statistical variance of the data packet interarrival time. High variability may harm the protocol stability and performance; It can be the result of packet loss bursts, after successive retransmissions. Same use cases as for the delay.
Airtime	Stands for the effective transmission time on the medium. This parameter is used as a fairness metric between the flows that share the same wireless channel.
Retransmission probability	It is function of the packet loss rate observed for both data and acknowledgement transmission. It can be used to estimate the link quality.

TABLE IV
PER-BSS STATISTICAL PARAMETERS

Parameter	Description
Channel capacity	Theoretical maximum traffic rate at the physical layer. For example, it is equal to 11Mbps for IEEE 802.11b and to 54Mbps for IEEE 802.11a.
Overall data throughput	Aggregated throughput of all data packets transmitted on the medium. It can be used to estimate the channel load.
Overall signaling throughput	Aggregated throughput of management and control frames (such as beacon, RTS/CTS/ACK) of all packets transmitted on the medium. It can be used to compute the channel overhead.
Packet loss spatial correlation	Identifies the packet loss related to the position relative to the AP and the other attached stations. This information is available from the correlation between packet logs (source and receiver probes) and from the wireless driver statistics logs (packets lost at the sending queues are not considered). It can be used to analyze and improve the performance of multicast transmission protocols [16], [17].
Load level	Number of packets present in the wireless medium per time unit. It indicates the level of medium usage.
Available bandwidth	Corresponds to the rate at which a new flow can send traffic without affecting competing flows. Different algorithms have been proposed to estimate the available resources, see [18], [19], [20].

tamps. When multiple sniffers are used, the independent traces have to be combined and synchronized down to microsecond granularity to construct a synchronized single trace of all frame transmissions [24], [25], [26], [27].

- **Processing:** Once an accurate packet trace is ready, the actual processing can start. First, if some packets present on the trace are not relevant for the analysis, they can be filtered out from the packet trace to speed up the following computations. Then the parameters to analyze can be extracted, possibly combined with other sources of statistics (like the ones provided by the wireless card drivers). Various computations can follow, such as average calculation and the result can be displayed to the user.
- **Monitoring:** If the processing task is done in realtime, i.e. when sniffing and processing operations are done simultaneously, it is called *monitoring* in the remainder of the paper. Monitoring is useful to analyze in realtime the network behavior, for example to detect network anomalies. It can be implemented using a single packet sniffer combined with a simple packet analyzer.

V. TOOLS FOR WIRELESS MEASUREMENTS

This section proposes a classification of the main open source tools that can be used to characterize wireless experiments. Wireless measurement tools come in three different flavors: Adaptations or derivatives from wired measurement tools, like Wireshark and Mognet; Monitoring tools specific to the wireless environment like Kismet, Wifiscanner and Wavemon; and tools which target experimental wireless measurements, like Airtraf, Jigsaw or WisMon. A tool is required for each of the tasks we have defined on section IV. Furthermore, some tools can be used for different purposes; for example, Wireshark can be used for capturing and to do filtering as a processing task. At the end of the section we

provide a classification where each of the tool is assigned to one or multiple tasks to execute during an experiment.

In the following, we distinguish between tools that retrieve wireless parameters through the driver of wireless cards from tools that capture and/or process logs of packets sniffed from the wireless channel. We present only the most essential features of each tool. The reader can find further technical details in the references provided in the text.

A. Tools Based on Driver-Level Statistics

As we mentioned in Section IV, the drivers of wireless cards can be used as a complementary source of statistical information. The madwifi driver [21] is currently the most advanced open source driver available for Atheros-based wireless cards. It uses the Wireless Extensions for Linux and has companion applications to simplify the configuration and the statistical data capture. It allows to print the internal PHY and MAC events to the system log through the athdebug and 80211debug utilities respectively [28]. Also, the driver keeps internal statistics which can be accessed through the athstats [29] utility for the PHY related statistics, and using the 80211stats utility for the MAC statistics. Although the madwifi driver itself is open source, it depends on the proprietary Hardware Abstraction Layer (HAL), which is only available in binary form. However, it is currently evolving to a full open source driver called ath5k, which does not depend on the HAL.

Furthermore, some devices store these statistics in management information bases (MIBs). In this case, the statistics can be retrieved using Simple Network Management Protocol (SNMP) tools [30]. However, note that SNMP statistics can sometimes be inaccurate and should consequently be used with caution [31].

Wireless Tools for Linux: The Linux Wireless Extensions (WE) and the Wireless Tools (WT) [32] form a generic API allowing a Linux driver to expose to the user space configuration and WLAN statistics. This set of tools creates a uniform interface at the user space to configure, control, query and debug the wireless interfaces. A typical usage example of wireless tools is the access to the aggregate data statistics using iwconfig, the setting of specific driver-level parameters with iwpriv and the listing of the results of access points in range. They use a very basic textual interface and are included by default on most Linux kernels. However, different or invalid implementations of these utilities may lead to erroneous results, so these statistics should be compared with known statistics to guarantee that the results are coherent. The Wireless tools for Linux are distributed as open source code under the GNU Public Licence (GPL).

WaveMon: WaveMon [33] is an example of a wireless monitoring tool that uses the Wireless Extensions for Linux. It is a *ncurses*-based lightweight wireless monitor tool that can be run with or without a GUI. It supports devices with low processing power and low resolution display and allows to watch in realtime the signal and noise levels, packet statistics and wireless card configuration. Two different views are available: A snapshot of the current state of the wireless link statistics, and a historical graph for the same parameters. During an experiment, Wavemon can provide basic insight about the link quality from the signal power side. It can not be used for batch mode experiments because packet logging is not available; all the processing is done in realtime. Current version is 0.4.0b and it is distributed as source code under the GNU public license for the Linux operating system. A typical application of Wavemon is monitoring during the experiment.

WRAPI: WRAPI [34] is not a tool but a hardware-independent library that allows applications to access MAC-layer information. It requires an application on top of it to call the functions, and execute data capture. So it is not functional by itself, but it is very useful to perform measurements on a Windows platform, and, to our knowledge, it is the only open source package that works on Windows XP. WRAPI uses the NDIS user mode I/O protocol to communicate from the user-mode side to the driver. Using this communication protocol, it is possible to query information and set parameters. A limited number of parameters is available, including wireless configuration, packet level statistics, current MAC address, signal strength and AP information, but per-packet information is not available. It is worth mentioning that the statistical information provided by the driver for part of parameters is preprocessed internally, e.g. to calculate running average. As preprocessing operations affect the measurement results, the fact that these operations are not documented and the corresponding source code is not publicly available is problematic to perform rigorous analysis. This library was developed for Windows XP exclusively, and Version 2.0 of the source code is available on the WRAPI website [34] without any license information.

Finallly, Figure 1 illustrates the position of driver-level statistic tools within the Operating System.

B. Measurement Tools Based on Packet Traces

In this section we discuss relevant open source tools that can be used to perform the measurement tasks described in Section IV, i.e., sniffing, merging, processing and monitoring. As most tools implement more than one of these tasks, it is difficult to classify them based on their functionalities and we have chosen to present them in the alphabetical order. To recapitulate, Table VI gives a snapshot of the main functionalities of each tool at the end of the section.

Airtraf: Airtraf [35] is a wireless analyzer that gathers accumulated wireless statistics from each station and for each TCP flow. It can be used to monitor wireless statistics during an experiment, such as packet count, byte count, bandwidth usage and signal strength. It can also analyze the state of APs and the associations between stations and APs. Airtraf is sniffing-based and depends on underlying libraries to capture packets. There is no graphical interface available as open source for the polling server. Airtraf allows to create a "near realtime" picture of the wireless parameters, access points, stations present in the BSS and on TCP flows transmitted between them. This information is presented either as a cumulative (for packet or byte counts), mean value (e.g. power value) or instantaneous value (e.g. current bandwidth usage). Unfortunately, there is no historic log nor graphical user interface display to analyze the behavior of parameters. Furthermore, there is no event log to follow the transactions from the management packets.

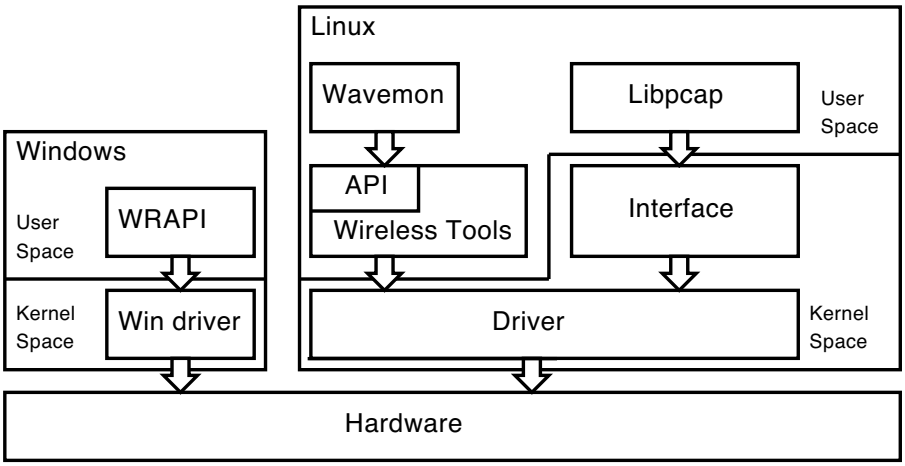

Fig. 1. Position of driver-level statistics tools

Version 1.1 of Airtraf for Linux is available under the GNU public license, but a commercial branch of this tool is available from Elixar Inc.

EasySnuffle: EasySnuffle [36] is a measurement tool composed of a collection of modules to insert on the kernel, device drivers and user space. These modules act as probes at MAC, IP and UDP layers and have been designed for analyzing performance of multimedia transmissions over wireless networks. The basic setup includes a specific wireless device driver for the Prism2 chipset, a custom modified kernel and a user application. The main advantage of this tool is the possibility to instrument the system at different levels of the communication stack. However, the last version of the tool was released in 2002 and the modules lack flexibility to evolve to a newer version of the kernel. The source code is available for the Linux operating system at the Snuffle website [37]. Although this tool is rather outdated, we have included it as an example of instrumentation through all the measurement chain. Using this type of instrumentation, it is easier to do a cross layer analysis since all the data is captured on the same time basis.

Jigsaw: Jigsaw [26] is a multi-sniffer tool for infrastructure wireless systems that combines the packet traces to generate a comprehensive view of events taking place in the network. It is used to fulfill the sniffing and merging tasks. To synchronize the time across traces gathered by multiple sniffers, Jigsaw identifies frames that are overheard by multiple (but not all) monitors. It tackles the problem of clock drifts (the change in skew over time) using an exponentially weighted moving average of past skew measurements to predict future skew on a per-instance basis. Jigsaw can be used offline to gather and synchronize all the collected traces from an experiment. The output is composed of a single file including all the packets collected during the experiment. This tool allows to reconstruct a complete description of all link- and transport-layer conversations. Some inference techniques are used to deduce the presence, time placement, and even contents of missing data. Version 2.4 of this tool is available under the GPL license from the Jigsaw website [38] for the Linux operating system.

Kismet: Kismet [39] is a monitoring tool used to discover wireless networks. It creates a list of available access points in the selected channels and a list of attached stations using information contained on collected packets. For each item, it provides detailed information such as the addresses, traffic and station activity. The lightweight text-based interface allows the user to build a monitoring system without loading a graphical server. The main advantage of this tool lies in its ability to recover and gather in a smart way per-station traffic and fill station information structures with the results. This tool can also provide GPS information about scanned APs using the *gpsd* open source tool. Flexibility of Kismet comes from the client-server architecture, built as a server processing engine and a client, interconnected with a proprietary protocol. The client (and possibly multiple clients) can be run on a different machine than the server. This tool does not provide any statistical analysis tools, although it can be used as a source of disassembled packets for data capture. Since Kismet includes a packet dissector, an application can be built on top of it to retrieve only the relevant parameters, which can be selected in realtime through the client-server communication protocol. Kismet runs on Linux with a large number of cards and can also run on Windows but only on cards that support AirPcap from CACE Technologies. It is licensed under the GPL and current version 2007-10-R1 is available at the Kismet website [39].

Mognet: Mognet [40] is a basic sniffer and analyzer tool that has been designed for personal digital assistants (PDAs) that support Java. It captures packets in realtime and disassembles the 802.11 headers using either a graphical or a text-based interface. Mognet can also generate output in libpcap format. The main objective of this tool is to display and disassemble packets with no further analysis, but with the possibility to examine packets collected in realtime. It does not include processing and analysis functionalities because the processing power is still limited in PDAs. It is inspired by Ethereal. Since this tool can generate trace logs, it can be integrated within experiments where merging and postprocessing are done in batch mode. Although Mognet has not evolved since 2003, the current 1.16 version still works on any wireless cards that support the monitor mode. Mognet is available under the GNU

public license and works on any PDA that includes a Java interpreter and a C compiler.

Pcap: The pcap library[7] [9] is not a tool but since it is the core of many tools such as tcpdump and kismet, it is worth mentioning it in this section. Libcap is an open source library that provides a high level interface to network packet capture systems. The goal of this library is to create a platform-independent API to eliminate the need of system-dependant packet capture in each application, as every OS vendor may implement its own capture mechanisms. As we mentioned in Section III-A1, different types of DLTs are specified in this library, including a variety of media like fiber optics, PPP serial links, ethernet and wireless networks.

Wifiscanner: Wifiscanner [41] is an analyzer/detector of IEEE 802.11b STAs and APs which can listen alternatively on all the 14 channels. It can be used to monitor parameters during an experiment and also for sniffing. Wifiscanner also includes an integrated IDS (Intrusion Detection System) to detect anomalies like MAC usurpation. A basic text-based interface is provided for passive sniffing operation. This tool provides the user a realtime packet disassembly of the 802.11 header. It also keeps cumulative values of the observed packet types at the MAC level. A list of current stations can be displayed with the number of transmitted packets. All network traffic can be saved in the pcap format for post analysis. Current version 1.02 is available under the GNU public license for Linux operating system [41].

Wireshark: Wireshark [42] (formerly Ethereal) is the de facto open source network protocol analyzer. It integrates a general-purpose packet sniffer and packet analyzer tool for almost any type of network. It includes a very powerful packet dissector and classifier tool and currently supports more than 700 different protocols. Specific filters can be built for each field of the captured packets. However, Wireshark lacks post-processing and analysis tools, providing only basic statistics and graphs. The main packet list uses coloring filters, which helps to identify faster which type of packets are present. It also includes decryption support for many protocols, including IPsec, WEP and WPA/WPA2. Packet timestamps come from the Winpcap- packet capture library, which is independent from Wireshark. Version 1.0.2 of Wireshark is available under the GNU GPL and runs on Windows, Linux, MAC OS X, Solaris, FreeBSD and NetBSD operating systems [42].

Wismon: Wismon [43] is a packet analyzer tool that has many useful functionalities for a wireless experimental usage. It provides physical parameters in realtime for evaluation during experiments and allows to record logs for further processing. It can be used for building a single view of the whole wireless communication channel. Wismon uses multiple probes running Kismet (version 2004-10-R1). Using a kismet patch, probes synchronize the timestamp with the beacon timestamp at the probe level, providing ready-synchronized packets. The Wismon tool uses a client-server architecture. The server creates a single list of packets and discards duplicated packets heard by the different probes. This list is thoroughly analyzed and the packet headers are classified per station. The Wismon client shows the list of the current stations at the server and displays for each station its characteristics in real time. Version 0.1.R3 of this tool is available under the Cecill[8] license for the Linux operating system from the Wismon website [43].

Wit: Wit [27] is MAC analyzer tool for IEEE 802.11 networks that includes a distributed passive sniffing mechanism. It collects traces obtained from multiple and independent passive sniffers and stores them in a common database. Wit can be used to perform the merging task of wireless experiments. Three processing steps are done to construct an enhanced trace of packets. First, a robust merging procedure combines the necessarily incomplete views from multiple, independent sniffers into a single, more complete trace of wireless activity. Next, an inference engine based on formal language methods is used to reconstruct packets that were not captured by any sniffer and to determine whether each packet was received by its destination. Finally, it derives network performance measurements from this enhanced trace. Wit is available online [44] as perl scripts that process data traces.

CRAWDAD repository of tools: The CRAWDAD [45] (Community Resource for Archiving Wireless Data At Dartmouth) website contains a repository of basic scripts and tools to process packet logs and SNMP statistics. Various scripts can be used to extract fields and flags from individual packets, to create a list of stations from the packet logs, to anonymize the packet traffic (in order to be published later in the public domain), and to estimate the location of devices. Most of these tools and scripts tools are available under the GNU public license (or some variations of it) for the Linux OS.

Snapshot of Tools: Table V-B and Table VI provide, respectively, the input/output formats and a snapshot of the main characteristics for each of the tools described above.

VI. Use of Tools

This section presents a set of case studies to demonstrate the use of tools discussed in Section V. We propose for each scenario the combination of tools that best suit the needs of the experiment. Then we present a list of common pitfalls to avoid.

A. Case Studies

In the following, nine examples of measurements are presented. We have selected the tools according to their pertinence and their practical approach: e.g., for configurability and standard logging format, the use of tcpdump file format for packet logs. In most cases, custom scripts that fit the wireless experiment needs have to be used jointly with the measurement tools. Driver-level tools like Wireless Extensions for Linux and WaveMon presented on Section V-A are independent from the packet capture tasks. So, they can be used on any of the case studies below to provide snapshots of PHY and MAC statistics.

- **Throughput:** Throughput measurement consists in counting the amount of bits transmitted per second between the AP and each of the stations. In this experiment,

[7]The pcap library is called libpcap on Unix-like systems and Winpcap on Windows systems.

[8]see http://www.cecill.info/ .

TABLE V
I/O FORMATS OF WIRELESS MEASUREMENT TOOLS

Input Method/Format			Tool	Output format				
Packet Capture	Pcap	Wireless Tools for Linux API		Text UI	Graphical UI	Text	Pcap	DB
		√	Airtraf	√				
√			Easysnuffle	√	√			
√	√		Jigsaw				√	
√	√		Kismet	√		√	√	
√			Mognet				√	
	√		Wavemon	√				
√			Wifiscanner	√			√	
√	√		Wireshark		√	√	√	
√			Wismon		√	√		
	√		Wit					√

the wireless traffic is captured using several wireless probes, which are configured in promiscuous mode. We propose to use Wireshark to capture and filter packets for each station, because it provides two usage modes: In the first place, a test can be done using the graphical user interface of Wireshark, where the wireless traffic can be observed while it is being captured. This helps to check if anomalies occur on the captured traffic (like periodic disconnection of an Access Point), which may void the experiment. In the second place, and in order not to interfere with the environment, the user can execute remotely a command-line version of Wireshark, called Tshark. If the analysis is targeted to a single or multiple flows, packet filtering is also possible using Wireshark as a postprocessing engine. After obtaining the filtered packet list, the user can execute a custom script to extract the packet description header containing the packet size and the timestamp, in order to compute various statistics such as throughput. WisMon can be also used to monitor the experiment remotely, using either a new probe or one of the probes used by Wireshark.

- **Contention Window:** The contention window size corresponds to the period between the end of the SIFS/DIFS and the start of the next packet. Note that the DLT header of each packet contains the receiving timestamp. For practical reasons, we propose to use Wireshark to capture the packets, and the algorithm described by Berger et al. to measure packet interarrival times with high precision [46]. This type of measurement only requires filtered logs of packet lengths and their timestamps, which are present in the DLT header. A custom script is used to extract timestamps and packet sizes. No merging is required since traffic is light and the probe is near to the source. Kismet can be used for monitoring and detecting any unexpected events. Indeed, in this scenario monitoring is only useful to identify possibly foreign packets coming from nearby stations not participating on the experiment.
- **Airtime:** As mentioned in Section III-B, the airtime measures the medium utilization from the calculation of transmitted packet length. This is useful to analyze the application-level throughput relative to usage of the physical wireless channel, see [47], [48]. We propose to use Jigsaw for the capture, synchronization and merging operations because for this experiment, we need the most complete view of the wireless environment, which can be provided by a single list of all the participating packets. Then, a custom script can be run to calculate the airtime, by extracting from the packet logs the timestamp, the packet length, the type of preamble, the duration and the transmission rate per packet. Monitoring can be performed using Airtraf as well as Wismon to watch the overall throughput from each of the concurrent sources. This solution is better than using separate tools for each task, because the overall processing can be done within the merged packet log.
- **Spatial packet loss:** In this experiment, the goal is to estimate the correlation between sent and received packets with respect to the position of stations. The pattern of bit errors changes according to signal reflections, diffractions and interferences, which may affect neighboring stations. For instance, such a behavior can be caused by interference created by an AP present on a nearby channel. We propose to use Wireshark to collect the packets and to only keep in memory those which belong to the flows under study. In this case, merging and synchronization operations must not be done because it is critical to analyze the same packets captured on different probes. Then, packet loss correlation can be computed on packet logs using a custom script that analyzes the packet sequence number to find out which packets are missing for each source. In such a simple algorithm, a packet lost by all the stations is suspected to be a collision while a packet lost only by individual stations is considered as noise. On the other hand, a packet lost by more than one station (but not all of them) is suspected to be the outcome of a local effect between stations of the same characteristics (region/distance). Note that heavy traffic conditions can generate packet drops before transmission on the wireless medium. This type of loss could be badly interpreted as collisions on the wireless medium. So, we recommend to monitor queue drop count both at the AP and at the STAs: such information can be obtained from specific instrumentation of the wireless card drivers. Monitoring can be achieved using Wismon, which provides per-station statistics for retransmissions and traffic.
- **Analyzing capture effect:** The capture effect, also called

TABLE VI
CHARACTERISTICS OF WIRELESS MEASUREMENT TOOLS

Tool	Sniffing	Merging	Processing	Monitoring	Comments	Platform
Airtraf	No	No	No	Throughput, Text-based interface	New version became commercial, including the web interface	Linux
EasySnuffle	Single probe	No	Statistics extraction	No	Driver and kernel instrumentation	Linux
Jigsaw	Multiple probes	real-time merging	Filtering, Traffic reconstruction	No	To sync, uses exponentially weighted of past skew measurements	Linux
Kismet	Multiple probes	No	No	Throughput, Text-based interface	Very popular and advanced wardriving tool	Linux and Windows (restricted)
Mognet	Single probe	No	Filtering	No	Java based, targeted for portable devices	Multiplatform (tool mostly written in Java)
Wavemon	No	No	No	Statistics calculation, Text-based interface	No logging	Linux
Wifiscanner	No	No	No	Network structure graph generation	Wardriving oriented	Linux
Wireshark	Single probe	No	Filtering and flow analyzer	No	Very extensive protocol decoding	Multiplatform
WisMon	Multiple probes	Real-time merging	No	Data classification per source, recent history buffer	Packet logging and offline analysis	Linux
Wit	No	Offline merging	Filtering, Traffic reconstruction	No	Has a formal language to describe conversations between hosts	Multiplatform (tool mostly written in Perl)

co-channel interference tolerance, is the ability of certain radios to correctly receive a strong signal from one transmitter despite significant interference from other transmitters. In other words, a frame with the highest received signal strength can be successfully decoded at the receiver in presence of simultaneous transmissions of several stations. Lee et al. have studied this problem using a testbed with the aim to capture as many collisions as possible [49]. For each collision, the timestamp, signal strength and bit rate parameters have been analyzed. We propose to use Jigsaw to perform packet capture, traces merging and synchronization, since here we need again the global view of the traffic, and also the ability to discriminate the existence of detected and undetected collisions. After the construction of a single list of packets, a custom script will serve to analyze the superposition of packets and to detect possible capture effect by inspecting timestamps and packet lengths. The experiment can be monitored either using WisMon to display a graphical view of the traffic involved with the measured power, or Airtraf, to get the mean values during the experiment.

- **Impact of Rx power on packet loss:** This experiment aims to study the sensitivity of a flow to variations of the receiving power (and consequently SNR). The experiment layout is simple: One AP sends data at decreasing power levels with fixed rate and constant packet size to three receivers placed at different locations within range. This experiment helps us understand the performance of packet-level power control to increase power efficiency on portable devices. For practical reasons, we suggest the use of Wireshark to control the packet capture and to filter data. Then, a custom script can be used to compare the sequence number between the transmitted and received packets in order to compute the packet loss. We also propose WisMon for monitoring, as this tool can provide per-second received power information.
- **Impact of Tx rate on packet loss:** The objective of this experiment is to measure the efficiency of the physical

rate selection algorithm using packet loss information. We keep the same layout than for the previous experiment: one transmitter AP sends packets with decreasing rates keeping a fixed transmission power level and a constant packet size to three receiver stations at different locations within range. But for this experiment, we propose different tools: Wireshark for capturing and filtering and Wit to create a database with the physical parameters of the transmitted packets. Finally, a custom script will use the sequence number from the database and the packet rate to analyze the percentage of packets received for each transmitted rate. The most important task to monitor this experiment is to ensure that the packet transmission from the source does not stop, and this can be done with either Kismet or Airtraf.

- **Impact of mobility:** Mobility has critical effects on packet loss, delay and throughput because it generates higher variations of the channel characteristics, increasing the bit error rate. To measure the impact of mobility, the traffic received at each mobile station has to be captured. We propose to use the combination of Mognet and Wit tools: Mognet to collect traffic received on stations, and Wit to analyze and reconstruct traffic patterns. Although Wit was not initially intended for mobility environments, its traffic inference engine can be helpful in analyzing very lossy channels. The experiment can be monitored with Kismet using a fixed transmitter or receiver, to ensure that there is no unexpected anomaly (such as a disconnection or a malfunctioning station).
- **Flood attack detection:** Flood attack is a common threat for APs. Although recent APs are ready to manage such attacks, there are still many devices that can be severely affected by them. To detect this type of attack, we propose to use the combination of Wifiscanner and Wireshark tools: Wifiscanner for monitoring and detecting traffic pattern of a flood attack, and Wireshark as a general purpose tool for capturing and storing the traffic. Then, a custom script can be used offline to identify potential sources of attack.

Table VII regroups the above examples of wireless measurements and recapitulates the corresponding recommended tools.

From the former examples of wireless measurements, several patterns arise: Wireshark is essentially a sniffing and filtering tool. Packet logs can easily be analyzed, merged and synchronized using Wit. Jigsaw covers the capture, merging and synchronization tasks altogether; its usage is more recommended when there is no special filtering to be done before merging the packet logs. Monitoring can be performed by Kismet to analyze which stations are participating on the experiment, and detect packets coming from other stations during the experiment. Wismon as a monitor is convenient for experiments where both PHY and MAC layers and involved. Mognet is often used as a replacement of Wireshark for mobility-enabled experiments, whereas Airtraf decomposes the traffic on different layers. Airtraf can be used for monitoring purposes during flow-oriented experiments and/or experiments with cross-layer interactions. Wifiscanner targets security experiments, since it can detect certain misbehavior patterns from the STAs. For the processing stage, all the solutions use a custom script. As we have mentioned earlier, there is a small collection of custom scripts available at the CRAWDAD repository, which can help as a starting point to fulfill more specific functions.

One can notice the lack of flexibility for data processing in the list of tools presented above. Although packet log capture, merging and synchronization utilities are available, once the traces are merged, the result is a binary file with tcpdump format to be processed. This file, which can become huge after a few minutes of experimentation, can be processed with the pcap library to extract the packets, decode them into fields and extract the relevant parameters to plot the graphs and analyze the results. This operation represents a heavy and unrewarding task. Moreover, custom scripts used to process the results are generally rewritten from scratch for new experiments. The need for custom scripts shows that there is no standard tool to process straight the data and obtain standard graphs like the probability density function (pdf) from a specific parameter, or the interarrival time for a particular packet flow.

B. Common Pitfalls to Avoid

In this section, we present a non exhaustive list of very common pitfalls which can occur during experiments and some suggestions to avoid them when possible.

- **CPU overload at the sniffer:** In Section IV, we mentioned that when a station uses the promiscuous mode, the wireless card driver sends a copy of any received or transmitted packets to a part of the kernel called the packet filter. Then all the packets have to be copied from the kernel space to the user space where the sniffer is actually running. Interruptions can be handled late if the OS is overloaded. If a packet arrives late, the arrival timestamp added by the capture library looses accuracy[9]. The copy operation consumes a lot of CPU time and can overload the machine, causing the kernel to drop packets. If only part of packets have to be analyzed, we suggest to construct a specific filter expression that fits the measurement needs, and apply it to the packet filter. Portoles-Comeras et al. [50] have measured the capture limits of a wireless experimental platform, based on commercial off-the-shelf hardware. They have shown that sniffers can offer full rate capture when correctly calibrated, i.e. up to the level of saturation loss. Within their layout, they observed a limit of 2500 packets per second for Atheros sniffers and 2150 packets per second for Prism-based cards. At the kernel level, there is place for improvement on the packet capture efficiency, as Deri [51] shows in the case of wired networks for high speed capture. Nevertheless, current 802.11a/b/g packet capture rate has not reached yet to the bound where these improvements would be needed.
- **Traffic overload:** Caution is required in interpreting packet loss in presence of high data rate traffic, because packet loss can also occur due to buffer overflow. We suggest to use flexible wireless card drivers such as madwifi

[9]Note that the receiver's wireless device further adds a timestamp on the radiotap header to identify the arrival of the first bit of the packet.

TABLE VII
RECOMMENDED TOOLS FOR DIFFERENT USAGES

Type of Experiment	Recommended Tools			
	Sniffing	Merging	Processing	Monitoring
Throughput	Wireshark	Wit	Wireshark & Custom script	Wismon
Contention Window	Wireshark	Not needed	Custom script	Kismet
Airtime	Jigsaw	Jigsaw	Custom script	Airtraf
Spatial packet loss	Wireshark	Not needed	Custom script	Wismon
Analyzing capture effect	Jigsaw	Jigsaw	Custom Script	Wismon or Aitraf
Impact of Rx power on packet loss	Wireshark	Not needed	Wireshark filtering+ Custom Script	Wismon
Impact of Tx rate on packet loss	Wireshark	Not needed	Wireshark filtering+ Custom Script	Kismet or Airtraf
Impact of mobility	Mognet	Wit	Custom Script	Kismet
Flood attack detection	Wireshark	Not needed	Custom script	Wifiscanner

which can be easily instrumented to identify potential packet loss due to high contention on the medium.

- **Card/OS-dependent performance:** The sniffing process can produce different results according to the type of wireless card driver and to the operating system. As we mentioned in Section III-A1, when a sniffer captures a frame, it replaces the PLCP header by a DLT header whose format depends on the wireless card driver. But the packets captured may also differ from one operating system to the other. To cope with heterogeneous header formats, the pcap library is widely used on Linux platforms. Libpcap opens a special type of capture socket to retrieve all types of packets. However, such a functionality is not available on Windows platforms. Indeed, the corresponding Winpcap library drops all control packets received and so, many frames are not available at the upper layers. Furthermore, using Winpcap, packets sniffed are converted in fake ethernet packets, which causes the removal of some important information, like reception power level, noise, channel, modulation or MAC timestamp. For Windows platforms, we suggest to use a commercial capture library such as Airpcap [52] from CACE technologies company, which allows to retrieve all data, control and management 802.11 frames.
- **Positioning sniffers and merging traces:** As we mentioned in Section IV, a single probe may not be able to observe all the frames sent to or from a particular AP due to radio reception and range. It is therefore very important to use spatially dispersed sniffers and to synchronize the different traces at the microsecond granularity. Merging wireless traces is a critical operation and we suggest the use of verification mechanisms such as the one proposed by Schulman et al. [53], which evaluates the fidelity of merged and independent wireless network traces by estimating their completeness and clock accuracy.
- **Imprecise RSSI measurements:** The Receiver Signal Strength Indicator is known to be inaccurate in different platforms. For instance, measurements provided by the Atheros 5212 chipset do not allow for a fine-grained differentiation in the range relevant to bit-rate selection (especially at bit-rates below 36Mbps) [54]. Due to the fact that the IEEE 802.11 standard does not specify a required method of measuring RSSI, signal strength numbers from different vendors should not be compared to each other, since they are probably measuring it in different ways [12].
- **Identifying sources of interferences:** Without special electromagnetic isolation, like anechoic chambers [55], 802.11 devices suffer interference from a large number of products operating in the unlicensed 2.4 GHz band, see Section II-A. When stations are not mobile, most of the channel power variability is due to moving objects around. For example, a temporal signal fading occurs when an object obstructs the line of sight (LOS) between the stations and the AP. Also, as IEEE 802.11 channels are not orthogonal, possible interchannel interference in crowded spectrum and unplanned configurations can decrease drastically the performance of wireless protocols. Furthermore, during an experiment, a number of nearby machines or devices passing by can interfere with the experiment/measurement, even if the packets are sent on non-orthogonal channels. This is the case for mobile stations that try to associate to the nearest APs using active probing. Another source of interference is due to recent *Smart APs* implementing internal algorithms that periodically scan the overall channels to detect neighbor APs and to dynamically select the clearest channel to use. Therefore, it is important to characterize all potential sources of interferences during the wireless measurements phase. A spectrum analyzer could be used jointly with wireless probes to complete and refine the measurement process.

VII. CONCLUSION

In this paper, we proposed a taxonomy of IEEE 802.11 wireless parameters and relevant open source measurement tools that can be used for wireless experimentations and monitoring. We focus on tools available in the public domain

for the Linux environment and discuss their features and usage with several case studies. Then, we present some common pitfalls to avoid while performing wireless measurements.

There is still a wide choice of development branches for collaboration and development for wireless monitoring and experimentation. For instance, there is still no support of the wireless headers for 802.11e parameter extraction and interpretation, nor support for MIMO radio parameters for the upcoming IEEE 802.11n standard. Data management for wireless experimentation is currently at an early development stage. Not only the packet traces should be collected and stored for further analysis, but also the layout, experimental conditions and configuration corresponding to the experimentation. Such information is very important to perform rigorous performance analysis of wireless protocols.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable suggestions that help to improve the presentation of the paper. This work was partially supported by the EU FP7 IST OneLab2 grant No 224263.

REFERENCES

[1] "IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999), June 12 2007.

[2] L. Kleinrock and F. Tobagi, "Packet switching in radio channels part II: The hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution," IEEE Trans. Commun., Vol. COM-23, No. 12, pp. 1417-1433, 1975.

[3] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," Proc. of ACM MSWiM, October 2004.

[4] D. Kotz, C. Newport and C. Elliott, "The Mistaken Axioms of Wireless-network Research", Technical Report TR2003-467, Dept. of Computer Science, Darmouth College, July 2003, [Online]. Available: http://pdos.csail.mit.edu/decouto/papers/kotz03.pdf.

[5] Wireless Warrior [Online]. Available: http://www.wireless-warrior.org/software/sniffing/.

[6] A. Mahanti, C. Williamson and M. Arlitt, "Remote analysis of a distributed WLAN using passive wireless-side measurement", Perform. Eval. 64, 9-12, pp. 909-932, October 2007.

[7] A. Goldsmith, "Wireless Communications", Cambridge University Press, ISBN 0-521-83716-2, 2005.

[8] S. Khurana, A. Kahol, and A.P. Jayasumana, "Effect of hidden terminals on the performance of IEEE 802.11 MAC protocol", Proc. of 23rd Conference on Local Computer Networks (LCN), pp. 12-20, 1998.

[9] Libpcap [Online]. Available: http://www.tcpdump.org/.

[10] AVS Capture Frame Format, [Online]. Available: http://www.locustworld.com/tracker/getfile/prism2drivers/doc/capturefrm.txt.

[11] Radiotap [Online]. Available: http://www.radiotap.org.

[12] J. Barker, "You Believe You Understand What You Think I Said: The Truth About 802.11 Signal And Noise Metrics", Document D100201, 2004 - Connect802 Corporation, [Online]. Available: http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf.

[13] M. Raya, J.P. Hubaux and I. Aad, "DOMINO: a system to detect greedy behavior in IEEE 802.11 hotspots", Proceedings of the 2nd international conference on Mobile systems (MobiSys), Boston, MA, June 2004.

[14] F. Vacirca and A. Baiocchi, "Characterization of Service Times Burstiness of IEEE 802.11 DCF," Wired/Wireless Internet Communications, Springer, 2007, pp. 223-234.

[15] J. Bardwell, "Converting Signal Strength Percentage to dBm Values," November 2002, WildPackets Inc., [Online]. Available: http://www.wildpackets.com/elements/whitepapers/Converting_Signal_Strength.pdf.

[16] D. Dujovne and T. Turletti, "Multicast in 802.11 WLANs: an experimental study," Proc. of the 9th ACM international Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Torremolinos, Spain, October 2-6, 2006. pp. 130-138.

[17] J. Lacan and T. Perennou, "Evaluation of Error Control Mechanisms for 802.11b Multicast Transmissions," International Workshop on Wireless Network Measurement (WinMee), Boston, MA, USA, April 3, 2006.

[18] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla and M. Y. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique," Proc. of ACM SIGCOMM'04, Portland, OR, USA, September 2004.

[19] K. Lakshminarayanan, V. N. Padmanabhan, and J. Padhye, "Bandwidth Estimation in Broadband Access Networks," Proc. of ACM IMC'04, Taormina, Sicily, Italy, October 2004.

[20] A. Johnsson, B. Melander, and M. Bjorkman, "Bandwidth Measurement in Wireless Networks," Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Porquerolles, France, June 2005.

[21] Madwifi project [Online]. Available: http://madwifi.org/.

[22] Gnuradio [Online]. Available: http://www.gnu.org/software/gnuradio/.

[23] Warp [Online]. Available: http://warp.rice.edu.

[24] J. Yeo, S. Banerjee and A. Agrawala, "Measuring traffic on the wireless medium: Experience and pitfalls", Technical report, CS-TR 4421, University of Maryland, College Park, December 2002. [Online]. Available: http://www.cs.umd.edu/Library/TRs/CS-TR-4421/CS-TR-4421.pdf.

[25] J. Yeo, M. Youssef and A. Agrawala, "Characterizing the IEEE 802.11 Traffic: The Wireless Side," CS-TR-4570, March 2004. [Online]. Available: http://www.cs.umd.edu/Library/TRs/CS-TR-4570/CS-TR-4570.pdf.

[26] Y.C Cheng J. Bellardo, P. Benko, A.C. Snoeren, G.M. Voelker and S. Savage, "Jigsaw: Solving the Puzzle of Entreprise 802.11 Analysis," in Proc. of ACM SIGCOMM, Pisa, Italy, September 11-15 2006.

[27] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analysing the MAC-level Behavior of Wireless Networks in the Wild," Proc. of ACM SIGCOMM, Pisa, Italy, September 11-15 2006.

[28] Athdebug and 80211debug documentation, Madwifi v0.9.4, [Online]. Available: http://madwifi.org/wiki/DevDocs/AthDebug.

[29] athstats.c, Madwifi v0.9.4 source code, [Online]. Available: http://www.madwifi.org.

[30] R. Presuhn et al., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)," IETF RFC 3416, December 2002.

[31] T. Henderson and D. Kotz, "Problems with the Dartmouth wireless SNMP data collection," Dartmouth Computer Science Technical Report TR2003-480, December 2003.

[32] Wireless Tools for Linux [Online]. Available: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.

[33] Wavemon tool [Online]. Available: http://www.janmorgenstern.de/projects-software.html.

[34] Wrapi [Online]. Available: http://sysnet.ucsd.edu/pawn/wrapi/.

[35] Airtraf tool [Online]. Available: http://airtraf.sourceforge.net.

[36] C. Hoene, B. Rathke and A. Wolisz: "EasySnuffle: A tool to measure the performance of multimedia flows over IEEE 802.11b", Technical University of Berlin TKN - Berlin - Germany, March 10, 2002, [Online]. Available: http://www.tkn.tu-berlin.de/research/easysnuffle/EasySnuffle.pdf.

[37] Easysnuffle tool [Online]. Available: http://www.tkn.tu-berlin.de/research/easysnuffle/.

[38] Jigsaw tool [Online]. Available: http://sysnet.ucsd.edu/wireless/.

[39] Kismet tool [Online]. Available: http://www.kismetwireless.net.

[40] Mognet tool [Online]. Available: http://www.monolith81.de/mognet.html.

[41] Wifiscanner tool [Online]. Available: http://wifiscanner.sourceforge.net.

[42] Wireshark tool [Online]. Available: http://www.wireshark.org.

[43] WisMon tool [Online]. Available: http://planete.inria.fr/software/WisMon/.

[44] Wit tool [Online]. Available: http://www.cs.washington.edu/research/networking/wireless/index.html.

[45] Crawdad repository [Online]. Available: http://crawdad.cs.dartmouth.edu/tools.php.

[46] G. Berger-Sabbatel, Y. Grunenberger, M. Heusse, F. Rousseau and A. Duda, "Interarrival Histograms: A Method for Measuring Transmission Delays in 802.11 WLANs," Research Report, LIG - Grenoble Informatics Laboratory, Grenoble, 2007, [Online]. Available: http://drakkar.imag.fr/spip.php?article242.

[47] K. Papagiannaki, M. Yarvis, and W. S. Conner, "Experimental Characterization of Home Wireless Networks and Design Implications," Proc. IEEE INFOCOM, Barcelona, Spain, April, 2006.

[48] R.G. Garroppo, S. Giordano, S. Lucetti, and L. Tavanti, "Providing air-time usage fairness in IEEE 802.11 networks with the deficit transmission time (DTT) scheduler," Wireless Network, 13, 4 August 2007, 481-495.

[49] J. Lee, W. Kim, S. Lee, D. Jo, J. Ryu, T. Kwon and Y. Choi, "An experimental study on the capture effect in 802.11a networks," Proc. of the Second ACM international Workshop on Wireless Network Testbeds,

Experimental Evaluation and Characterization (WinTECH), Montreal, Quebec, Canada, September 10, 2007.
[50] M. Portoles-Comeras, M. Requena-Esteso, J. Mangues-Bafalluy, M. Cardenete-Suriol, "Monitoring wireless networks: performance assessment of sniffer architectures," Proc. of IEEE ICC, Istambul, Turkey, pp.646-651, June 2006.
[51] L. Deri, S.P.A. Netikos, K. Via Del Brennero and L.L. Figuretta, "Improving passive packet capture:beyond device polling", Proc. of SANE, Amsterdam, The Netherlands, September 2004.
[52] Airpcap product, CACE technologies [Online]. Available: http://www.cacetech.com/products/airpcap_family.htm.
[53] A. Schulman, D. Levin and N. Spring, "On the Fidelity of 802.11 Packet Traces", 9th Passive and Active Measurement conference, Cleveland, Ohio, April 2008.
[54] K. Ramachandran, H. Kremo, M. Gruteser, P. Spasojevic and I. Seskar, "Scalability Analysis of Rate Adaptation Techniques in Congested IEEE 802.11 Networks: An ORBIT Testbed Comparative Study", Proc. of WoWMoM, Helsinki, Finland, June 2007.
[55] G. Rahmatollahi, S. Galler, J. Schroeder, K. Jobmann and K. Kyamakya, "Propagation Delay Based Positioning Using IEEE 802.11b Signals", Proc. of 3rd Workshop on Positioning, Navigation and Communication (WPNC), Hannover, Germany, March 2006.

Diego Dujovne obtained his Electronic Engineer 6-year degree from National University of Cordoba, Argentina in 1999. Between 1999 and 2001 he worked as lecturer at the Informatics and Electronics Departments and from 2002 to 2004 he worked as a full-time Adjoint-Professor as the director of the Digital Signal Processing Lab at UNC, Argentina. He obtained his PhD at the Planète project-team at INRIA Sophia Antipolis, France in 2009, and he is currently a full-time researcher and lecturer at Universidad Diego Portales, Chile. His research interests include MAC layer development for wireless networks, WLAN multicast improvements and Wireless experimental measurements. Additionally, he is member of IEEE since 1994.

Thierry Turletti received the M.S. (1990) and the Ph.D. (1995) degrees in computer science both from the University of Nice - Sophia Antipolis, France. He has done his PhD studies in the RODEO group at INRIA Sophia Antipolis. During the year 1995-96, he was a postdoctoral fellow in the Telemedia, Networks and Systems group at LCS, MIT. He is currently a senior research scientist at the Planète group at INRIA Sophia Antipolis. His research interests include multimedia applications, congestion control and wireless networking. Dr. Turletti serves on the Editorial Board of the Wireless Communications and Mobile Computing (WCMC), Wireless Networks (WINET) and Advance on Multimedia (AM) journals.

Fethi Filali received his Computer Science Engineering and DEA degrees from the National College of Informatics (ENSI) in 1998 and 1999, respectively. At the end of 1999, he joined the Planète research team at INRIA (National research institute in informatics and control) in Sophia-Antipolis to prepare a Ph.D. in Computer Science which he has defended on November 2002. During 2003, he was an ATER (Attaché Temporaire d'Enseignement et de Recherche) at the Université of Nice Sophia-Antipolis (UNSA) and he joined on September 2003 the Mobile Communications department of Institut Eurécom in Sophia-Antipolis as an Assistant Professor. He is/was involved in several French-funded (Dipcast, Constellation, Rhodos, Cosinus, Airnet, WiNEM) and IST FP6/7 (Widens, Newcom, Daidalos, E2R, Multinet, Unite, Chorist, iTetris, Newcom++) projects. In the context of some of these projects, he designed and developed an open, flexible and efficient architecture for the support of heterogeneous radio technologies. This architecture was integrated in EURECOM's wireless software-radio platform. His current research interests include WIMAX (802.16)-related communication mechanisms, QoS support in IEEE 802.11-based networks, sensor and actuator networks (SANETs), vehicle adhoc networks (VANETs), routing and TCP performance in wireless networks. He served as a technical reviewer of several international conferences and journals. Additionally, he is a member of IEEE and IEEE Communications Society. In April 2008, he was awarded the «Habilitation à Diriger des Recherches» (HDR) from the University of Nice Sophia-Antipolis for his research on wireless networking.