Interarrival Histograms: A Method for Measuring Transmission Delays in 802.11 WLANs

Gilles Berger-Sabbatel, Yan Grunenberger, Martin Heusse, Franck Rousseau, Andrzej Duda
LIG - Grenoble Informatics Laboratory[*]
Grenoble, France
{gberger,ygrunenb,heusse,rousseau,duda}@imag.fr

ABSTRACT

Traditional measurement tools are not suitable for fine-grain performance evaluation of 802.11 wireless networks. To improve this situation, we have developed *Interarrival Histograms*, a method for measuring transmission delays based on gathering statistics of interarrival intervals at a receiving station. By measuring a sufficient number of interarrival interval samples with a clock resolution of 1 μs, the precision of estimating the mean transmission delay is well under one microsecond. Unlike other proposals, it does not require expensive custom-made cards, because it uses commercially available equipment. It allows flexible measurements not only of wireless cards, but also of access points, which cannot be done with other proposed methods, unless one can instrument the access point kernel. In this paper, we describe the principles of the proposed method, derive the estimation error of measurements, and experimentally demonstrate its precision by applying it to the 100 Mb/s Ethernet. To illustrate the cases in which the method can help analyzing the complex behavior of 802.11 networks, we present measurements of several commercially available access points and wireless cards to show some anomalies and differences with the standard.

1. INTRODUCTION

Measuring transmission delays in IEEE 802.11 wireless networks [12] and experimentally evaluating their performance is fairly difficult. Their MAC layer relies on a set of parameters such as interframe intervals (DIFS, SIFS, AIFS), backoff slots, and the size of the contention window. Some of these parameters are constant and others are randomly chosen, but they are defined with the precision of a microsecond. Obviously, we can easily measure high level performance metrics such as throughput or fairness, but investigating how they depend on lower layers parameters is much more difficult. For example, the developers of wireless drivers for open source operating systems such as Linux or *BSD must often cope with incomplete or reverse engineered specifications so they cannot rely upon expensive tools to verify the low-level behavior of a given wireless card. An implementation of a driver can present performance problems, but the developer does not have any clue about the cause of problems. Implementing a MAC layer or modifying an existing one on a wireless card may also require sophisticated measurement tools to validate the behavior of the implementation. In all these cases, we need a method for measuring transmission delays with sufficient precision. Such a method should be simple and usable in a set up with standard hardware without requiring in depth modifications of the kernel.

During our past work on various aspects of 802.11 performance [1,8,10,11], we have experimentally observed different metrics such as throughput, round trip delays, or fairness, but it was always difficult to gain more insight into the behavior of different implementations of 802.11. When we wanted to implement our own access method [7], we wondered how to measure its fine-grain performance and no existing method was suitable for this goal. To improve this situation, we have developed *Interarrival Histograms*, a method based on measuring statistics of packet interarrival intervals at a receiving station. The method is simple, yet powerful and does not require instrumentation of the measured wireless station, all measurements being done at the destination host or at a monitor station. A source station sends a flow of packets to a wired host connected to a 802.11 access point via a direct 100 Mb/s Ethernet link and the host timestamps packets upon arrival with the clock resolution of 1 μs. Interarrival intervals of packets are then computed, statistically processed,

[*]LIG is a joint research laboratory of CNRS (*Centre National de la Recherche Scientifique*), INRIA (*Institut National de Recherche en Informatique et Automatique*), INP Grenoble (*Institut Polytechnique de Grenoble*), UJF (*Université Joseph Fourier*), and UPMF (*Université Pierre-Mendès-France*).

and finally presented in the form of a histogram to visually grasp the most frequent values of the transmission time and easily deduce some low-level characteristics of measured wireless cards. Statistical processing of the measured intervals gives an error estimation that is even lower than the clock resolution (1 μs). This fact may be surprising—obtaining such a good precision with off-the-shelf hardware is usually unbelievable, but we provide a theoretical derivation of the error estimation and experimental validation of the precision.

Our method works in two modes: it is either a wired host connected to an access point or a wireless monitor station that measures interarrival intervals. This last mode provides slightly better precision of measurements, but usually we use both to compare histograms.
The paper makes several important contributions:

- We define an original method for measuring transmission delays in 802.11 wireless networks with the precision lower than 1 μs. Histograms are generated with the resolution of 1 μs. Unlike other proposals, the method does not require expensive custom-made cards, because it uses commercially available equipment. It allows flexible measurements not only of wireless cards, but also of access points, which cannot be done with other proposed methods, unless one can instrument the access point kernel.
- We derive the estimation error of measurements and show that because of a particular dependency of random variables under consideration, the estimation error decreases fairly quickly with the number of samples.
- By measuring the 100 Mb/s Ethernet, we experimentally demonstrate the surprising precision of the proposed method.
- To show its usefulness in the context of 802.11 wireless networks, we measure several commercially available access points to see how they conform to the 802.11 specifications and analyze the behavior of several wireless cards.

Note that the method concerns transmission delays over a 802.11 link so it focuses on MAC layer characteristics and can only implicitly capture some physical layer effects when they influence the transmission delay (collisions, retransmissions due to channel errors, influence of channel contention etc.). It can measure transmission delays under adverse channel conditions, but it is not intended to measure strict physical characteristics such as signal strength, interference, or the influence of hidden nodes.

The paper is organized as follows. Section 2 overviews the related work. Section 3 describes the principles of the proposed method, derive the estimation error, and experimentally validate its precision. In Section 4, we report on measurements of several wireless cards and evaluate the influence of some system parameters such as processor speed and interrupt frequency. In Section 5, we illustrate how the method can be used by measuring several commercially available access points and wireless cards. Section 6 concludes the paper.

2. RELATED WORK

An overwhelming part of research work on wireless networks validates new concepts or protocols with simulation. However, even widely used simulators such as NS2, OPNET, or GloMoSim do not provide sufficiently realistic models of complex wireless behavior and their results may significantly differ for the same problem [4]. Experimental evaluation of wireless ad hoc networks also shows that much of the theoretical work on wireless protocols is based on wrong assumptions [14]. But precise measurement of wireless cards, access points, or experimental prototypes is not that easy and usually researchers confine themselves to simple measurements of the throughput and the round trip time.

In our previous work, we have tried to find new aspects of 802.11 performance by experimenting with available cards, for instance, the discovery of the *Performance Anomaly* [8], in which a station transmitting with a low bit rate limits the performance of faster stations, came out from a measurement session. Based on many such experiments, we have begun to look for a method for precise measurement of transmission times and finally developed the proposed method of *Interarrival Histograms.*

There are several similar recent attempts. Bianchi *et al.* have developed a hardware platform based on custom-made FPGA wireless cards for precise measurements of 802.11 WLANs [2, 16]. Their cards can record time with the resolution of 50 ns, generate any kind of frames even invalid ones, and operate in a way that violates the standard specifications. They thus obtain interesting results by testing how commercially available cards perform and react to a non-standard behavior, for instance they inject invalid frames and observe whether a tested card waits for the EIFS period. This approach to measurements is limited to testing wireless cards—measuring access points is not possible. Our method achieves a similar measurement precision without the need for difficult to develop custom-made FPGA cards. However, in our approach, we are not able to inject non-standard frames, because we use commercially available cards.

Dangerfield *et al.* used a measurement set up at the sender side for analyzing the behavior of an EDCA implementation [5]. The precision of measurements was around 10 μs thus allowing the authors to measure the throughput and the mean delay with sufficient accuracy for evaluating the impact of EDCA parameters on performance. They further improved their method by taking advantage of timestamping on wireless cards to achieve the precision of 1 μs [15] (we also use this feature of some wireless cards). This work presents the results of measurements as histograms in a form similar to ours. The main difference with our method is that they measure the transmission time on the sender station, while we do so at a receiving wired host or at a monitor station. In this way, we are able to measure the performance of commercial access points, while their set up can only measure a wireless card plugged into a instrumented PC running Linux.

Finally, we want to mention other standard measurement approaches based on clock synchronization. If two synchronized stations can measure time with high precision, obtaining statistics on transmission times is straightforward [6]. Time synchronization can be achieved by using stations with GPS devices, but such an instrumentation may be expensive and is only justified for long distances. In indoor situations, one can connect wireless stations in a closed neighborhood by an additional cable, but it requires instrumentation of their hardware to synchronize clocks.

To summarize, we do not claim that using histograms is new, because several papers already visualized measurement results in this way, nor reporting on backoff characteristics of commercially available 802.11 wireless cards. We think that the principles of our method are original and it presents the advantage of only requiring cheap off-the-shelf hardware to measure with high precision what cannot be measured by other methods (e.g. commercial access points).

3. PRINCIPLES OF THE METHOD

Our goal is thus to define a method for measuring transmission delays that only requires recording timestamps in the kernel or on a wireless card with a microsecond resolution on a receiving station.

3.1 Principles

To understand the method, let us start with the description of the measurement set up. Figure 1 presents the details of the configuration: a wireless station communicates with a wired host through an 802.11 access point in the infrastructure mode and

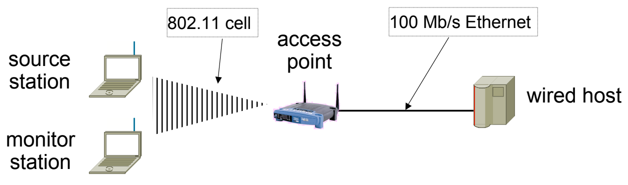

Figure 1: Measurement configuration.

a direct 100 Mb/s Ethernet link. A home-made application tool composed of a client and a server similar to `iperf` generates a continuous stream of UDP packets of a given rate and size to the destination: such a traffic source keeps the buffers on the network interfaces full so that it sends them as soon as the MAC layer allows to do it. Traffic can be sent either from the wireless stations to the wired host, if we intend to test the wireless card, or in the opposite direction if we want to test the access point. Traffic is captured and analyzed either at the monitor station or at the destination station. The former can give more precise results at the price of a slightly more complex setup. We have developed a program based on the `pcap` library that captures packets and records the interarrival intervals of packets. It controls the instant of storing measurements on disk to avoid I/O operations during the capture.

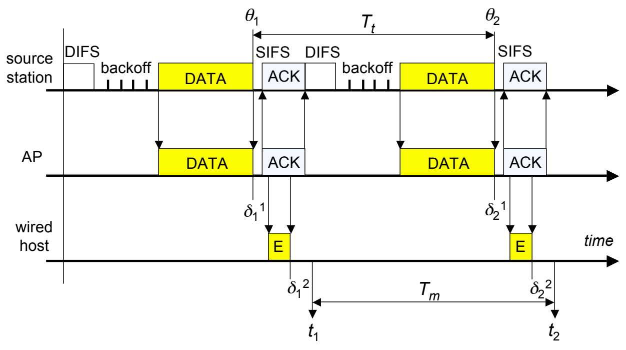

Figure 2: Principle of measurements at the wired host.

Figure 2 illustrates the principle of measurements based on *Interarrival Histograms*. We want to measure T_t, the overall transmission time of a data frame that includes all the overhead: interframe intervals (DIFS, SIFS, AIFS), the backoff, and the transmission times of the data and ACK frames. Note that usually the transmission time to estimate starts with the DIFS interval and ends with the reception of an ACK. However, as our method is based on the intervals measured at the receiving host, T_t to estimate in our method corresponds to the interval presented in Figure 2. If we gather a large number of samples, the both definitions of the transmission time are equivalent.

Figure 2 presents the sequence of events when we measure the arrival instants at the wired host. The source wireless station generates a packet to the wired host and sends it to the access point in a data frame after the DIFS delay and the backoff interval chosen from the contention window CW. The frame is received by the access point and retransmitted on the Ethernet interface after some delay δ^1. The access point confirms the reception of the data frame with an ACK sent to the source station. The Ethernet frame (denoted by E in the figure) arrives at the wired host and after some delay δ^2 the network driver timestamps the packet with instant t_1. The same sequence of events repeats for the next packet timestamped with instant t_2 upon arrival. As we measure the transmission time on short distances (several meters), we neglect the propagation delay.

In our measurement set up, we have carefully considered the problem of precise timestamping of packets. Upon reception of a frame, the network interface generates an interrupt. The network driver gets the frame, record the instant of its reception with the resolution of 1 μs, and passes it to the `pcap` library. Timestamping is done in the kernel space so that it is not biased by random delays that might occur when timestamping is performed in the user space: the only delay involved in timestamping is the interrupt processing time, which has small jitter on a lightly loaded machine (a few microseconds). However, jitter may increase if the card generates too many interrupts, which may happen for small packets. In some configurations, it is also possible to get a timestamp set by the hardware of the network interface as soon as the frame is received (i.e. before any software processing), which can even lower the variability of delay δ^2.

To see how we are able to estimate T_t, the overall transmission time, consider the following relations:

$$t_1 = \theta_1 + \delta_1^1 + T_E + \delta_1^2, \qquad (1)$$

and

$$t_2 = \theta_2 + \delta_2^1 + T_E + \delta_2^2, \qquad (2)$$

where T_E is the transmission time of the data frame on the Ethernet. We assume that compared to other sources of delay variability such as δ^1 and δ^2, the transmission time on the Ethernet link can be considered as constant. We measure T_m, the difference between the consecutive timestamps corresponding to the reception of packets:

$$T_m = t_2 - t_1 = \theta_2 - \theta_1 + \delta_2^1 - \delta_1^1 + \delta_2^2 - \delta_1^2, \qquad (3)$$

which yields

$$T_m = T_t + \delta_2^1 - \delta_1^1 + \delta_2^2 - \delta_1^2, \qquad (4)$$

We can observe that

$$E(T_m) = E(T_t), \qquad (5)$$

because the differences in δ^1 and δ^2 cancel out. Moreover, the variance of the measured interval is

$$Var(T_m) = Var(T_t) + Var(\delta^1) + Var(\delta^2), \qquad (6)$$

where $Var(\delta^1)$ and $Var(\delta^2)$ are variances of the retransmission delay at the access point and of the time needed to intercept a packet and assign a timestamp, respectively. We can see that by only timestamping the arrival of packets at the receiving station, we can estimate the variable transmission time on the 802.11 link even if several delays add to the total transmission time.

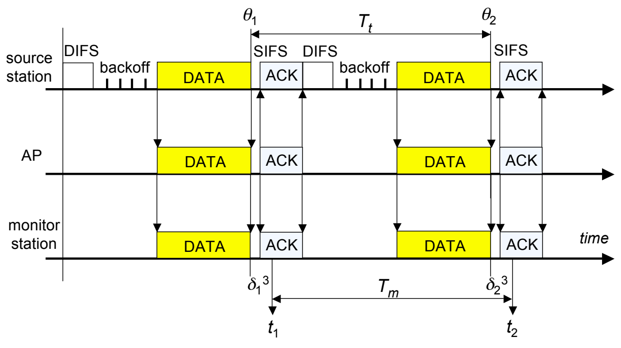

Figure 3: Principle of measurements at the monitor station.

Figure 3 presents the sequence of events when packets are captured by a monitor station. As previously, the source wireless station sends a data frame to the access point, which is overheard by the monitor station set into the promiscuous mode. The frame is received by the access point and confirmed with an ACK frame. The access point retransmits the packet on the Ethernet to the wired host (not shown in this diagram). The monitor station timestamps the frame with instant t_1 after some delay δ^3. The same sequence of events repeats for the next packet timestamped with instant t_2. In a similar way, we can derive the relation:

$$T_m = T_t + \delta_2^3 - \delta_1^3. \qquad (7)$$

So, the mean is

$$E(T_m) = E(T_t), \qquad (8)$$

as well as the variance

$$Var(T_m) = Var(T_t) + Var(\delta^3), \qquad (9)$$

where $Var(\delta^3)$ corresponds to the variance of the time needed to intercept a packet and assign a timestamp on the monitor station. Our experiments show that this method is more precise than recording timestamps at the wired host, however only some types of wireless cards provides sufficient characteristics for obtaining good precision.

3.2 Precision

Let us evaluate the precision of the method for measuring T_t in the last set up (measurements at the monitor station). We will focus on estimating the inherent error of the measurement method, so we assume in this section that T_t is constant. Denote by

$$x_i = T_t + \delta_{i+1} - \delta_i, \quad i = 1, \ldots, n \qquad (10)$$

sample i of the measured transmission time T_m [1] in a sequence of n measurements. Let us assume that

$$\delta_i = \delta_{min} + d, \qquad (11)$$

where δ_{min} is a constant and d is normally distributed with mean μ and standard deviation σ: $\mathcal{N}(\mu, \sigma)$. We assume that σ is small compared with δ_{min} so that δ_i is a good approximation of a positive random variable. We consider the estimator of the partial mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = T_t + \frac{1}{n}\sum_{i=1}^{n}(\delta_{i+1} - \delta_i), \qquad (12)$$

to obtain

$$\bar{x} = T_t + \frac{\delta_{n+1} - \delta_1}{n}, \qquad (13)$$

because all intermediate terms cancel out. δ_{n+1} and δ_1 are independent normal random variables, so the distribution of $\delta_{n+1} - \delta_1$ is also normal with parameters $\mathcal{N}(0, \sqrt{2}\sigma)$. Thus, the estimator of the mean also follows the normal distribution:

$$\bar{x} \sim \mathcal{N}(T_t, \frac{\sqrt{2}\sigma}{n}). \qquad (14)$$

We can find the confidence interval for the partial mean by considering random variable

$$\frac{\bar{x} - T_t}{\sqrt{2}\sigma} n, \qquad (15)$$

which is normal with parameters $\mathcal{N}(0, 1)$. So we obtain

$$P(\bar{x} - \epsilon \le T_t \le \bar{x} + \epsilon) = 1 - \alpha, \qquad (16)$$

where

$$\epsilon = \frac{\sqrt{2}\sigma}{n} K_{\alpha/2}. \qquad (17)$$

[1]cf. Eq. 7 in which we drop the mark 3 in variable δ for simplicity.

For $\alpha = 0.003$ (99.7% confidence level), constant $K_{\alpha/2} = 3$ and we obtain the following error estimation of the partial mean:

$$\epsilon = \frac{3\sqrt{2}\sigma}{n}. \qquad (18)$$

Even if we cannot take advantage of this error estimation for our method, because in general we do not know σ, it is interesting to see that the error quickly decreases in function of n—the usual increase of precision with an increasing sample size (function of $1/\sqrt{n}$) becomes here $1/n$, because samples x_i are not independent.

As x_i are not independent random variables, we cannot directly apply standard methods for finding the confidence interval for all measurements. Rather, we will construct a series of partial mean estimations for N sequences of measurements that we consider independent:

$$\bar{x}_j, \quad j = 1, \ldots, N. \qquad (19)$$

Let us define the estimators of the mean and the variance

$$\bar{X} = \frac{1}{N}\sum_{j=1}^{N}\bar{x}_j, \quad \bar{S}^2 = \frac{1}{N}\sum_{j=1}^{N}(\bar{x}_j - \bar{X})^2. \qquad (20)$$

Random variable

$$\frac{\bar{X} - T_t}{\bar{S}}\sqrt{N-1} \qquad (21)$$

has the Student t distribution with $N-1$ degrees of freedom and we can derive the confidence interval as

$$P(\bar{X} - \epsilon \le T_t \le \bar{X} + \epsilon) = 1 - \alpha, \qquad (22)$$

where

$$\epsilon = \frac{\bar{S}}{\sqrt{N-1}} t_{\alpha/2, N-1}. \qquad (23)$$

For large N and $\alpha = 0.003$ (99.7% confidence level), $t_{\alpha/2,N-1} = 3$ and we obtain the following error estimation of our method:

$$\epsilon = 3\frac{\bar{S}}{\sqrt{N-1}}. \qquad (24)$$

This last estimation means that the precision of our method can be made arbitrarily good by increasing the number of sequences N, if the estimation of the standard deviation $\bar{S}$ is finite.

The set up with measurements at the wired host can also benefit from similar precision, however the error estimation of the partial mean will now include the standard deviations of delays δ^1 and δ^2. Then, the estimators defined in Eq. 20 applied to the series of N partial means give us the desired error estimation according to Eq. 24.

Table 1: Transmission time statistics for the Ethernet in function of the frame size.

Frame size [bytes]	200	400	600	800	1000	1200	1400
Computed transmission time [μs] (Eq. 25)	21.280	37.280	53.280	69.280	85.280	101.280	117.280
Histogram peak position [μs]	21	37	53	69	85	101	117
Mean measured transmission time ($\bar{X}$) [μs]	21.280	37.280	53.280	69.280	85.280	101.280	117.280
Difference average/computed [μs]	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Estimated standard deviation ($\bar{S}$)	0.007	0.007	0.013	0.011	0.013	0.012	0.012
Error estimation (ϵ, Eq. 24)	0.001	0.001	0.001	0.001	0.001	0.001	0.001

3.3 Validation

To validate our method and evaluate its precision, we have applied it to the case in which the transmission time is well known and almost constant: the 100 Mb/s Ethernet. In the experiment, the Ethernet cable directly connects two stations (no switch in between), so that the MAC delay is almost constant and the only variable part comes from random delays introduced by capturing and timestamping a packet (δ^2). This corresponds to the case for which we have derived the error estimation in the previous section.

For a packet of l bytes, the transmission time in the experiment should be

$$T_t = (l + 42) \times 0.08 + 1.92 \ \mu s, \qquad (25)$$

where 42 is the size of the UDP/IP/Ethernet headers in bytes, 0.08 is the transmission time of one byte, and 1.92 is the constant overhead of the Ethernet MAC layer (preamble, CRC, and interframe interval).

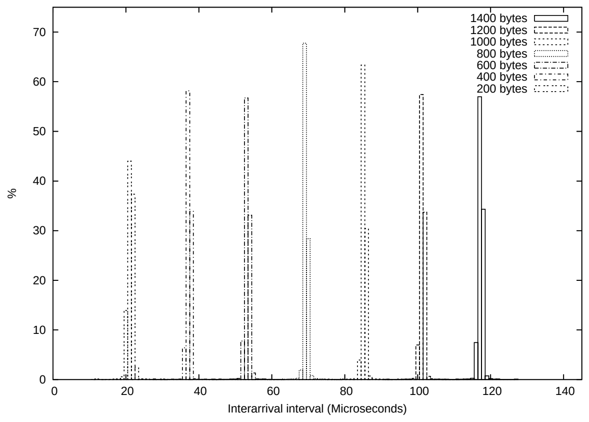

Figure 4: Histograms of transmission times for the 100 Mb/s Ethernet.

Figure 4 shows the histograms of the transmission times for packet sizes of 200, 400, ..., and 1400 bytes. We generate the histograms with the resolution of 1 μs. Each histogram corresponds to the observation of around 100,000 packets of a given size with $N = 1200$ sequences of n samples with n varying between 50 to 100. We can observe seven sharp peaks of 3 μs width.

Table 1 presents the detailed results with the estimation of the error:

- Computed transmission time (Eq. 25),
- Histogram peak instants,
- Mean measured transmission time ($\bar{X}$),
- Difference between the computed and measured interval,
- Estimated standard deviation ($\bar{S}$),
- Error estimation (ϵ, Eq. 24).

We can observe that the estimated mean of the measured transmission times is equal to the computed transmission time up to the rounding error (all computations were done with 3 significant figures after decimal point). Moreover, the error estimation based on Eq. 24 is equal to $1ns$, which is very small, because we used fairly large number of sequences N. The positions of peaks correspond to the computed and measured values up to the clock resolution of $1\mu s$. These measurements confirm our analysis of the precision in Section 3.2. The good precision of the method comes from the fact that we do not measure a single interval, but a series of intervals. Thus, the mean can be estimated with a very good precision, even better than clock resolution, because of the quick decrease of the estimation error in function of n. The positions of histogram peaks can also be used as estimations of the transmission time, but with the precision of clock resolution.

Having validated the method for almost constant transmission times of the Ethernet, we want to apply it to transmission over 802.11 wireless links. The transmission times in this case follow a complex distribution with several peaks basically corresponding to the values of random backoff. Instead of computing the estimations of the mean and the variance, we use histograms that allow us to visually grasp the most frequent values of the transmission time and infer some characteristics of tested wireless cards. Still the precision of histograms is around $1\mu s$, which is sufficient for our purposes. Thus, in the rest of the paper we present the measurement results of 802.11 wireless networks in form of histograms.

4. TUNING OF THE METHOD

We have experimented with our method to analyze the influence of some configurations or system parameters on its precision. In particular, we have tested wireless cards for the use on the monitor station and evaluated the influence of processors with different speed and various frequencies of kernel interrupts on the method.

We have used the measurement configuration as presented above (cf. Figure 1). If not stated otherwise, we use the Netgear-Prism GT card at the source station, because it follows fairly closely the specification of the 802.11 standard. The wired host is a workstation with an Athlon XP 2400+ processor running Linux FedoraCore 6 (kernel 2.6.18). Wireless stations are laptops with a Pentium Core 2 Duo, running Linux FedoraCore 6.

All measurements are done in a residential area without any other 802.11 network in favorable transmission conditions—wireless stations are close to the access point (2 m). This does not mean that the method cannot be used for analyzing performance behavior of 802.11 networks in adverse channel conditions, we simply limit the reported results in this paper to good channel conditions. Each histogram results from a session of 50,000 to 100,000 captured packets. The size of the UDP payload is 1472 bytes to fit the maximum Ethernet frame size.

4.1 Wireless cards as a monitor

We have first compared the results obtained from the capture at the wired host with measurements done at the monitor station with timestamping done in the kernel.

Figure 5 present the histograms of T_m for the D-Link-Atheros card[2] measured by a monitor station and the wired host. It contains 16 regularly spaced similarly shaped peaks. The distance between the first and the last peak is 135μs corresponding exactly to 15 slots of 9μs (802.11g). The first peak appears at 330μs, which is 4μs more than the expected value (326.04μs, cf. the grid in the background corresponding to the values defined in the 802.11g standard).

We can see that both histograms show the same values of peaks, however the histogram gathered at the wired host is a little bit wider, which comes from a slightly greater variability of delays: δ^1 and δ^2 with respect to δ^3.

[2]We identify wireless cards by giving its manufacturer (D-Link) and the name of its chipset (Atheros).

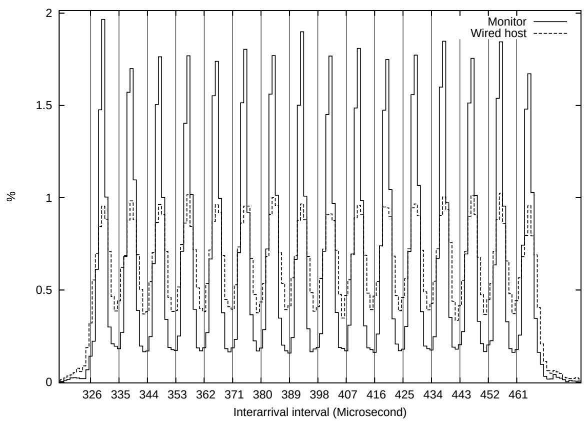

Figure 5: Histogram for the D-Link-Atheros card.

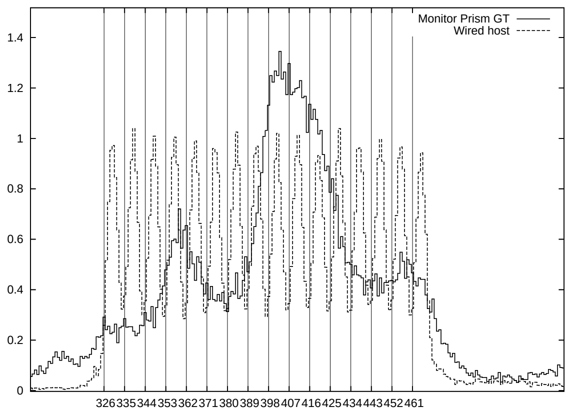

Figure 6: Histogram for Netgear-Prism GT

Figure 6 shows the histograms measured by the Netgear-Prism GT card at the monitor and at the wired host. We can see that they do not correspond, which means that we cannot use this card at the monitor, because the processing time of frames is too variable, if timestamping is done in the kernel.

Figure 7 presents a similar histogram for the Intel Centrino-IPW2915 card. The 12 last peaks are regular and corresponds well to the capture at the wired host, however the first 4 peaks are fairly distorted. From these tests, we can conclude that only the D-Link-Atheros card offers sufficient characteristics for the use at the monitor station, if timestamping is done in the kernel.

4.2 Influence of the processor

Figure 8 presents histograms when three laptops with different processors act as monitors. They use

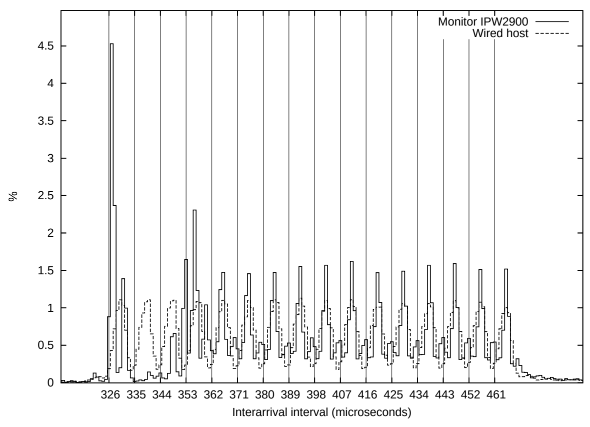

Figure 7: Histogram for Intel Centrino-IPW2915.

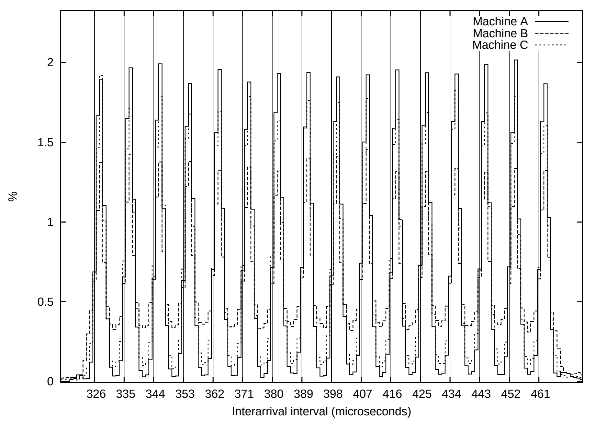

Figure 8: Influence of the processor.

the D-Link-Atheros wireless card, the same driver `madwifi-ng`, and run the same operating system (Linux Fedora Core 5):

- A: a Dell laptop with Pentium M at 1.7 GHz,
- B: a HP Omnibook Xe4500 laptop with Pentium IV M at 1.7 GHz,
- C: a HP Omnibook Xe3 laptop with Pentium III M at 800 MHz.

The slightly wider peaks of the B laptop probably come from the difference between the architecture of Pentium III and Pentium IV. We can see that the histograms correspond fairly well, so that the processor speed does not have much influence on capturing and timestamping packets.

4.3 Frequency of kernel interrupts

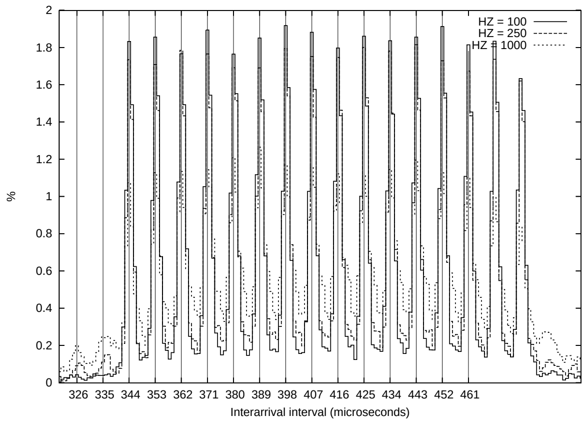

Figure 9: Histograms in function of the interrupt frequency HZ.

Table 2: Influence of HZ.

HZ	1000	250	100
% Histogram	92.2	96.50	96.50
1st peak	344.4	344.4	344.3
difference	1.62	1.25	1.17

We have tested our method for different values of the kernel variable HZ that defines the frequency of timer interrupts of 1/HZ s. Figure 9 shows the histograms for three values of HZ. We have obtained the best results for HZ= 100 as show the details in Table 2.

In each case, there is still a small proportion of intervals which fall outside of the histogram peaks. The Linux kernel does not provide support for real-time and for various reasons, the interrupts generated by the reception of a packet may sometimes be processed with a delay that can exceed one slot, as can be seen by examining the shape of the first and the last peak.

4.4 Timestamping on the wireless card

Two wireless cards (Netgear-Prism GT and D-Link-Atheros) offer the possibility of obtaining the timestamp associated by the hardware with the reception of a frame. Figure 10 shows the histogram based on this functionality obtained when both monitor and test stations use the Netgear-Prism GT card. It is fairly precise with a difference of 0.5 μs with respect to the middle of a peak. We can see 16 small peaks beginning at 362 μs that correspond to some packets sent at the bit rate of 48 Mbit/s. Capturing this behavior was not possible when using timestamping in the kernel.

Figure 11 presents the histogram obtained when the monitor and source stations use the D-Link-Athe-

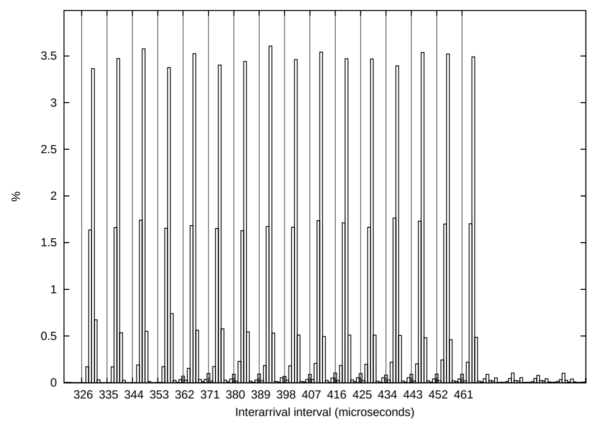

Figure 10: Timestamping by the Netgear-Prism GT card.

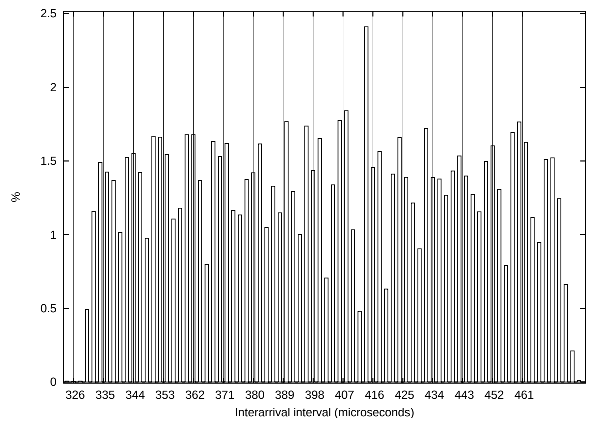

Figure 11: Timestamping by the the D-Link-Atheros card.

ros cards. Compared to Figure 5, in this case timestamping is done on the wireless card. The histogram is less precise with peaks spaced every 2 μs, which suggests that the clock resolution is of that order.

We thus conclude that to achieve the best precision, we need to use the monitor station with the Netgear-Prism GT card with timestamping done on the card.

5. TESTING WIRELESS EQUIPMENT

In this section, we illustrate the capacity of the proposed method to analyze different performance problems of wireless equipment. The emphasis is not on particular behavior of wireless cards, which was already reported in the literature, but what our method can do with its precision. In this part, we have used Dell Latitude D620 with Intel Core 2 duo processor at 2.00 GHz running Linux Fedora Core 6, 2.6.18.1 kernel with HZ=100 as the monitor station.

5.1 Tests of access points

We measure access points in an inverse configuration: the wired host generates traffic to a wireless station and the monitor station captures and timestamps packets. Recall that precise measuring of commercial access points cannot be done with existing methods, unless one can instrument the access point kernel.

The access points are the following:

- Linksys WRT 54GL (802.11g) under OpenWRT firmware,
- Asus WL-500G Premium (802.11g) under DD-WRT firmware,
- D-Link DWL 7100 (802.11a+g) with SuperG extensions of Atheros.
- Cisco Aironet 1132 AG (802.11a+g),

We fix the source rate to 30 Mb/s to minimize the effect of the saturation at the access point, because some access points perform badly if the arrival rate of packets exceeds the capacity of the wireless link (around 33 Mb/s at the transport level). If not stated otherwise, the receiving card is Intel Centrino-IPW3945 and the monitor uses the Netgear-Prism GT card with timestamping on the card.

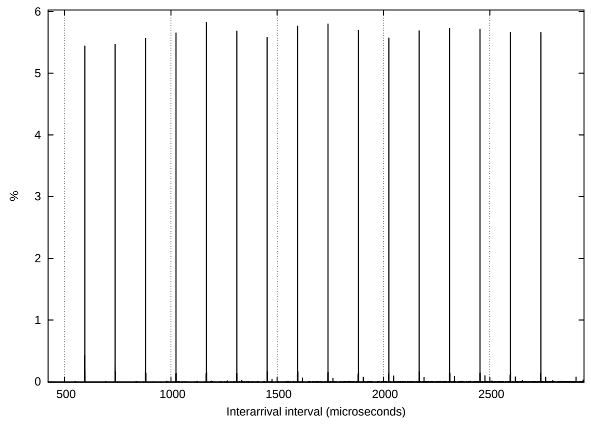

Figure 12: Linksys WRT 54Gl, default configuration.

Figure 12 shows the results for Linksys WRT 54GL. We can see 16 regular peaks spaced at 143 μs with the first one at 595 μs. These surprising values are consistent with the measured throughput of 7 Mb/s and an AIFS of 2 slots, a slot being equal to 143 μs. This behavior of the card comes from the configuration variable of the access point called `Wireless Distance`, which by default is equal to 20 km.

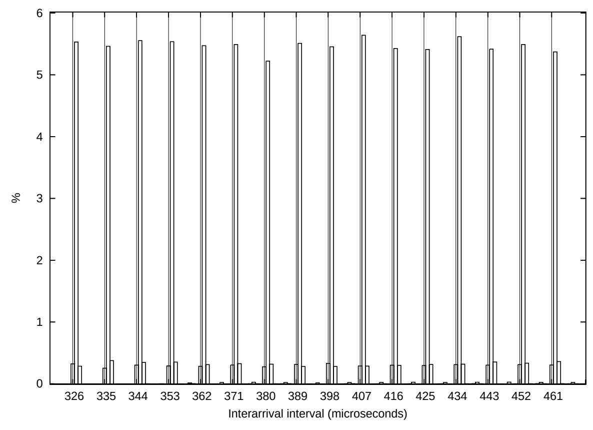

Figure 13: Linksys WRT 54GL, null variable of `Wireless Distance`

If this variable is set to 0, the slot becomes 9 μs and the throughput increases significantly. Figure 13 presents the histogram when this variable is set to 0. We can now see regular and narrow peaks with the first peak at 327 μs (only 1 μs difference from the standard value).

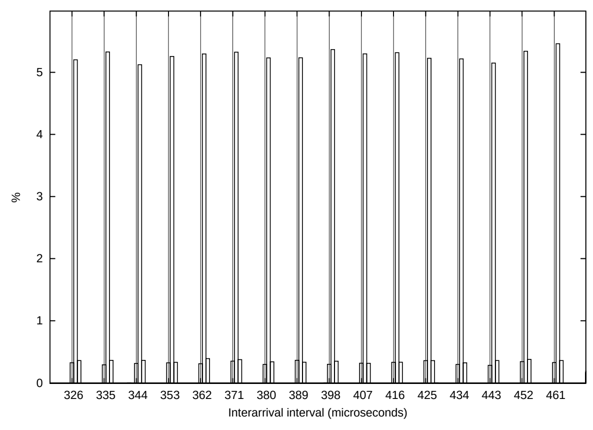

Figure 14: Asus WL-500G Premium.

The next tested access point is Asus WL-500G Premium with the histogram presented in Figure 14. The monitor card is Netgear-Prism GT with timestamping on the card. We can observe that the histogram is similar to that of Linksys WRT 54GL with the `Wireless Distance` variable set to 0—it is fairly regular with narrow peaks, the first one starting at 327 μs. As a side effect, these two experiments confirm the precision of measurements, because almost 90% of the values of each peak is contained in the central value. So, wider peaks denote either a variation of the timings on the sender side (when the same method is used) or a larger variation of the timestamping delay (when host timestamping is used).

Figure 15 presents the histogram for the D-Link DWL 7100 access point. The first peak is at 326 μs, which closely fits the standard. However, we can note a superposition of three histograms corresponding to different bit rates.

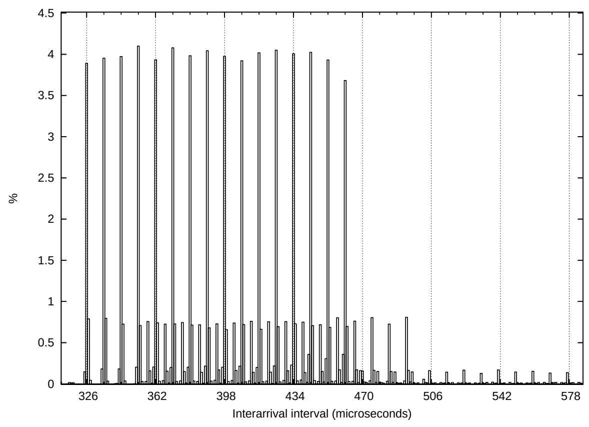

Figure 15: D-Link DWL 7100.

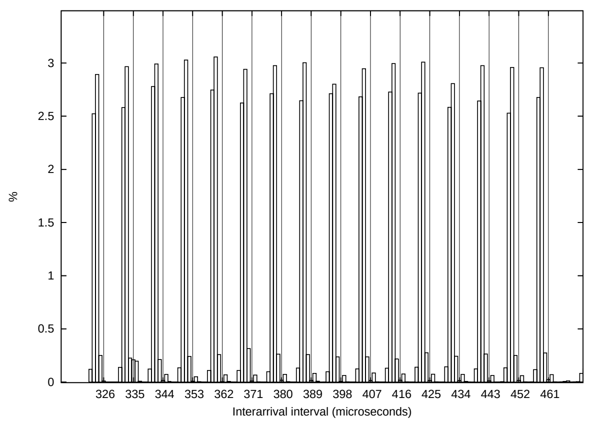

Figure 16: Cisco Aironet AP 1132 AG.

Finally, we have analyzed the Cisco Aironet AP 1132 AG access point. Figure 16 shows its results: the peaks appear too early compared to the standard values by 4 μs, the first peak appearing at 322 μs. This behavior was surprising and lead us to a more detailed analysis. We have set up a measurement session to gather statistics on the behavior of the access point in contention with a test station. There is a downlink UDP flow from the Cisco access point to a second wireless station and the test station with the Intel Centrino-IPW2915 card sends an uplink flow to the wired host through the access point. In this set up, the access point is a sender and a receiver while the test station is only a sender. A third wireless station acts as a monitor to measure transmission times.

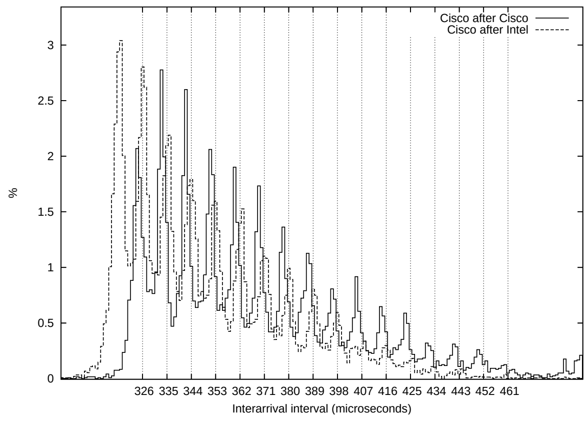

Figure 17: Histograms of the Cisco Aironet AP 1132 AG contending with the Intel Centrino-IPW2915 card.

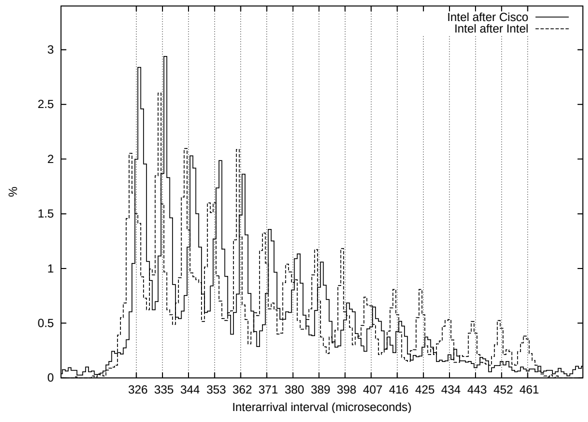

Figure 18: Histograms of the Intel Centrino-IPW2915 card contending with the Cisco Aironet AP 1132 AG.

Figure 17 presents the histogram of the transmissions done by the access point after its previous transmission (Cisco after Cisco) and a second one corresponding to the transmissions by the access point after a transmission by the test station (Cisco after Intel). Similarly, Figure 18 presents the histogram of the transmissions done by the test station after its previous transmission (Intel after Intel) and a second one corresponding to the transmissions of the test station after a transmission by the access point (Intel after Cisco). We can observe that the test station sends its frames independently of which entity sent the previous frame, but the access point sends its frames one slot earlier, if the previous transmission was done by the test station. The test station sends its frames more often after a transmission by the access point, so the contention is finally in favor of the test station: its throughput is 13.5 Mb/s vs. 13 Mb/s of the access point. Note also that the distribution of interarrival intervals has changed—it is no longer uniform, because the observed intervals have a complex distribution with a variable part, which is the minimum of a uniformly distributed random variable and the residual backoff time, resulting in a higher probability of short values.

Note that this comparison reports on the performance behavior of contending wireless stations—the measured transmission time includes the random backoff in the presence of two stations.

5.2 Tests of wireless cards

We have measured several wireless cards plugged into the source station in the standard configuration (cf. Figure 1). The access point is D-Link DWL 7100 and as previously, Dell Latitude D620 acts as a monitor.

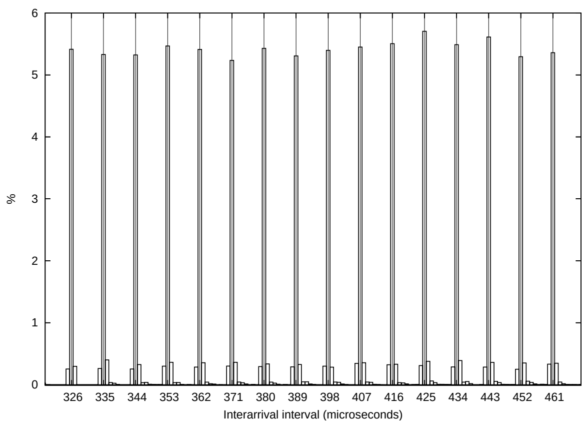

Figure 19: Histogram of Intel Centrino-IPW3945 measured with the D-Link access point.

The first card is the Intel Centrino-IPW3945. Figure 19 presents its histogram when it uses the D-Link DWL 7100 AP. It is perfectly regular with the first peak at 326 μs.

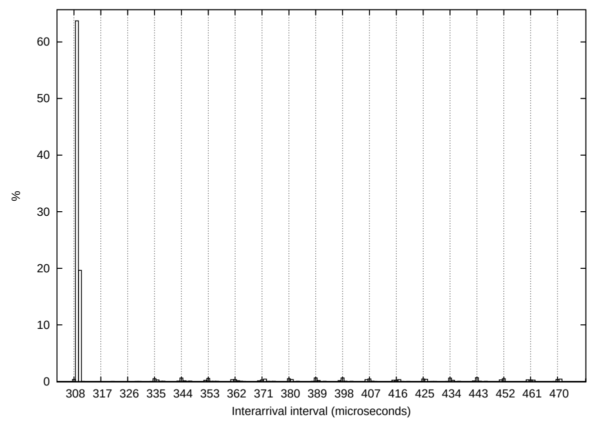

Figure 20: D-Link card in *Frame Bursting* mode.

Figure 20 presents the histogram of the D-Link-Atheros card that supports SuperG extensions with *Frame Bursting*: it sends bursts of frames separated by the SIFS interval. We can see an important peak at 309 μs corresponding to the frames sent without contention (90% of frames) and a series of small peaks representing frames sent after a contention interval.

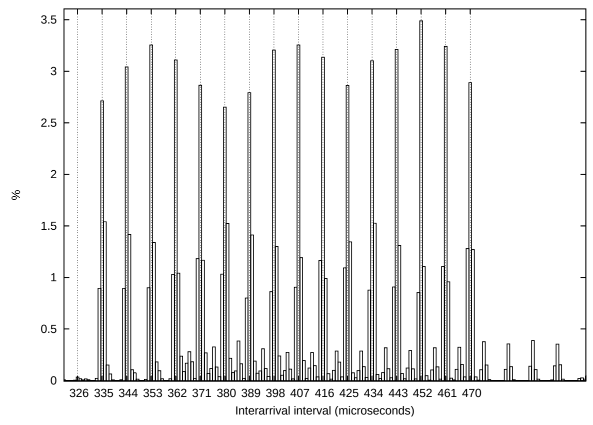

Figure 21: D-Link card, regular mode.

We can compare this behavior with Figure 21 that shows a corresponding histogram without *Frame Bursting*. There are two superimposed histograms, the second one beginning at 367 μs represents frames sent at 48 Mbit/s. This comes from a rate control algorithm called *Sample Rate*—periodically, the card sends frames at a lower rate to determine the right rate to use [3]. Such behavior probably shows an incorrect implementation of this algorithm, because

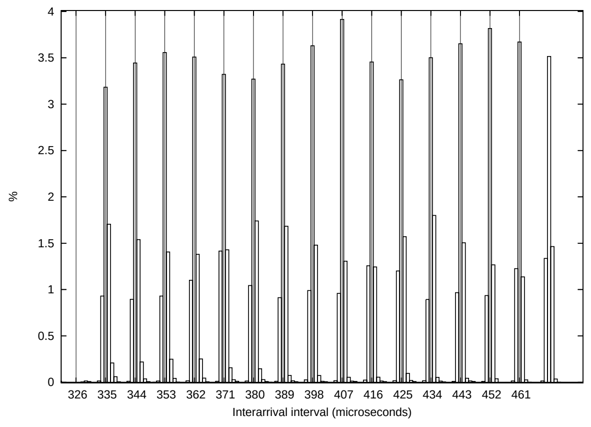

Figure 22: D-Link card, fixed bit rate of 54 Mb/s.

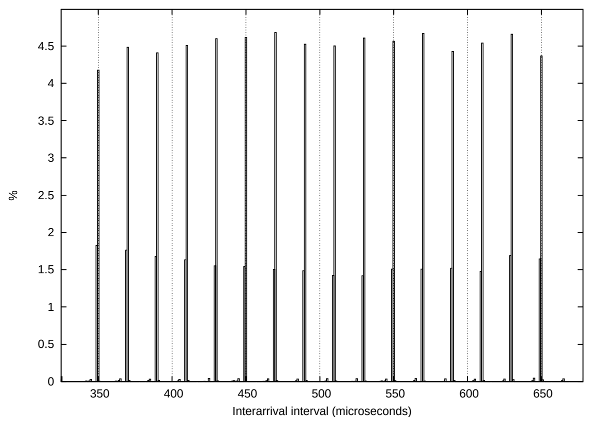

Figure 23: Linksys-Broadcom card.

in the conditions of the experimentation, the card should not switch to a lower rate, because it will not increase the throughput anyway. When we fix the bit rate, we obtain a much regular histogram (cf. Figure 22). The first peak is at 335.2 μs, which is 9 μs (1 slot) greater that the standard value.

Finally, we have tested the Linksys-Broadcom wireless card under an experimental driver. We have observed that by default the card operates at 11 Mb/s (802.11b) and switches to 802.11g only if we fix the bit rate to 54 Mb/s. Figure 23 presents its histogram. We can see several anomalies:

- the first peak is at 350 μs instead of 326,
- the peaks are separated by 20 μs instead of 9 μs.

It appears that even if the card transmits at 54 Mb/s with the contention window of 16, it applies the time intervals corresponding to 802.11b (in particular the slot of 20 μs). Such timings are consistent with the throughput we have measured, which was around of 22 Mb/s instead of 29 Mb/s.

5.3 Discussion

We summarize here most common performance issues that we could identify with our method:

- *Interframe intervals and backoff.* Some cards do not respect the minimum interframe intervals (too early access to the channel) or use wrong values of the backoff slot. For almost all the wireless cards, the distribution of the backoff follows fairly closely the uniform distribution, but in the past, we have discovered some bugs in the implementation of the pseudo-random generator, e.g. one card generated 15 values instead of 16 with a double probability of the first value. Since then, the bug has been corrected.

- *Bit rate adaptation.* The method can detect multiple bit rates used by wireless cards. The algorithm for setting the bit rate in function of transmission conditions is not specified in the standard. So, the manufacturers can implement proprietary solutions such as *Auto Rate Fallback* [13] or proposed improvements like *Sample Rate* [3], but usually they perform poorly, because 802.11 does not distinguish between collisions and missed transmissions. This situation cannot improve unless the standard radically changes the access method to something that allows to set up the bit rate in a more optimal way such as in *Idle Sense* [9].

- *Default configuration.* Advanced equipment often provides a sophisticated management interface, but when the default configuration values do not correspond to the most common cases, users may face performance problems. Our example with the `Wireless Distance` variable shows that the default configuration may correspond to an unrealistic situation and the name of a configuration variable may be counter-intuitive for changing the duration of the backoff slot.

- *Extensions of the standard.* Our method can help evaluating extensions that some manufacturers support by anticipating the advances in standardization. We have shown how *Frame Bursting* can be identified and evaluated.

6. CONCLUSION

In this paper, we have proposed *Interarrival Histograms*, a method for measuring transmission delays in 802.11 wireless networks using commercially available equipment. The method is simple, yet powerful—it allows us to identify the temporal characteristics of tested cards and access points with high precision. We have derived the estimation of the measurement error and experimentally validated the method by applying it to the measurement of the 100 Mb/s Ethernet. To illustrate the cases in which the method can help analyzing the complex behavior of 802.11 networks, we have measured several commercially available access points and wireless cards to show some anomalies and differences with the standard.

By gathering sufficient number of interarrival interval samples and by using timestamping with 1 μs resolution, the precision of the method is sufficiently high for fine-grain measurements of 802.11 networks. At the first glance, such a precision is surprising, but our derivation of the error estimation provides the explanation—the influence of random delays incurred during timestamping and interrupt processing is canceled out when long series of interarrival intervals are measured.

7. REFERENCES

[1] G. Berger-Sabbatel, A. Duda, O. Gaudoin, M. Heusse, and F. Rousseau. Fairness and its Impact on Delay in 802.11 Networks. In *Proc. of GLOBECOM 2004*, Dallas, USA, 2004.

[2] G. Bianchi, A. D. Stefano, C. Giaconia, L. Scalia, G. Terrazzino, and I. Tinnirello. Experimental Assessment of the Backoff Behavior of Commercial IEEE 802.11b Network Cards. In *Proc. of IEEE INFOCOM 2007*, 2007.

[3] J. Bicket. Bit-rate Selection in Wireless Networks. Master's thesis, Massachusetts Institute of Technology, February 2005.

[4] D. Cavin, Y. Sasson, and A. Schiper. On the Accuracy of MANET Simulators. In *Proc of POMC '02, Second ACM International Workshop on Principles of Mobile Computing*, pages 38–43, 2002.

[5] I. Dangerfield, D. Malone, and D. J. Leith. Testbed Evaluation of 802.11e EDCA for Enhanced Voice over WLAN Performance. In *Proc. of WiNMee, Second International Workshop on Wireless Network Measurement*, Boston, Massachusetts, USA, 2006.

[6] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 2003.

[7] Y. Grunenberger, M. Heusse, F. Rousseau, and A. Duda. Experience with an Implementation of the Idle Sense Wireless Access Method. In *Proc. of ACM CoNEXT, to appear*, 2007.

[8] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *Proc. of INFOCOM 2003*, 2003.

[9] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs. In *Proc. of ACM SIGCOMM 2005*, August 2005.

[10] M. Heusse, P. Starzetz, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Bandwidth Allocation for DiffServ Based Quality of Service over 802.11b. In *Proc. of GLOBECOM 2003*, November-December 2003.

[11] M. Heusse, P. Starzetz, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Scheduling Time-Sensitive Traffic on 802.11 Wireless LANs. In *Proc. of QoFIS 2003, LNCS 2856*, Stockholm, Sweden, 2003.

[12] IEEE. IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[13] A. Kamerman and L. Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Tech. Journal*, 2:118–133, 1997.

[14] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. Technical Report TR2004-507, Dept. of Computer Science, Dartmouth College, June 2004.

[15] D. Malone, D. J. Leith, and I. Dangerfield. Verification of Common 802.11 MAC Model Assumptions. In *Proc. of PAM 2007, Passive and Active Measurement Conference*, Louvain-la-neuve, Belgium, 2007.

[16] A. D. Stefano, G. Terrazzino, L. Scalia, I. Tinnirello, G. Bianchi, and C. Giaconia. An Experimental Testbed and Methodology for Characterizing IEEE 802.11 Network Cards. In *Proc. of WOWMOM '06, IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks*, pages 513–518, Niagara-Falls, Buffalo, NY, USA, 2006.