TCP & UDP Performance over a 2.4 GHz Wireless LAN

George Xylomenos and George C. Polyzos

\{xgeorge,polyzos\}@cs.ucsd.edu

Computer Systems Laboratory
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, California, 92093-0114

September 8, 1998

\(^1\)This report presents research that was supported in part by Sprint Corporation.
Abstract

This report presents a comprehensive set of measurements made on a commercial 2.4 GHz DSSS Wireless LAN and analyzes its behavior in detail. We examine the impact on performance of issues such as system heterogeneity, bidirectional (TCP) communications and error modeling, that have not been discussed elsewhere. We describe the testing environment, justify the design of our test suite, and explain the post processing steps in detail. Our experiments gathered enough data to uncover many problems with TCP and UDP performance in this system. We examine the causes of these problems and discuss the effectiveness of alternative solutions based on the data gathered.
## Contents

1 Introduction 1

2 Goals 1

3 Experimental Setup 2
   3.1 Hardware .......................................................... 2
   3.2 Software .......................................................... 3
   3.3 Environment ...................................................... 4

4 Test Description 5
   4.1 Individual Tests .................................................. 5
   4.2 Test Suite .......................................................... 6

5 Output Summary 6

6 Analysis of Test Results 8
   6.1 General Remarks ................................................. 8
   6.2 Scenario 1 .......................................................... 8
   6.3 Scenario 2 .......................................................... 14
   6.4 Scenario 3 .......................................................... 20
   6.5 Scenario 4 .......................................................... 22
   6.6 Scenario 5 .......................................................... 25
   6.7 Scenario 6 .......................................................... 31

7 Discussion 33

8 Conclusions 36
1 Introduction

The field of wireless communications has experienced an explosive market growth worldwide in recent years, in areas such as cellular telephony, satellite communications and wireless LANs (WLANs). Wireless LANs are becoming increasingly popular for high speed communications over small areas that are hard or impossible to wire for conventional networking. With numerous vendors offering WLAN systems at constantly dropping prices and a proposed standard (IEEE 802.11 [2]) that will eventually enable interoperability among the newer versions of such products, WLANs are becoming quite attractive solutions. Typically, WLANs emulate wired LANs such as Ethernet, so that they may be deployed as drop-in replacements of their conventional counterparts. Naturally, this means that they can be easily connected to the Internet to extend its reach over wireless channels.

Integrating WLANs into the Internet is not completely transparent, as their lower available bandwidth and higher error rate compared to wired LANs, make their presence more obvious to the Internet user. Packet losses due to errors have unfortunate effects on traditional Internet protocols such as TCP, even under moderate loss rates [1]. Although the behavior of higher layer protocols under wireless link errors can be demonstrated and analyzed using theoretical error models and simulations, the actual performance of these protocols in production systems is much less understood. Since proposing appropriate solutions for performance problems requires a clear understanding of the underlying process causing each problem (the errors in this case), measuring the performance of actual systems under a variety of conditions is an important step towards improving those systems. Careful test design also enables subsequent performance comparisons among the systems with and without the proposed improvements, using regression testing.

In this report we present a comprehensive set of measurements of a commercial WLAN and analyze its behavior, extending published work in many directions. In Section 2 we outline the goals of our measurement process to establish a basis for test design. Section 3 details our experimental setup, while Section 4 explains the rationale behind each individual test and the complete test suite in detail. Section 5 describes the data gathered during our tests and the post processing steps that we performed. These data are used in Section 6 to describe in detail the performance of both unidirectional (UDP) and bidirectional (TCP) communications. We review these results and discuss in more detail their implications in Section 7. We conclude with a summary of our findings in Section 8.

2 Goals

The basic aim of our tests is to compile a comprehensive set of data characterizing the performance of a WaveLAN [10] WLAN system in terms of throughput and error behavior under various operating conditions. Although measurements of similar systems have been published before [4, 5, 8], we believe that additional measurements were needed to further clarify system performance. Our tests extend these published results in multiple dimensions:

System Heterogeneity: We conducted tests between hosts with varying processing power and different implementations of the wireless interfaces. By varying both these parameters we can examine their influence on performance, in contrast to earlier work that kept one or both of these parameters constant.

New Implementations: Earlier results described systems operating at the 900 MHz band, while our work examines the newer 2.4 GHz band wireless interfaces, that claim enhanced performance. In addition, we use hosts with faster processors that can potentially offer more load to the WLAN, thus exhibiting the true peak performance of the system.

Bidirectional Communications: Our measurements show the performance of both unidirectional (UDP) and bidirectional (TCP) communications, in contrast to the purely unidirectional results reported earlier. Introducing bidirectional traffic through TCP acknowledgments means lower throughput and the potential for collisions on the shared medium.

Error Modeling: Earlier measurements have been used to establish error models that could be used to simulate WLANs [8]. We extend this work by providing more comprehensive measurements (multiple combinations of locations and packet sizes), in an attempt to create more detailed and accurate models. Our TCP measurements can also be used to assess the accuracy of unidirectional models for bidirectional communications.
A final distinguishing characteristic of our tests is that, unlike previous reports, we did not use an Operating System based on the BSD distributions, choosing instead the Linux OS which we were more familiar with. Due to the differences in both generic networking code and wireless interface device drivers, our results may not be directly comparable to those previously reported. While the Linux networking code does not showcase the cutting edge of Internet protocol development, it is updated regularly and its use is widespread enough to represent fairly well the average Internet host.

Our measurements are limited in some dimensions due to practical constraints. We have not tested all combinations of hosts and wireless interfaces, since this would tremendously increase human and hardware requirements. Our tests were made over one hop only, the WLAN link, even though TCP in particular has been shown to have quite different behavior when the wireless link is part of a longer path [1], because we could only retain complete network control over this link. We hope that our one hop results can be used to create simulation models to assess TCP performance over longer paths under more controlled conditions. We also did not study the effects of mobility on our WLAN, since mobility tests are hard to reproduce, and also because we wanted to focus on the effects of the (fixed) wireless link alone on Internet protocols. Note that the range and form factor of our WLAN interfaces makes their use on the move unlikely. We did not measure one or two way delay in our experiments. Unlike digital cellular systems where considerable delay is introduced by physical layer coding, moderate propagation delays and low transmission speeds, in a WLAN delay is composed of (a) very short propagation and transmission delays, and (b) the MAC (Medium Access Control) layer contention time, which should be negligible for unidirectional (UDP) tests and short for bidirectional (TCP) tests with one way data traffic. An interesting topic, left for further research, is the growth of the MAC layer contention time when multiple hosts are competing for access to the shared medium. Finally, we did not reproduce tests that intend to determine the effective range [4] or the behavior under heavy interference [5] of the WLAN, focusing instead on system behavior under realistic operating conditions.

3 Experimental Setup

3.1 Hardware

Three host systems equipped with wireless interfaces were used (in pairs) for our experiments. During each experiment only the two communicating hosts had their wireless interfaces active. To eliminate local interference we switched off our WLAN bridges. Our location was also explored with a WLAN monitoring utility to discover if any other WLANs were operating nearby, and no external transmissions were detected. All hosts were connected via a standard wired Ethernet LAN with other laboratory machines and the Internet. By default the wired LAN was used for all communications, and only direct communication between the experimental machines used the WLAN.

The WLAN used was a Digital RoamAbout 2.4 GHz Direct Sequence Spread Spectrum (DSSS) system, which is an OEM version of the AT&T/NCR WaveLAN [10]. Two other variants of this system exist, a 900 MHz DSSS and a 2.4 GHz Frequency Hoping Spread Spectrum (FHSS) version. The wireless interfaces for this system are available in ISA and PCMCIA card versions, for desktop and laptop machines, respectively. The two interfaces slightly differ in their implementation due to the use of different Ethernet controllers (Intel 82586 for the ISA card and Intel 82593 for the PCMCIA card) and radio module packages (this may influence the effectiveness of their antennas), but they both implement the same Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC scheme and are fully compatible over the air. Both cards have a nominal bandwidth of 2 Mbps, the same as other variants of the system, but the PCMCIA cards have been shown to be slightly slower than their ISA counterparts in the 900 MHz systems [8].

The MAC protocol is similar to other CSMA variants in that hosts are required to wait for a silent medium before transmitting. While each card is transmitting it is hard to detect collisions, which means that after a collision the medium remains used (and wasted) until the longest of the collided packets ends its transmission. Since the cost of collisions is higher than in CSMA/CD systems were collided transmissions are aborted on detection, a collision avoidance scheme is employed. Each host that wants to transmit waits until the medium becomes silent (plus a short interframe gap time), and then chooses a random time slot in which to start transmitting. If the medium has not been seized by another transmitter until that slot, the host seizes the channel. The random slot is chosen from a time window

\[1\] The bridges periodically transmit packet bursts due to their transparent bridging algorithms that discover and update the network topology.
that is set to a small value after seizing the channel, and is doubled for every consecutive failed attempt to seize the channel (i.e. the contention windows expands exponentially). If multiple hosts seize the channel during the same time slot, an undetected collision occurs.

In terms of interfacing with the host machine, these cards emulate a wired Ethernet in terms of both programming and packet formats (i.e. same MAC headers, CRCs, and maximum packet size of 1500 bytes). An additional feature is that after receiving a packet, the interfaces can report to the driver some radio related metrics:

**Signal Level:** Measured at the beginning of each transmission using the receiver’s automatic gain control circuitry, indicates how powerful is the received signal.

**Noise Level:** Measured during the interframe pause after reception ends in the same manner as signal level, indicates how powerful is the background noise.

**Signal Quality:** Derived from the information used for antenna selection when reception begins (each card uses two antennas).

The signal and noise level metrics are reported in hardware specific numbers rather than in some standard unit, so they only make sense in terms relative to their highest and lowest values. The signal quality metric indicates the magnitude of the ratio between the main and secondary lobes of the received signal [10], i.e. it determines the clarity of the signal with respect to multipath propagation and fading.

The three host systems used had two network interfaces attached (one wired and one wireless), each of which was assigned a different IP address and DNS name. Table 1 shows the names and main characteristics of each host. In the following we refer to each host by its wired name, since this is the name by which it known to the outside world. Note that the first two systems (IOS and MYKONOS) are desktop PCs while the third (SYROS) is a laptop. Two desktop machines with different processors were employed to determine the effect of processing power on WLAN performance. The laptop has roughly the same processing power as the slower desktop (the processor runs at a lower frequency but has a larger cache memory) but differs in the wireless interface used.

<table>
<thead>
<tr>
<th>Wired Name</th>
<th>Wireless Name</th>
<th>Processor</th>
<th>Memory</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS</td>
<td>CROTONE</td>
<td>Pentium MMX 200 MHz</td>
<td>64 MB</td>
<td>ISA 2.4 GHz DSSS</td>
</tr>
<tr>
<td>MYKONOS</td>
<td>SPEZZANO</td>
<td>Pentium 166 MHz</td>
<td>64 MB</td>
<td>ISA 2.4 GHz DSSS</td>
</tr>
<tr>
<td>SYROS</td>
<td>SALERNO</td>
<td>Pentium MMX 150 MHz</td>
<td>48 MB</td>
<td>PCMCIA 2.4 GHz DSSS</td>
</tr>
</tbody>
</table>

Table 1: Hosts used for the experiments

3.2 Software

All hosts ran the Linux operating system, kernel version 2.0.32. The wireless interfaces were controlled by the WaveLAN drivers included with the kernel and the corresponding PCMCIA package, compiled as loadable kernel modules. During the tests the hosts were in multiuser mode, but no users were logged in and no user tasks were executing. Similarly, since the tests were performed late in the evening, the wired Ethernet that was used for all host communications except the experiments, was very lightly loaded. All the experiments were performed over two consecutive days to minimize variations due to external influences. Of course, supplementary measurements and tests also took place during the weeks before and after the recorded tests.

Since we wanted to approximate as closely as possible the behavior of a system running a stock kernel and system software, we only modified slightly the wireless interface drivers to record and report detailed statistics. When compiled with the appropriate options, both ISA and PCMCIA drivers record a histogram of the radio metrics that the interface reports (signal level, noise level and signal quality). These histograms are simply three arrays with each element corresponding to a single value of each of the reported metrics, updated every time a packet is received by

---

2 These units are the same though, therefore signal and noise metrics can be directly compared to each other.
incrementing the appropriate array elements. We added a private (i.e. only applicable to this driver) `ioctl()` system call option that can reset these histograms to all zeroes. A second option was added to dump these histograms plus all interface statistics maintained by the kernel (number of received and transmitted packets, collisions, errors, etc.). The performance of the drivers with or without detailed statistics gathering was virtually the same, as verified by preliminary tests. Access to these facilities was made possible by the program `iwpriv` which is part of the Linux wireless tools. The program did not need to be modified (it does not in itself interpret the options). The histograms were reset by typing `iwpriv setstats` and dumped, along with all other interface statistics, by typing `iwpriv getstats`.

The main program used for testing was the `ttcp` benchmark that sends a number of packets of a specified size to a receiver using either TCP or UDP. `ttcp` reports at the end of the test various transfer and OS related metrics, separately at the sending and receiving ends. We modified it by adding an option for UDP transfers, where each outgoing packet is stamped with a sequence number, so that the receiver can detect packet losses. In this mode the receiver prints out the size of each run of lost and received packets as they occur, plus a total number of losses at the end. These outputs can be used to compute both a mean loss rate and the distribution of periods of loss and no loss, so as to formulate a two state model of the link, with a good and a bad state [8]. This version of the benchmark was named `ettcp`.

We also used two other programs to monitor the tests and gather statistics, without modifying them in any way. The first, `tcpdump`, records a detailed log of packets that are sent and received by an interface, and is especially useful for watching the progress of TCP sequence numbers and acknowledgments. We used `tcpdump` to capture all packets moving through the wireless interfaces, so as to detect any external packets, but we did not see anything beyond our own test transmissions. The second program, `nstat`, reports statistics kept at the IP, ICMP, UDP and TCP protocol layers, but since it aggregates statistics across all interfaces, it was mostly used to verify that no unusual network activity was taking place at the wired Ethernet interface. We did not use any of the sources of link layer statistics provided by Linux (namely, `ifconfig` and the pseudo file `/proc/net/wireless`), as our `ioctl()` options reported all these statistics and more.

### 3.3 Environment

The measurements were made at the 5th floor of the Applied Physics and Mathematics building at the UCSD campus. Figure 1 shows an approximate floor plan of the area where the experiments took place (the plan is not in scale). For each test, we placed each of the participating hosts at one of Locations 1, 2 and 3 (marked as *Loc.n* in the figure) in rooms 5313, 5414 and 5438, respectively. Since these rooms are either laboratory facilities or machine rooms,
they contained numerous other computers and hardware, but no direct sources of RF signals. The distance between Locations 1 and 2 is approximately 45 feet, between Locations 2 and 3 again about 45 feet, and between Locations 1 and 3 about 60 feet.

We executed the same set of experiments for six different host and location combinations or scenarios, described in Table 2. The ordering of the hosts on the table is not significant, as we ran each test in both directions. Note that IOS remains at Location 1 during all tests, while MYKONOS and SYROS move. Scenarios 1 and 2 show the baseline performance of the WLAN under optimal circumstances, since the hosts were placed next to each other, with the first scenario using only ISA cards and the second mixing ISA and PCMCIA cards. For Scenarios 3 and 4 we moved MYKONOS and SYROS across the corridor, and basically repeated Scenarios 1 and 2, but over a longer distance and with obstacles between the interfaces. For Scenarios 5 and 6 we kept the ISA cards at the same places as in Scenarios 3 and 4, and moved SYROS to a location far from both, but further away from IOS.

Before the recorded tests, we used a monitoring utility to estimate the ambient noise of the environment and the signal levels that we would get under various scenarios. While adequate connectivity was maintained between all chosen locations, longer distances and more obstacles reduced signal level and slightly increased noise, as expected. The locations chosen do not represent the limits of the system (we found instead that connectivity could be maintained over longer distances and more obstacles) but they introduce enough variability for our experiments. They also do not suffer from excessive multipath fading, i.e. the signal quality metrics were always high, regardless of the exact location of the external radio modules with respect to their hosts. We kept each host immobile for the duration of each test (to avoid mobility induced effects) and over all tests in each scenario. Our intention was to reproduce an average office environment under reasonable operating conditions.

4 Test Description

4.1 Individual Tests

Each test consisted of executing `ettcp` with appropriate parameters, and recording statistics before, during, and after the run. A script was executed on both hosts participating in the test, with slightly different parameters for the sender and the receiver, to automatically execute `ettcp` and record data. The main script parameters were the direction of the transfer, the name of the peer, the packet size and the protocol to be used. The packet size parameter was total IP datagram size, so depending on the protocol, the data packet size passed to `ettcp` by the script was either 40 or 28 bytes less, for TCP and UDP, respectively. In order to distinguish between data from multiple tests, we also passed to the script the number of the scenario in execution and the number of the current test (since tests were repeated for reliability), so that data output files could be labeled unambiguously.

The script would first reset and dump private `ioctl()` data (using `iwpriv`), then dump `nstat` data, start `tcpdump` in promiscuous mode to record all packets passing through the wireless interface, and finally start `ettcp` with appropriate packet size, host and protocol parameters, to transfer 10,000 packets without pause. Although these tests were short in duration (at the peak speed of 2 Mbps a 10,000 packet transfer with 1500 byte UDP packets should take roughly one minute), the amount of data transferred was considerable (1–15 MBytes per test). Even if these runs

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Host A/Location</th>
<th>Host B/Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IOS/Loc.1</td>
<td>MYKONOS/Loc.1</td>
</tr>
<tr>
<td>2</td>
<td>IOS/Loc.1</td>
<td>SYROS/Loc.1</td>
</tr>
<tr>
<td>3</td>
<td>IOS/Loc.1</td>
<td>MYKONOS/Loc.2</td>
</tr>
<tr>
<td>4</td>
<td>IOS/Loc.1</td>
<td>SYROS/Loc.2</td>
</tr>
<tr>
<td>5</td>
<td>MYKONOS/Loc.2</td>
<td>SYROS/Loc.3</td>
</tr>
<tr>
<td>6</td>
<td>IOS/Loc.1</td>
<td>SYROS/Loc.3</td>
</tr>
</tbody>
</table>

Table 2: Description of testing Scenarios
were not long enough for the system to reach its steady state, we believe that they are more representative of realistic operating conditions than huge file transfers that never occur in practice.

To avoid TCP using a different segment size than the one desired, an appropriate `ettcp` option (that sets a corresponding TCP socket option) was employed. After the transfer was over, `tcpdump` was stopped, and then `nstat` and `iwpriv` were called again to dump their statistics, so that the changes between the beginning and the end of the test could be later examined. Apart from the data packets (and their retransmissions for TCP) a few control packets for UDP (to start and end the transfer) and acknowledgment packets for TCP were expected to be seen on the `tcpdump` traces. Each of the four programs employed directed its output to a separate file, with a unique name composed by combining the script parameters and an identifying letter signifying which program produced it. These data files were later analyzed in an off-line mode. Note that data for each test were recorded separately at both the sending and receiving sides.

4.2 Test Suite

Each test was repeated 5 times with the exact same parameters, so that we could estimate variance between runs. All tests were run in both directions, by having each of the two peers assume both the roles of sender and receiver. Tests that were blocked for some reason (due, for example, to loss of UDP control packets) and would not terminate, were manually stopped and repeated, i.e. only completed tests were recorded. Before each sequence of repeated tests, the link was brought down and then up, and `ping` was used to establish connectivity before starting the test script. All tests were performed for both TCP and UDP, and for 4 different IP datagram sizes (including IP and TCP or UDP headers): 100, 500, 1000 and 1500 bytes (the maximum possible). Since 8 files were produced by each run (4 at each side), the total number of data files produced was 3840 (8 files, 5 repetitions, 2 directions, 2 protocols, 4 packet sizes, 6 scenarios) for the 480 tests that we performed.

The test suite was designed to capture various statistics of interest for our purposes. The scenarios show the performance of the system when heterogeneous systems and interfaces communicate over various distances. Executing tests in both directions shows if there are any performance asymmetries. The various packet sizes show the effects of various amounts of overhead and packet error probability (longer packets are more likely to be corrupted) on total throughput. Since only UDP performance measurements have been published [4, 5], our TCP tests show for the first time the effects of bidirectional traffic (due to acknowledgments) on a shared medium WLAN. Finally, by repeating each test multiple times, we could estimate by looking at the variance between the results how statistically significant our observations were.

5 Output Summary

Each of the programs used produced a separate data file for each test, suitable either for machine or human consumption. Accordingly, we wrote software to present the former in a human readable format, and to extract from the latter the data needed for automated analysis. In this section we describe the output of each program used, and some of the post processing steps that we took to transform the raw data into the results that we present below. Since we cannot claim that we have exhaustively analyzed the available raw data, we present them in detail so that other researchers can use them or extend our results.

The main testing program, `ettcp`, produces a variety of metrics at the conclusion of each test, including elapsed time, bytes transferred and total throughput, plus OS related metrics such as I/O calls, system and user run time. For UDP tests, our code also reported the length of each sequence of lost and received packets (using sequence numbers to detect gaps), and a total number of received and lost packets. We generally used the metrics reported by the data receiver rather than those of the sender for analysis, since for UDP results only the receiver reported losses. In general, the sender and receiver reported slightly different numbers, especially for UDP where there is no feedback from the receiver to confirm when the data transfer is finished.

---

3One of the tests was later determined to be deficient in that one of the data files was not complete. To avoid complicating post processing, the data files for this test were replaced by those of another repetition of the same test.
The `ettcp` outputs were in human readable form, with each metric preceded by a short description, so post processing mainly consisted of extracting, aggregating and reformatting these metrics for consumption by other programs. For the UDP tests where detailed loss traces were available, we calculated the probabilities for runs of lost and received packets to be of a given size, by aggregating all 5 repetitions of each test into one sample. The outputs were 2 files for every test, one each for lost and received packet runs, showing the PDF and CDF of the run sizes. Since all tests consisted of sending 10,000 packets (with sizes depending on the test) both the absolute number of occurrences of each run size and the PDF/CDF functions are comparable across tests. If we assume that wireless link errors follow a two state process, with a good state during which packets are received correctly, and a bad state during which packets are lost or corrupted, the CDFs of lost and received packet runs can be used to simulate this process [8].

Another post processing direction for the `ettcp` output is to calculate mean, minimum and maximum values for the reported metrics across all repetitions of each test, again at the receiver. The metrics used were throughput, total bytes received, total time elapsed, plus packets lost and received in absolute and relative (percentage) terms. The throughput metrics are comparable across tests since the throughput units (KBytes/sec) do not depend on packet size, while bytes received and time elapsed are comparable only across tests using the same packet size. The average lost and received packets statistics are also comparable across tests as we used the same number of packets throughout.

An examination of throughput and loss rate metrics can reveal what is the optimal packet size for the WLAN under examination, i.e. where overhead (which favors long packets) and loss (which favors short packets) are best balanced. By also showing minimum and maximum values we can estimate how reliable our averages are.

The `iwpriv` outputs (i.e. the data returned by the private `ioctl()` option) are just a sequence of integers, one for each metric tracked by the wireless interface driver, and then the histogram values for signal and noise level and signal quality (each histogram entry shows the number of packets received with the corresponding metric having that value). For each test we actually recorded in the output file first the values before the experiment, and then the values after the experiment. Thus, we had to write code that would subtract the initial from the final values to show the net effects of the test, and other programs that would combine the corresponding metrics so that we could aggregate or calculate mean/minimum/maximum metrics across all repetitions. Note that in bidirectional tests we also have packet reception at the sender (the TCP acknowledgments) so it makes sense to also process sender files. We also wrote a program to present the data in human readable form, by prepending each metric with its description.

From the aggregate files one direction for analysis is to study the metrics for transmitted and received packets and for detected collisions, and see how they differ among the data sender and receiver. For example, the numbers of data packets transmitted by the sender and received by the receiver will not match when packets are lost, and this can be studied in both directions for TCP tests, where these differences indicate the amount of actual loss encountered on the link. By comparing this figure with the loss seen in the corresponding UDP tests, we can estimate the amount of undetected collisions caused by the bidirectional TCP traffic. Another option is to study the noise and signal level histograms and compare them between scenarios, to see how different locations, hosts and interfaces influence them.

The `nstat` output files are in human readable form, and as in the `iwpriv` outputs, they included values before and after a test. Although we wrote programs to calculate net test effects, aggregate numbers, and reformat them in human readable form, these results were not analyzed further as they reflect statistics aggregated across all interfaces of a host. Finally, `tcpdump` output files, which were recorded in raw format could be presented in human readable form by executing `tcpdump` in an off-line mode. These files record complete traces of packets passing through the interface on both directions, and they are very useful for TCP tests were they show the progress of the transfer in terms of data packet and acknowledgment sequence numbers sent and received. It is interesting to plot side by side the data and acknowledgment packets as seen on both sides of the transfer, and see the gaps and retransmissions when packets are lost due to collisions. Since these tests could not be aggregated in any meaningful way, the main use of these data is to explain why a particular throughput or loss result arises, on a case by case basis. Our processing programs simply normalize the packet time stamps returned by `tcpdump` so that they start from zero when the test begins, and split data and acknowledgment sequence numbers so that they can be plotted separately.

---

4 Packet reception and loss only makes sense for UDP tests, default values are reported for TCP tests.
5 The signal quality metric was always near its peak value since we chose locations with low multipath fading, so it was not analyzed further.
6 `nstat` statistics were recorded solely to verify that no unusual network activity was taking place during the experiments.
7 This format is more compact and faster to record than human readable output.
6 Analysis of Test Results

6.1 General Remarks

During the experiments it became clear that in UDP tests with 100 byte packets, 90-95% of the packets sent by ettcp were never transmitted by the sender, as evidenced by the tcpdump traces at the sender side. Furthermore, these packets were never seen at the link layer, as reported by the iwpriv statistics. The most likely cause of this problem is that the networking code above the link layer drops these very short UDP packets due to buffer overruns, since despite their small size they incur the same per packet overhead as longer packets. We ran the same tests on a wired 10 Mbps Ethernet, where about 50% of the packets were lost. This seems to confirm our packet dropping hypothesis, since the wired interfaces could empty their buffers at a faster pace, so that less overruns would occur. This problem occurred with all hosts and interfaces, so we do not show below any 100 byte UDP results as their value is questionable. On the other hand, since TCP uses window based flow control, the sending process can only attempt to pass to the networking code without pause only as much data as the flow control window allows (up to 32 KB in our tests), so that packet dropping before transmission is much less likely than in the UDP case. Since our results are reasonable for the 100 byte packet TCP tests, given their large TCP/IP overhead factors, we present them along with the other TCP test results.

Another interesting phenomenon is that even though we used the ettcp option that sets a corresponding TCP socket option requesting data segments to be transmitted immediately, not all packets were of the same size (which was expected to be equal to the data segment supplied plus TCP/IP overhead). Occasionally, two or more segments would be combined in packets larger than expected (obviously this could only happen for the TCP tests with packet sizes below the maximum of 1500 bytes). This normally occurs when outgoing packets are delayed long enough before transmission for a queue to form at the sender. Since TCP keeps track of its transmission queue in terms of bytes rather than packets [9] it is reasonable to assume that the TCP option employed only prompts for immediate transmission of each new data segment if possible, but does not prohibit combining outstanding segments into larger packets later on. Although queue forming can be caused by MAC layer delays when contending for access to the channel (a rather plausible explanation with the bidirectional traffic of TCP), examination of tcpdump output shows that usually this event took place after sending a long run of packets, stopping, and then getting an acknowledgment. This sequence of events indicates that the sender exhausted its available transmission window (the minimum of the flow and congestion control windows), and while waiting for an acknowledgment that would enable it to advance these windows, a few more data segments were placed in the transmission queue, that were later combined into longer packets when the sender was allowed to proceed.

A final point of note is the issue of what exactly is reported by the drivers for the collision metrics, since collisions are not detected with CSMA/CA. On most TCP tests a large number of collisions is reported at one of the peers (at the receiver on ISA tests and at the ISA host on mixed ISA and PCMCIA tests), which grows with packet size. The metric reported could be the number of times that a host did not seize the medium because another host seized it earlier during the same contention period or the number of times that the medium was sensed busy. Neither of these theories explains all the results though, especially the discrepancies between the ISA only and the mixed ISA and PCMCIA tests. It is also unclear to what extent these numbers are determined by the hardware and its implementation of the MAC layer protocol and how much they are influenced by driver settings. The fact however that in all TCP tests the two peers substantially diverge on this metric, shows that this number cannot signify detected collisions and that we have to use other metrics (such as the difference between sent and received packets) to estimate the number of collisions.

6.2 Scenario 1

The first set of experiments employed two hosts with ISA cards, placed next to each other to avoid any signal degradation due to distance or obstacles. This setup should exhibit the peak performance of the ISA cards in our testbed, and also reveal any asymmetries due to differences in available processing power. Indeed, such an asymmetry is evident in Figure 2 which shows the mean, minimum and maximum packet loss rate (as a percentage) calculated from the 5 repetitions of each UDP test, as measured at the receiver. When the slower host (MYKONOS) is the sender, packet loss is negligible, but the faster host (IOS) seems to be overrunning the capabilities of the slower receiver, leading to a low packet loss rate. The acknowledgment itself could have been delayed due to MAC layer contention.

---

8 The acknowledgment itself could have been delayed due to MAC layer contention.
amount of loss (up to about 0.6%), but with significant variance. As expected, the loss rate is higher for longer packets, but instead of being a result of the bit error rate of the link, it is rather due to the processing limitations of the receiver, which become more apparent as the load increases.

To gain more insight into the loss process, we calculated the probability that a run of packets of a given length will be received or lost, using the `ettcp` output that shows these runs in detail, aggregating all 5 repetitions of each test in one sequence. The cumulative distribution function (CDF) of these probabilities for both received and lost packet runs can be used to represent (and simulate) this process using a two stage model. Nearly all losses are single packets, one packet giving enough time to the receiver to resynchronize with the sender. Looking at the CDF of the received packet runs in Figure 3 for the 1500 byte packet tests, in the direction from the slower to the faster host the CDF is a jagged line, and it remains so for packet runs of up to 50,000 packets. This is due to the negligible error rate which causes only a few received packet run sizes to be realized and plotted for the CDF. In the reverse direction, where more losses occur, there is one packet lost at least every 1000 packets (nearly all received packet runs are less than 1000 packets), strengthening the hypothesis that the sender overruns the receiver. The curve is smoother due to more frequent losses.

In Figure 4 we see the throughput (mean, minimum and maximum) for UDP and TCP, for various packet sizes. UDP throughput behaves as expected, slightly rising as packet sizes increase and UDP/IP overhead drops, with a peak data rate of about 225 KBps (1.8 Mbps), much higher than what has previously been reported. Since the previously reported values of 1.2–1.6 Mbps were achieved on hosts using the slower Intel 80486 processors, the higher throughputs that we observed could be explained by the faster hosts used. They could also be due in part to the 2.4 GHz version tested (previous tests used the 900 MHz version), or to the different host OS (Linux instead of a BSD derivative). Whatever the cause, the nearly symmetric performance of two hosts with considerable differences in processing power and the nearly nonexistent variance of the results, indicates that this is most likely the true peak

---

9 Test runs involving more than 50,000 back to back packets were infeasible, as the wireless interfaces usually got stuck in the process.

10 For the actual maximum WLAN throughput we should add the UDP/IP overhead transmitted, for an aggregate rate of 229 KBps (1.83 Mbps), without including MAC layer overhead.
throughput achievable for our WLAN under the Linux OS, i.e. performance peaks when the slower host is sending and the additional processing power of the faster host does not add anything. When the faster host is sending, throughput is actually slightly lower, due to the packet loss shown earlier.\textsuperscript{11}

A more interesting observation that can be made with respect to Figure 4 is that TCP throughput is nowhere near its UDP counterpart, and it actually drops when the packet size is increased from 500 to 1000 bytes. For the 100 and 500 packet sizes throughput is higher when the slower host is sending, since there are no losses to cause TCP to retransmit complete packet windows (see Figure 2). For the 1000 and 1500 byte packets however, throughput is practically identical in both directions even though there are no errors in one direction while there are some errors in the other. At the largest possible packet size, TCP throughput is only about 30\% of the UDP throughput, and this cannot be explained solely by the higher TCP overhead or by the reverse TCP traffic due to acknowledgments, and certainly not by the loss rate.

The explanation becomes clear by looking at Figure 5, where we can see the mean, minimum and maximum number of data and acknowledgment packets sent and received on the direction from IOS to MYKONOS across the 5 repetitions of the tests. The gaps between the sent and received curves for both data and acknowledgment packets show that there is considerable loss taking place on the link, growing with packet size, and since these gaps are of the same magnitude, these losses must be due to collisions that go undetected at the CSMA/CA MAC layer. The direction from MYKONOS to IOS is nearly identical, as seen at Figure 6, explaining the similar throughput results. Undetected collisions cause data loss that has to be recovered from by retransmissions, and these are clear in the 1500 byte packet tests, since the total number of data packets sent is much more than 10,000. At the same time, the phenomenon of sending packets with larger sizes than expected, as we mentioned earlier, is clear for the 1000 byte tests, where the total number of packets received is less than 10,000. The metric reported as collisions by the hardware (also plotted on Figures 5 and 6) is not the number of real collisions taking place on the medium: the CSMA/CA MAC layer cannot detect collisions, so this metric is some other quantity measured by the hardware, which is large at the receiver and

\textsuperscript{11}Throughput is measured at the receiver, so that lost packets are not counted.
Figure 4: Scenario 1, UDP and TCP throughput

Figure 5: Scenario 1, TCP data, acknowledgments and collisions, from IOS to MYKONOS
nearly zero at the sender on all TCP tests using two ISA cards.

Careful study of the `tcpdump` traces for these tests verifies that undetected collisions are a major problem, since when data and acknowledgment packets collide some data packets after the lost one are also retransmitted, even if they had not been lost.\(^{12}\) When some of the duplicate acknowledgments that TCP sends to initiate fast retransmission [9] are lost due to collisions, the sender will exhaust its transmission window waiting for three duplicate acknowledgments, and then it will wait until the next timeout before restarting at the lost packet, leaving the medium unused in the meantime. This can be seen in Figure 7 and 8 which show a packet trace produced by `tcpdump` for the first 0.6 seconds of one of the 1500 byte packet TCP tests from MYKONOS to IOS. Figure 7 shows data and acknowledgments as seen by the data sender (MYKONOS), while Figure 8 shows the same data from the viewpoint of the data receiver (IOS). Due to collisions, some data packets never arrive at IOS, and, conversely, some acknowledgments never arrive at MYKONOS. At time 0.05 three data packets with consecutive sequence numbers are sent (see Figure 7), but the third one is lost (does not appear in Figure 8). This packet has collided with the acknowledgment for the first of these three data packets (the acknowledgment appears on Figure 8 but not on Figure 7). Unaware of the collision, the sender transmits four more packets right before time 0.1, and the receiver replies with three duplicate acknowledgments for the last received data packet in sequence, a signal that loss has occurred, but one of them is lost due to another collision with the third of the four new data packets (see again the differences between Figures 7 and 8). Since only two duplicate acknowledgments have been received, the sender does not go into recovery mode, and since it has exhausted its window, it has to wait for the next timeout.

Between the first transmission of the lost packet at time 0.05 and its retransmission at time 0.25, 200 ms elapse, during most of which the channel is idle. Linux uses a 10 ms granularity clock (as opposed to 500 ms of many BSD derivatives), with a 200 ms lower limit for the TCP timeout value, and this is indeed the timeout interval for the WLAN as evidenced by the trace. After the timeout, the host goes in slow start mode [9], so it has to reopen its congestion window, as shown in Figure 7 after time 0.25. A bit later, at around time 0.4, another collision occurs between data

\(^{12}\)This is largely avoided by the proposed TCP variants that use the selective acknowledgment (SACK) option [6].
Figure 7: Scenario 1, TCP trace snapshot for a 1500 byte packet transfer from MYKONOS to IOS (sender’s view)

Figure 8: Scenario 1, TCP trace snapshot for a 1500 byte packet transfer from MYKONOS to IOS (receiver’s view)
and acknowledgment packets, but this time only two duplicate acknowledgments are transmitted anyway, causing the sender to stall until the next timeout occurs. This is a case where frequent losses cause the **ACK clock** property of TCP to be lost, i.e. the sender exhausts its window before transmitting enough data packets after a loss to trigger three duplicate acknowledgments (see [6] for more details). The periods of inactivity between timeouts are the main reason why TCP throughput is so much lower than UDP throughput, and we should note that the situation would be much worse with 500 ms granularity timers as the timeouts would be longer and more time would be wasted. It is evident then that when two ISA cards are communicating, the CSMA/CA scheme performs in a quite unsatisfactory manner, and is not able to avoid enough collisions to keep system throughput at acceptable levels.

Finally, Figure 9 shows the signal level distribution for the data packets received during 1500 byte packet TCP tests in both directions, while Figure 10 shows the corresponding noise level distribution. We have combined the measurements from all 5 repetitions of these tests into one histogram, so that each curve represents 50,000 received data packets plus their retransmissions. In all scenarios, these distributions were nearly identical across all packet sizes and both protocols (TCP and UDP) for each scenario, so we chose 1500 byte TCP tests arbitrarily. We show the exact same portion of signal and noise levels for all scenarios for comparison purposes.\(^\text{13}\) For this scenario the measurements are nearly symmetric, which is quite reasonable given that both interfaces were exactly the same. Since the two hosts were placed next to each other, these distributions are the optimal cases for both signal and noise levels for our environment. Note that signal and noise metrics use the same units, which are defined by the hardware.

### 6.3 Scenario 2

This scenario combines an ISA and a PCMCIA card, with the hosts placed next to each other as in the previous case. This test establishes a baseline for the mixed interface case and since the processing power of SYROS and MYKONOS is similar, comparisons between the two baseline scenarios are direct. Figure 11 shows the loss rate for UDP tests,

\(^{13}\)As mentioned earlier, the signal quality metrics were nearly identical across all scenarios.
revealing that the loss from the faster ISA host to the slower PCMCIA host is similar to scenario 1, only a bit higher with 1500 byte packets (up to 0.8%). In the reverse direction though, unlike in the ISA to ISA case, there is loss from the slower to the faster host (up to 0.2%). A close examination of the tcpdump trace files for UDP reveals that the lost packets never really leave the sending interface, in a manner similar to the UDP packet dropping phenomenon with 100 byte packets, discussed earlier, but in a smaller scale. Since the interface controller of the PCMCIA cards only has a single transmit buffer (in contrast to the ISA card controller that uses multiple buffers), it seems that these losses could be attributed to sender buffer overruns, that do not occur in ISA cards which do not suffer from the same limitation. Loss variance is lower than scenario 1 but still quite significant, and could be the reason why the loss rate in the PCMCIA to ISA direction seems to drop slightly with growing packet sizes.

Although the loss distribution in the IOS to SYROS case is uninteresting, with all loss runs consisting of a single packet, in the reverse direction we have loss runs of 1–5 packets, as seen in Figure 12. Again similarly to scenario 1, the received packet run distribution shown in Figure 13 shows that when the faster host is sending, losses occur at least every 1000 packets, hinting to periodic overwhelming of the receiver. In contrast, when the slower host is sending packet losses occur less frequently, but last for more packets (see Figure 12).

Figure 14 shows TCP and UDP throughput statistics in both directions. As in scenario 1, throughput from the faster to the slower host is higher for UDP than for TCP, due to the losses that make TCP retransmit packets occasionally (and of course the added load of reverse traffic). When the PCMCIA host is sending though, TCP throughput is lower even though loss is lower, which verifies that the PCMCIA cards can only achieve lower throughputs than their ISA counterparts [8]. This performance gap is evident in Figure 15 where the sequence numbers when the ISA host is transmitting increase at a steeper slope than when the PCMCIA host is the sender, even though losses on the ISA to PCMCIA direction cause occasional retransmissions of lost and previously transmitted packets. When the PCMCIA host is the sender, packets are sent in bursts with very short gaps between them. These gaps are probably caused by the interface controller stalling due to buffer shortages, as explained above. Until the sender restarts transmission, some time is lost, further contributing to the degraded performance of PCMCIA cards.
Figure 11: Scenario 2, UDP packet loss

Figure 12: Scenario 2, lost UDP packet runs (1500 byte packets)
Figure 13: Scenario 2, received UDP packet runs (1500 byte packets)

Figure 14: Scenario 2, UDP and TCP throughput
Another reason for the lag in performance is the longer interframe gap in the PCMCIA card (96 bits) compared to the ISA card (32 bits) which causes the PCMCIA cards to pause for a longer time between sending consecutive frames. Note that these numbers are used in the Linux drivers where the contention window is 512 bits for both cards. In the BSD derived system driver described in [8] the same interframe gap (32 bits) is used for both cards, but the PCMCIA driver uses a larger initial contention window (512 instead of 384 bits for the ISA cards), which again makes the PCMCIA card pause longer between sending frames. The less aggressive timing of the PCMCIA cards is evident in Figure 16 where a part of a UDP transfer (where no contention occurs) is shown with one point for each packet sent (the sequence numbers used for loss detection are ignored in the figure). The PCMCIA sender clearly leaves a slightly larger gap between transmissions, causing the lag in throughput between ISA and PCMCIA hosts.

A more intriguing problem is why UDP is slower than TCP in the PCMCIA to ISA direction as seen in Figure 14. Part of this discrepancy is probably due to measurement fluctuations (note the high variance of the UDP results). Another reason why UDP is slower than TCP in this direction, despite the loss induced TCP retransmissions (see Figure 11), is the aggregation of data segments into larger packets which only takes place for TCP transmissions. This is implied by the nearly identical data rates between UDP and TCP for the 1500 byte packet case where no aggregation is possible. The main cause for the very low UDP performance however is seen in Figure 16: occasionally the PCMCIA sender pauses for more than one second, leaving the medium unused, despite the complete absence of contention in UDP tests. Examining the tcpdump traces for UDP tests we discover that the packet losses mentioned above for this direction occur exactly at the beginning of these long pauses. It is very likely then that the sender buffer limitations that stall TCP may turn into real buffer overruns, or at least chronic buffer shortages, with UDP, causing the interface to reset itself after dropping a few packets, which would explain the long pauses observed in Figure 16. While flow control prevents this problem from becoming too severe with TCP, it does not apply to UDP traffic, resulting in very degraded performance. This phenomenon causes the peak UDP throughput for this baseline scenario to be only 160 KBps (1.28 Mbps) in the PCMCIA to ISA direction, compared to the 223 KBps (1.78 Mbps) of the reverse direction.

In Figure 17 it is clear that on the ISA to PCMCIA direction the discrepancy between sent and received packet
Figure 16: Scenario 2, UDP data (1500 byte packets)

Figure 17: Scenario 2, TCP data, acknowledgments and collisions, from IOS to SYROS
metrics among the transmitter and receiver is minimal, meaning that only a small amount of packets are lost on the link, unlike in scenario 1. This is the reason why TCP performance is much better than that of scenario 1 with its frequent collisions and subsequent timeouts and retransmissions. Apparently the slightly different implementations and timing of the two interfaces eliminate the synchronization phenomenon observed when two ISA cards are communicating, which causes TCP performance to collapse due to the large number of undetected collisions. In the reverse direction, shown in Figure 18 the situation is very similar, with even less discrepancies between sender and receiver. In both directions, the aggregation of TCP segments into larger packets is quite pronounced for the 1000 byte packet tests. The metric reported as collisions by the hardware rises with packet size, but in this scenario it is always the ISA host (whether sender or receiver) that reports large metrics while the PCMCIA host reports zeroes. This is probably related to the differences between the interfaces, but as it is unclear exactly what this metric represents, we cannot draw any conclusions from these results.

Finally, the signal level distribution shown in Figure 19 and its noise level counterpart shown in Figure 20 are quite asymmetric. While the signal level from ISA to PCMCIA is similar to the ISA to ISA case of scenario 1, in the PCMCIA to ISA direction it is lower. Similarly, the noise level detected at the ISA receiver is similar to scenario 1, while at the PCMCIA receiver it is higher. This could indicate that the PCMCIA cards (and their antennas) emit weaker signals and are less tolerant to noise than the ISA cards. Alternatively, it could be explained as an artifact of the slightly different implementations of the two interfaces, since the antenna modules used by the two cards are quite different. In any case, these histograms establish a baseline for all other mixed ISA and PCMCIA host tests.

### Scenario 3

For this set of experiments, the two ISA hosts used for scenario 1 are placed 45 feet apart, with two intervening walls (see Figure 1). The loss rates shown in Figure 21 are similar to those for scenario 1, in that there is nearly no loss when the slower host is sending, while low loss (up to 0.9% for 1500 byte packets) occurs in the reverse direction. Although the loss rate for 500 byte packets is lower than that of scenario 1, this is probably due to the high variance.
Figure 19: Scenario 2, signal level histogram (TCP, 1500 bytes)

Figure 20: Scenario 2, noise level histogram (TCP, 1500 bytes)
of the loss rates seen in those experiments. The lost and received run distributions are also nearly identical to scenario 1, i.e. nearly always one packet is lost every less than 1000 packets, and the explanation is again that the faster sender overruns the slower receiver.

Throughput, shown in Figure 22, is also nearly identical to scenario 1 in both directions, including the abysmal TCP performance compared to UDP. The explanation is a nearly identical gap between sent and received packets caused by a high number of collisions that are never detected by the sender, leading to timeouts and retransmissions. We do not present the corresponding graphs as they are practically identical to those for scenario 1. Examination of the tcpdump traces also reveals similar behavior to scenario 1.

The main difference between this scenario and scenario 1 is shown in Figure 23. Even though the signal level distribution shown is symmetric, the signal levels are much lower than in scenario 1, due to the larger distance and intervening obstacles. However, the noise distribution shown in Figure 24 is the same as in scenario 1, verifying the absence of any significant sources of external noise in our testing environment. It seems then that even though the signal level drops with increasing distance, packets can still be captured as long as the signal level is above a threshold and noise is not significant. In this case, the signal and noise distributions are not overlapped at all (note that both levels are expressed in the same, hardware dependent, units), thus there exists an adequate margin to avoid losses due to excessive noise. As a result, despite the signal level deterioration from scenario 1 to scenario 3, the results are practically the same in all other respects.

6.5 Scenario 4

Scenario 4 differs from scenario 2 in the same way that scenario 3 differs from scenario 1, i.e. we use the same hosts (one ISA and one PCMCIA) but at a distance of 45 feet. Although there seems to be some variation in the loss rates for UDP, shown in Figure 25, it is probably an artifact of the large variances seen on both scenarios, and therefore the results can be explained as statistical fluctuations. Loss and reception metrics for run sizes and the corresponding CDF curves are also nearly identical, so they are not repeated here.
Figure 22: Scenario 3, UDP and TCP throughput

Figure 23: Scenario 3, signal level histogram (TCP, 1500 bytes)
Figure 24: Scenario 3, noise level histogram (TCP, 1500 bytes)

Figure 25: Scenario 4, UDP packet loss
Regarding throughput, the results are again similar to scenario 2, as can be seen from Figure 26. The only difference is that TCP results are here closer to each other in both directions, and nearly equal to the UDP throughput in the PCMCIA to ISA direction. The low variance seen on the results may indicate that they are closer to the true averages than the large variance results shown for scenario 2. Transmission and reception statistics are again similar to scenario 2, with the misleadingly named collision metric high only at the ISA host in both cases. Thus, the same discussion presented for scenario 2 is valid for scenario 4 in these areas.

Finally, the signal (Figure 27) and noise (Figure 28) distributions are similar to those of scenario 4, including the asymmetry between ISA and PC cards, but the signal levels are uniformly lower, while the noise levels are essentially the same. This is the same behavior seen when comparing scenarios 1 and 3. The asymmetry is again most likely due to different interface and antenna implementations, while the changes in signal levels at identical noise levels, despite a virtually identical performance in all other respects, verify our observations for scenario 3, namely that the distance and obstacles for these scenarios reduce signal strength but do not significantly obstruct communication.

### 6.6 Scenario 5

In this set of experiments we moved the PCMCIA host 45 feet away from the slower of the ISA hosts (MYKONOS). This is the first scenario where both hosts have comparable processing power, although their wireless interfaces differ. Looking at the loss rates in Figure 29, we can see that loss on the PCMCIA to ISA direction is very similar to scenarios 2 and 4 for this direction. The drop in average loss rates is most likely statistically insignificant given the very large variability of the results. In the ISA to PCMCIA direction however the loss is practically zero. This shows that when the peers are matched in processing power loss due to receiver overruns is avoided, while the loss remaining on the PCMCIA to ISA direction is probably due to sender limitations, as discussed in scenario 2. Similar to scenarios 2 and 4 for the PCMCIA to ISA direction, the distribution of loss runs (Figure 30) ranges between 1 and 5 packets, while the distribution of received packet runs (Figure 31) shows that losses are not periodic, as when a fast sender overruns the receiver. These figures do not show the corresponding distributions for the ISA to PCMCIA direction since the loss
Figure 27: Scenario 4, signal level histogram (TCP, 1500 bytes)

Figure 28: Scenario 4, noise level histogram (TCP, 1500 bytes)
there is too low to make any impact.

Regarding throughput, Figure 32 shows ISA to PCMCIA throughput to be higher than in the reverse direction. Since no loss occurs on this direction, unlike in previous scenarios when the sender used to overrun the receiver, UDP and TCP throughput are very close, and indeed the TCP data rate of 213 KBps (1.7 Mbps) is the highest we saw during our tests. UDP throughput on the same direction is nearly the same as the peak throughput measured when the faster sender was transmitting, confirming our earlier observation that MYKONOS is fast enough to achieve peak throughput results from the ISA cards. By not being too fast, it avoids losses due to receiver overruns. In the direction from PCMCIA to ISA, TCP performance drops due to the less aggressive timing of the PCMCIA cards discussed before and also because of the (small) packet loss rate. As before, UDP lags behind TCP in throughput when the PCMCIA host is sending, due to the reasons that we discussed in Scenario 2. The actual throughput numbers are similar to those of scenarios 2 and 4.

Figure 33 and Figure 34 confirm the observations made for scenarios 2 and 4, with nearly identical curves for lost and received packets, showing that practically no undetected collisions occur on the medium, thus maximizing performance. Note that since there are no receiver overruns and the loss rate is practically zero in the ISA to PCMCIA case, the data and acknowledgment curves in Figure 33 completely overlap each other, in contrast to the small gaps seen in scenario 2 where some data packets where lost. As before, the ISA host reports a large number for the (misnamed) collision metric, regardless of the direction of data transfer. The figures also show that data segments are occasionally aggregated into larger packets, especially in the 1000 byte packet tests, as before.

Finally, the signal (Figure 35) and noise (Figure 36) level histograms are very similar to those of scenario 4, especially in the noise levels measured. The signal levels are slightly lower in both directions, due to variations in intervening obstacles from one scenario to the other (the distance is roughly the same). The usual asymmetry between ISA and PCMCIA reported statistics is also apparent in these figures, again most likely due to antenna and interface implementation differences.
Figure 30: Scenario 5, lost UDP packet runs (1500 byte packets)

Figure 31: Scenario 5, received UDP packet runs (1500 byte packets)
Figure 32: Scenario 5, UDP and TCP throughput

Figure 33: Scenario 5, TCP data, acknowledgments and collisions, from MYKONOS to SYROS
Figure 34: Scenario 5, TCP data, acknowledgments and collisions, from SYROS to MYKONOS

Figure 35: Scenario 5, signal level histogram (TCP, 1500 bytes)
6.7 Scenario 6

Scenario 6 is again a variation on scenario 4, only the distance between the ISA and PCMCIA hosts is about 60 feet and more obstacles are placed between the peers. Unlike scenario 5 (and like scenario 4), the ISA host is faster than the PCMCIA host. Figure 37 shows that packet loss is quite similar to scenario 4, only slightly higher in some cases, although the high variance of the results cannot make us very confident about their accuracy. This variance may also explain the peculiar behavior of decreasing losses with increasing packet size. Still, loss rate is higher on the ISA to PCMCIA direction, presumably due to the fast sender overrunning the receiver, as in previous scenarios. The loss run distributions are also similar to previous ones, with ISA to PCMCIA runs of 1 packet, and runs of 1–5 packets in the reverse direction. However the distribution of received packet runs is more symmetric between the two directions (Figure 38), although again the ISA to PCMCIA direction suffers from more frequent interruptions.

The throughput results shown in Figure 39 are similarly commonplace for the PCMCIA to ISA direction, i.e. similar TCP and UDP performance, with UDP slightly lower. However, in the ISA to PCMCIA direction, although UDP does much better than TCP. TCP has a sudden drop in performance in the 1500 byte packet case. Since the sent and received packet curves (not shown) are nearly identical to scenario 4, i.e. the curves are overlapping thus no undetected collisions occur, and the packet loss rate is actually lower for the longer packets (see Figure 37), the only explanation is that this is a statistical fluctuation (note the high variance of the results for this metric, and the correspondingly high variances of the loss rates). We do not show the signal and noise level histograms as they are essentially the same as in scenario 5, slightly asymmetric between ISA and PCMCIA, with low noise as usual and signal levels appropriately scaled for the increased distance between the hosts.
Figure 37: Scenario 6, UDP packet loss

Figure 38: Scenario 6, received UDP packet runs (1500 byte packets)
7 Discussion

There are many conclusions to be drawn from the results presented above, including answers to the questions posed as research goals in Section 2. By design, our tests did not stress the system beyond its range, and despite the distance and obstacles between communicating hosts the 2.4 GHz DSSS cards performed admirably in our environment. By using faster hosts than previous researchers [4, 5], we found what seems to be the saturation point of the WLAN, achieving peak data throughputs of 1.8 Mbps between ISA cards, a slightly lower 1.78 Mbps from ISA to PCMCIA, and a much lower 1.28 Mbps from PCMCIA to ISA, which verifies that ISA cards due to more aggressive timing (either due to shorter interframe gaps or due to shorter initial contention windows) send data faster than their PCMCIA counterparts [8]. These throughputs hold for unidirectional UDP tests with no contention for reverse traffic, and when scaled to include UDP/IP and MAC layer overhead, the ISA rates approach the nominal throughput of the system which is 2 Mbps. Although the observed throughput follows the trend in the reported results [4, 5, 8] of increasing data rates with increased host processing power, the high throughputs observed compared to earlier reports could also be partly explained by the newer version of the WLAN used (2.4 GHz instead of the older 900 MHz variant) and the different OS (Linux instead of BSD). Since the data rate achieved by both ISA based hosts is roughly the same, despite their different processing abilities, we also conclude that a 166 MHz Pentium system is powerful enough to saturate the wireless channel.

A more interesting result is that faster hosts have their drawbacks too. First of all, it is impossible to send short back to back packets in UDP mode without losing most of them somewhere in the protocol stack, a problem that also appears at a smaller scale in 10 Mbps wired LANs. We actually encountered this problem with all packet sizes less than about 500 bytes, not just on the 100 byte packets exhaustively studied during our experiments. As long as the sender is throttled back by a flow control mechanism though (as in the TCP case) no such problems arise. Since an application mix that would send huge amounts of very small packets back to back is rather unlikely, this peculiarity should not be a cause for concern.

The really problematic issue with fast sender hosts is their potential for overrunning slower receivers, causing
occasional loss of a single packet before the receiver can catch up again with the sender. This problem arose when a fast ISA based host was sending to a slower receiver that used either ISA or PCMCIA cards. It should not be a problem for PCMCIA senders, since they do not saturate the link, leaving a safe margin for the receiver. Note that PCMCIA receivers are quick enough to catch up with the fast ISA senders, provided that the processing power mismatch is small. This loss is a problem for TCP as it causes frequent retransmissions and invocation of congestion control procedures that reduce throughput. Since loss occurs regardless of packet size (and is higher for longer packets) it would be a significant problem for bulk data transfers that employ large packets and attempt to saturate the link. We considered introducing very short pauses between packet transmissions to avoid these overruns, but due to the 10 ms minimum system timer granularity in our system, the pauses were long enough to reduce throughput more than the losses.

Besides throughput differences that are explained by the larger gaps between packets with PCMCIA senders, we noticed a few other asymmetries between ISA and PCMCIA cards. The difference in measured signal levels (PCMCIA cards recorded lower levels) was probably due to the differences in card and antenna implementation, but it could also mean that the PCMCIA cards are effective over a smaller range. A more important distinction is that with a PCMCIA sender the ISA receiver would always see losses of a few packets occasionally, that were actually never transmitted by the sender. Obviously, since this is a sender related problem, it would also occur with PCMCIA receivers. The problem is most likely due to PCMCIA sender buffer limitations that cause packets to be dropped before transmission when the interface controller is overloaded. In contrast, in the ISA to PCMCIA direction the only loss was due to receiver overruns (single packets), since when an ISA sender with similar processing power to the PCMCIA receiver was used, there was practically no loss despite the more aggressive timing of the ISA cards. ISA cards do not suffer from the sender buffer constraints of PCMCIA cards, and thus do not present this packet dropping problem.

In addition, during TCP transfers the PCMCIA sender would momentarily pause every few packets, probably due to the buffer limitations at the interface controller, also contributing to the gap in performance between ISA and PCMCIA senders. The most peculiar phenomenon was that during UDP transfers the PCMCIA sender would sometimes stall for up to one second, causing UDP to lag behind TCP in terms of throughput. This is exactly the opposite of what usually happens, i.e. normally TCP is slower due to larger header and processing overhead, and the need to recover from losses by retransmitting packets. These long pauses could be due to the interface controller resetting the sender under excessive load, caused by the same buffer limitations that lead to pauses during TCP transfers. While TCP is limited by its flow control mechanisms in the load that it can offer to the interface, UDP has no such restraints, and thus could easily overwhelm the interface controller leading to lengthy reset actions that dramatically affect UDP performance.

It is interesting to compare CSMA/CA to CSMA/CD, in particular how effective it is in avoiding collisions that cannot be detected on the wireless link so as to abort wasted transmissions early on. We extended the previously reported research by performing TCP tests, where traffic is inherently bidirectional, exactly to see how well CSMA/CA performs under contention. Although the WLAN interfaces report a collisions metric, it is unclear what it signifies exactly, and its peculiar behavior (high on the receiver and zero on the sender for ISA to ISA tests, and high on the ISA host regardless of direction on mixed ISA and PCMCIA tests) does not help us to draw any conclusions related to its meaning. The real (and undetected) collision rate can instead be seen by comparing the number of sent and received data and acknowledgment packets seen at either side of each transfer. It turns out that collisions are a significant problem in the ISA to ISA case, dramatically decreasing the throughput of TCP (to 30% of UDP throughput), despite a low to nonexistent loss rate. Loss of duplicate acknowledgments or loss of the ACK clock properties of TCP [6] result in frequent invocation of timeout initiated recovery, where the time between the sender’s windows closing and the timeout occurring is wasted, hence the low TCP throughput. This large number of collisions is a sign that ISA hosts due to their identical and aggressive timing parameters are quite prone to synchronization when scheduling transmissions.

In contrast, when an ISA host communicates with a PCMCIA host, collisions are practically nonexistent in both directions, so that TCP throughput is maximized, being limited only by other losses and the timing parameters of the sender, i.e. the PCMCIA senders are always slower than their ISA counterparts. We conjecture that that CSMA/CA could also work well between two PCMCIA hosts, due to their less aggressive sending behavior (compared to ISA hosts) that does not saturate the link, leaving more room for acknowledgments to be sent. In conclusion then, CSMA/CA works well when the sender and receiver can avoid collisions due to timing differences (the mixed PCMCIA and ISA case). It may also work well when the sender cannot saturate the link. It works less than admirably when both hosts are aggressive and synchronized (ISA to ISA case), leading to the paradox of the faster ISA cards exhibiting much lower TCP throughput than the PCMCIA implementations. One possible approach to remedy those problems would
be to relax the timing of the ISA hosts by increasing their initial contention windows. This would reduce peak UDP performance, but it could avoid the dramatic drop in TCP throughput.

It is important to notice that undetected collisions are a very significant problem for TCP, not just because they waste time due to already garbled packets being fully transmitted, but mainly because they lead to retransmissions and even timeout based recovery. At the TCP level this problem could be reduced by smarter retransmission policies such as those employing the SACK option, which have been shown to be more robust than standard TCP mechanisms under frequent losses [6]. A more general solution for the CSMA/CA problem of undetected collisions is MAC layer acknowledgments (as proposed by the IEEE 802.11 standard [2]). Since these are transmitted without contention (their transmission starts during the interframe gap while other senders are blocked from seizing the channel) they can potentially improve throughput for TCP and other bidirectional protocols, as they provide explicit collision detection. Even though collisions would still occur wasting bandwidth for the duration of the garbled packets, TCP throughput is mainly affected by recovery time. Combined with the MAC layer retransmissions proposed in the standard, these mechanisms can repair many losses without intervention by TCP (and in particular, before timeouts occur). Long chains of retransmissions of course can cause TCP to time out anyway and retransmit in duplicate the same data as the MAC layer [3], and this is more likely in short paths with low round trip delay (one hop WLANs being such a case). MAC layer acknowledgments and retransmissions can also be a waste of time for protocols and applications that would prefer sending new packets to retransmitting old packets, such as real time audio over UDP. This argues for limited and/or protocol dependent retransmission policies, which are best left to software layers, such as LLC (Logical Link Control), rather than the embedded MAC layer hardware.

Regarding modeling, even though our results can be used to provide more input for two state link models for unidirectional (UDP) traffic (along the lines proposed in [8]), it was mostly implementation peculiarities rather than the bit error rate of the link that caused losses. Receiver overruns in the ISA sender case and clustered losses in the PCMCIA sender case can however easily be included in a two state model (after all, it is the error distribution that matters rather than their exact source). Even though some of the packet run distributions were not very smooth, this is a problem that can be solved by measuring longer packet runs so as to approximate the long term behavior of the system more closely. Realistic transfers of course would only see this behavior over shorter terms, as in our 10,000 packet tests, i.e. even if the system has a steady state for its loss behavior, most sessions would not experience it anyway. One complicating factor for these models is that they are dependent on both the combination of interface types used (ISA or PCMCIA) and the relative speeds of the simulated hosts, making them hard to generalize. This means that many simulations with different parameters would need to be ran to comprehensively test a protocol improvement over the WLAN model.

What is rather questionable is whether such a model is adequate for capturing bidirectional traffic, given the problems we encountered during the ISA to ISA TCP tests, especially in view of the fact that TCP traffic is so prevalent on the Internet today. In addition, when multiple users and sessions share the link, there is more contention and potential for collisions, which was the dominant problem in these tests, diminishing the effects of other losses. Accurate modeling requires that we capture not only the loss distribution, but also the correlation between the losses on both directions (especially useful for TCP when duplicate acknowledgments collide with data or many data packets are lost close to each other). This leads us to believe that to get good results the actual CSMA/CA protocol, with its limitations, must also be simulated besides the residual error behavior of the link, as this seems to be the most important factor determining transport layer performance in most of our tests. It is furthermore very inaccurate to employ CSMA/CD MAC layer simulators for that purpose by simply increasing the penalty for undetected collisions (to compensate for not aborting garbled transmissions). It is the effects of the loss on higher layers rather than the MAC layer loss penalty that counts, thus the fact that CSMA/CD collisions are not losses inherently differentiates them from CSMA/CA collisions.

A final point of note is the effect of timers with fine granularity for TCP. Linux uses 10 ms granularity timers, as opposed to the 500 ms of many BSD derived systems, which can potentially cause problems because they are too tight, although various measures are taken to avoid making TCP timeouts unstable (such as having a 200 ms lower limit for timeouts). For example, when using a LAN, the fine granularity of the timers can lead to very tight estimates of the round trip time and its variance, so that timeouts can occur more often than desired, when delay is caused by, say, MAC layer contention, or, more possibly, the variable delays of longer paths. In our WLAN tests however, the TCP traces show that when duplicate acknowledgments or multiple data packets in a window are lost due to excessive collisions
in the ISA to ISA tests and the transmission window is closed, by having timeouts after only 200 ms the idle period between transmissions is minimized, and TCP throughput remains at a decent level. It is very likely that with coarser 500 ms timers the timeouts would be too large, and more time would be spent waiting for a timeout than transmitting data, thereby reducing TCP throughput to abysmal numbers. For our experiments, the finer timer granularity was extremely beneficial, but it is an open question whether this would be the case for longer paths [7] or when MAC layer acknowledgments and retransmissions (that increase delay variability) are used as in the IEEE 802.11 proposed WLAN standard.

8 Conclusions

The measurements reported here extend previous work in many directions, including bidirectional traffic, different implementations of hosts, interfaces and OS, and heterogeneous systems. Our analysis has uncovered many interesting points and issues for further research, that are summarized below:

- Achievable UDP data rates reach 1.8 Mbps for ISA senders and 1.28 Mbps for PCMCIA senders.
- Host processing power influences throughput, with a 166 MHz Pentium based system having adequate processing power to get peak throughput.
- Fast senders can overwhelm slower receivers leading to periodic packet loss.
- When two ISA hosts are used there is a potential for a lot of undetected collisions in TCP mode, leading to degraded performance.
- PCMCIA hosts pause every few packets in TCP mode due to sender buffer limitations in the PCMCIA cards, regardless of the receiver.
- Under UDP PCMCIA senders suffer from buffer overruns that can lead to lengthy interface resets and packet losses, greatly reducing performance.
- Although TCP is influenced by all types of losses, frequent collisions are the worst as they may lead to timeout initiated recovery.
- CSMA/CA performs well with bidirectional traffic as long as the hosts are not synchronized.
- Less aggressive MAC layer timing or more robust TCP recovery mechanisms could be beneficial over CSMA/CA.
- Limited MAC layer acknowledgments and retransmissions could be beneficial for TCP but problematic for other protocols.
- UDP loss models can be formulated for simulations, but they are dependent on the mix of interfaces and hosts.
- TCP simulation models should take into account CSMA/CA behavior, and in particular collisions, in order to reproduce accurately a WLAN.
- Fine granularity timers speed up timeout initiated recovery considerably, but they may be too sensitive if MAC layer retransmissions are employed.

References


