TCP and UDP Performance over a Wireless LAN

George Xylomenos and George C. Polyzos
{xgeorge,polyzos}@cs.ucsd.edu
Center for Wireless Communications & Computer Systems Laboratory
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, California, 92093-0114, U.S.A.

Abstract— **We present a comprehensive set of measurements of a 2.4 GHz DSSS wireless LAN and analyze its behavior. We examine issues such as host and interface heterogeneity, bidirectional (TCP) traffic and error modeling, that have not been previously analyzed. We uncover multiple problems with TCP and UDP performance in this system. We investigate the causes of these problems (radio hardware, device drivers, network protocols) and discuss the effectiveness of proposed improvements.**

I. INTRODUCTION

Wireless communications are experiencing explosive market growth in areas such as *cellular telephony*, *satellite communications* and *wireless LANs*. Wireless LANs (WLANs) support high speed networking over small areas that may be hard to wire conventionally. Numerous vendors are offering WLAN systems at dropping prices, while the new IEEE 802.11 standard [1] will eventually enable product interoperability. Typically, WLANs emulate a wired LAN (e.g. Ethernet), which makes them easy to connect with the Internet. However, their lower bandwidth and higher loss rate makes their presence felt. Even moderate packet loss due to wireless errors has severe effects on Internet protocols such as TCP [2]. In order to ameliorate these performance problems we need a clearer understanding of WLAN behavior, therefore measuring and analyzing the performance of systems under realistic conditions is an important task.

To this end, we present here a comprehensive set of measurements of a WLAN and analyze its behavior, extending previous results in many ways. In Section II we outline our measurement goals to provide a basis for test design. Section III details our experimental setup, while Section IV explains the rationale behind individual tests and the test suite, as well as the data gathered during and after testing. These data are used in Section V to describe the performance of both unidirectional (UDP) and bidirectional (TCP) communications. We review these results and discuss their implications in Section VI. We conclude with a summary of our findings in Section VII.

II. MEASUREMENT GOALS

Our aim was to compile a comprehensive set of data describing the performance of a WaveLAN [3] system in terms of throughput and loss under various *realistic* conditions. In order to find solutions for performance problems, we first need to locate their root causes. This is quite difficult since perceived network performance is influenced by network *and* host processing hardware, interface device drivers and network protocol implementation in the OS. By varying these parameters during experimentation it is easier to identify which aspect of the system should be modified to improve performance, either in existing or future designs. Our work thus aims to extend published results [4], [5], [6] in many ways.

- *System Heterogeneity:* We used hosts with varying processing power and different wireless interface implementations. Previous work kept one of these parameters fixed.
- *New Implementations:* Published results described the 900 MHz systems while we examined the improved 2.4 GHz version. We also used hosts with faster processors that could potentially achieve higher throughputs.
- *Bidirectional Communications:* We measured the performance of TCP, in addition to (previously examined) UDP. Bidirectional traffic in the form of TCP data and acknowledgments reduces throughput and introduces collisions.
- *Error Modeling:* Previous measurements were used to define wireless error models [6]. We present additional measurements and also analyze bidirectional traffic effects.
- *Operating System:* We employed the Linux OS instead of the BSD UNIX derivatives used in previous work. A comparison among these results provides insight on the effects of device driver and network protocol implementations.

Regarding measurement limitations, we tested single hop paths only, even though TCP has been shown to perform differently over longer paths [2], so as to maintain complete control over the path. We did not study the effects of mobility, since the form factor and range of our WLAN makes it unsuitable for operation on the move. Delay was ignored, as it is too short on high speed WLANs to significantly affect performance. We did not measure effective range [4] or behavior under interference [5], focusing instead on normal office conditions.

III. EXPERIMENTAL SETUP

A. Hardware

We employed three hosts for our experiments, with two of them active in the WLAN during each test. A monitoring utility verified that no other WLANs were operating nearby. The hosts were also connected to each other and the Internet via an Ethernet that was used to control the tests. The WLAN used was the Digital RoamAbout 2.4 GHz DSSS system, an OEM version of the Lucent WaveLAN [3], also available in 900 MHz DSSS and 2.4 GHz FHSS versions. These interfaces are available as ISA (desktop) and PCMCIA (laptop) cards, which slightly differ in their Ethernet controllers and radio module/antenna packages. They both implement a *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) MAC scheme and are fully compatible over the air. Both interfaces support a nominal bandwidth of 2 Mbps, same as the other WaveLAN versions.

Due to the difficulty of detecting collisions during transmission on this system [3], instead of aborting garbled transmissions as in CSMA/CD, the medium remains used, and wasted, until all collided packets end. To avoid this, CSMA/CA is more

TABLE I

DESCRIPTION OF HOSTS USED

Name	Processor	RAM	Interface Type
IOS	Pentium 200 MMX	64 MB	ISA 2.4 GHz DSSS
MYKONOS	Pentium 166	64 MB	ISA 2.4 GHz DSSS
SYROS	Pentium 150 MMX	48 MB	PCMCIA 2.4 GHz DSSS

conservative. Each host that wants to transmit waits for a silent medium, plus an interframe gap, and then chooses a random time *slot* from a *contention window*. If the medium has not been seized until that slot, the host seizes the channel. The window is set to a small value after seizing the channel, and is exponentially increased on each consecutive failure to do so. A collision occurs if many hosts pick the same random slot.

The WLAN interfaces emulate Ethernet cards in terms of programming and packet formats (MAC headers, CRCs, 1500 byte maximum packet size). After every packet reception the interface reports to the driver a *signal level*, measured at the beginning of the transmission, and a *noise level*, measured during the interframe gap after reception ends. Since these metrics are hardware dependent they are only significant with respect to their highest/lowest values, but they are also comparable with each other. A *signal quality* metric is also reported, showing (roughly) how much the signal is affected by multipath propagation. This is used to select one of the two built-in antennas.

Table I shows the names and characteristics of each host. Two hosts (IOS and MYKONOS) are desktop PCs while the third (SYROS) is a laptop. We used desktops with different processors to determine the effect of processing power on performance. The laptop has roughly the same processing power as the slower desktop, since its processor operates at a lower clock frequency but has more on-chip cache memory.

B. Software

All hosts ran the Linux OS, kernel version 2.0.32, using the supplied WaveLAN drivers as loadable kernel modules. The hosts were in multiuser mode during testing, but with no user tasks executing. The tests were performed late in the evening, to ensure that the Ethernet used for control would be unloaded. We made only a minor modification to the wireless interface drivers to record and report detailed statistics plus histograms of signal and noise levels. The histograms can be reset to all zeroes and both statistics and histograms can be dumped on demand. Preliminary tests verified that system performance remained virtually the same after our modifications.

For testing we used the `ttcp` benchmark which sends a number of packets of a specified size to a receiver using either TCP or UDP, reporting at the end various transfer and OS related metrics. We added an option for UDP tests that uses packet sequence numbers so that the receiver can detect and report packet losses (as they occur and in total), and named this version `ettcp`. Besides the statistics provided by the wireless interface driver, we used `nstat` to gather IP, UDP and TCP statistics aggregated across all interfaces, so as to check for unexpected network activity during the tests. We also used `tcpdump` to record detailed logs of all packets sent and received by the wireless interfaces during each test. These logs can be used for detailed off-line study of TCP and UDP activity.

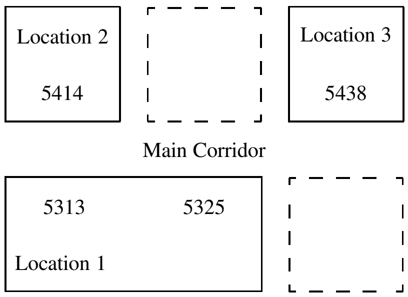

Fig. 1. Location map for the experiments

TABLE II

DESCRIPTION OF TESTING SCENARIOS

Scenario	Host A/Location	Host B/Location
1	IOS/Location 1	MYKONOS/Location 1
2	IOS/Location 1	SYROS/Location 1
3	IOS/Location 1	MYKONOS/Location 2
4	IOS/Location 1	SYROS/Location 2
5	MYKONOS/Location 2	SYROS/Location 3
6	IOS/Location 1	SYROS/Location 3

C. Environment

Fig. 1 shows a floor plan (*not* in scale) of the area where the experiments took place (5th floor of the AP&M building at UCSD). Hosts were placed at one of Locations 1, 2 and 3. These rooms are laboratories and machine rooms containing numerous hardware devices but no direct sources of interference. The distance between both Locations 1 and 2 and Locations 2 and 3 is about 45 feet, while between Locations 1 and 3 it is about 60 feet. We executed the same set of experiments for six different host and location combinations or *scenarios*, described in Table II, in both directions. Scenarios 1 and 2 show baseline performance under optimal circumstances (adjacent hosts), with either ISA only or mixed ISA and PCMCIA cards. Scenarios 3 and 4 are similar to 1 and 2 with the hosts separated by some obstacles. For Scenarios 5 and 6 we kept the ISA hosts as in scenarios 3 and 4 and moved the PCMCIA host. Before testing we used a monitoring utility to verify that ambient noise and signal levels in each location were adequate for communication. We chose locations representing various reasonable operating conditions rather than system limits. The locations did not suffer from excessive multipath fading, as the signal quality metrics were always high. Hosts were kept immobile during each test to avoid mobility induced problems.

IV. TEST AND OUTPUT DESCRIPTION

A single test consisted of executing `ettcp` with appropriate parameters and recording statistics before, during, and after the run, on both sides of the transfer. The main test parameters were transfer direction, peer names, packet size (including TCP/UDP/IP headers) and protocol to be used. A test script first reset and dumped interface statistics and `nstat` output, then started `tcpdump` to record all packets through the wireless interface, and finally started `ettcp` to transfer 10,000 packets. Although at peak speed a 1500 byte packet UDP test only takes a minute to complete, the amount of data transferred (15 MB in this case) is large enough to represent a realistic transfer session.

After the transfer ended, tcpdump was stopped, and nstat and interface statistics were again dumped for comparison with their values before the test. All tests were performed in both directions between the peers. Each test was repeated 5 times with the same parameters to estimate variance between runs.

Tests that could not terminate, for example when UDP control packets were lost, were manually stopped and repeated. Before each set of repetitions the link was reset and ping was used to establish connectivity. All tests were performed for both TCP and UDP, with four IP datagram sizes (including protocol headers): 100, 500, 1000 and 1500 bytes (the maximum). The test suite was designed to exhibit any variations caused by heterogeneous hosts and interfaces. Executing tests in both directions reveals any performance asymmetries. The multiple packet sizes show the effects of varying amounts of overhead and packet error probability on throughput. Our TCP tests show for the first time the effects of bidirectional traffic (due to acknowledgments) on a shared medium WLAN, and how it compares with unidirectional (UDP) traffic. Multiple test repetitions show how statistically significant the results are.

A variety of metrics is produced by ettcp, including transfer time, bytes transferred and total throughput. Since UDP senders do not get any feedback, we present receiver metrics for more accuracy. For UDP tests, the receiver reports each sequence of lost and received packets and their totals. Such traces can be used to calculate probability distributions for lost and received packet runs. These can be used to model the link with a two state process: a *good* state where packets are received correctly, and a *bad* state where packets are lost or corrupted [6]. We can also calculate mean, minimum and maximum values for the reported metrics across test repetitions. Throughput and loss rate are comparable across all tests since their units are independent of packet size. These can be used to determine the optimal packet size where overhead (which favors long packets) and loss (which favors short packets) are best balanced.

Interface statistics (such as number of packets received and transmitted) and histograms (signal/noise levels) were dumped before and after each run, to determine the net effect of each test. We can calculate mean, minimum and maximum values for the statistics and aggregate histograms across test repetitions. In TCP tests packets move in both directions so these statistics are important on both sides. The difference between sent and received packets between the two sides shows the actual amount of loss on the link. Any TCP losses above UDP levels can only be attributed to MAC layer collisions. Signal and noise level histograms can be compared among scenarios to see how different locations, hosts and interfaces influence them. We do not present signal quality metrics (always near their peak value) and nstat output (aggregated across all host interfaces). The tcpdump output files were examined off-line on a case by case basis, to explain the performance of specific TCP or UDP tests. These traces show all packet transmissions/receptions against time. In trace graphs, we normalize time to start from zero.

V. Analysis of Test Results

A. General Remarks

During testing we noticed that in *all* UDP tests with 100 byte packets, 90-95% of packets sent by ettcp were never transmitted, and actually did not even reach the interface according to the driver. The reason was buffer shortages at the UDP level, due to the very fast generation rate of short packets, which caused datagrams to be dropped. When running the same tests over the faster wired interfaces for comparison purposes, fewer packets (50%) were dropped as expected. TCP tests with 100 byte packets did not suffer from such drops, because TCP uses window based flow control, with a maximum window of 32 KB in our tests. This prohibits the sending process from passing to the network code huge bursts of data without pause.

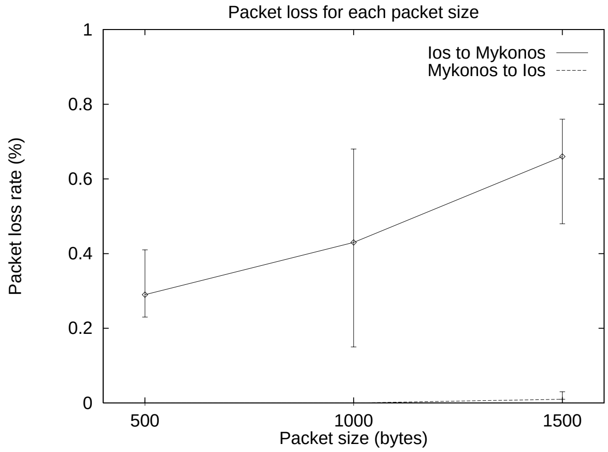

Fig. 2. Scenario 1, UDP packet loss

Even though ettcp used a TCP socket option to transmit data segments immediately, we occasionally saw packets larger than expected, except in the 1500 byte packet tests where the maximum WLAN packet size was reached. The reason is that TCP keeps track of its transmission queue in terms of bytes rather than packets [7], thus any segments whose immediate transmission is deferred may be later combined into larger packets. Such delays can be caused by MAC layer contention due to the bidirectional traffic of TCP. However, examination of tcpdump output showed that usually larger packets were sent after sending a long run of packets, stopping, and then getting an acknowledgment. This means that the sender exhausted its TCP transmission window and while waiting for an acknowledgment (which could itself have been delayed by MAC layer contention) more data segments were queued at the sender.

Another issue is determining the number of collisions at the CSMA/CA MAC layer. The hardware (through the drivers) reports a collision metric like CSMA/CD interfaces do, but since collisions are *not* detected on our WLAN, it is unclear what it represents. As it was zero with UDP tests but high with some TCP tests, it is probably related to the collision avoidance scheme, but in a non-obvious manner. We can estimate the number of real (undetected) collisions using the difference between the numbers of sent and received packets on each side of the transfer, after subtracting unidirectional loss rates.

B. Scenario 1

The first scenario employed two hosts with ISA interfaces placed next to each other to avoid signal degradation. The goal was to determine the peak performance of ISA cards and reveal processing power induced asymmetries. This is evident in Fig. 2 which shows mean, minimum and maximum packet loss

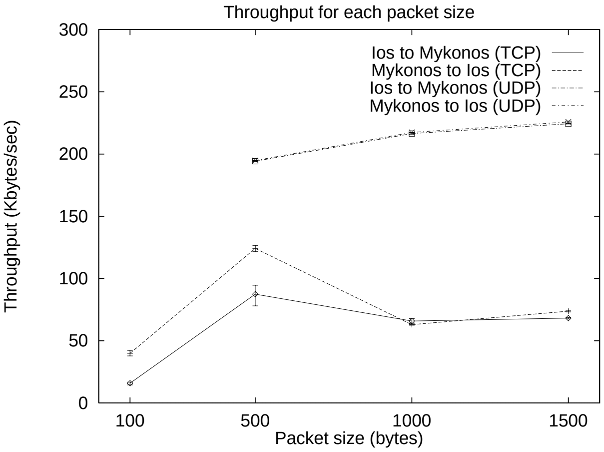

Fig. 3. Scenario 1, UDP and TCP throughput

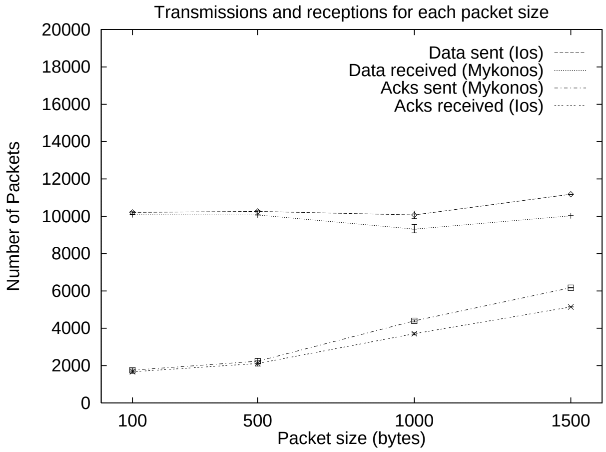

Fig. 4. Scenario 1, TCP data and acknowledgments, from IOS to MYKONOS

rate (as a percentage), across all UDP test repetitions (measured at the receiver). When the slower host (MYKONOS) is sending, packet loss is negligible. In contrast, the faster host (IOS) overwhelms a slower receiver, leading to loss (0.3–0.6%) which grows with packet size, i.e. increased receiver load. The packet loss distribution recorded by `ettcp` shows that nearly all loss periods last for one packet and occur every about 1000 packets. This implies that a fast sender semi-periodically overruns the receiver, which catches up after a single packet. The low loss in the reverse direction must be due to wireless errors.

Fig. 3 shows TCP and UDP throughput (mean, minimum and maximum across repetitions) with varying packet size. Net UDP throughput increases with packet size since UDP/IP overhead drops. The peak data rate is about 225 KBps (1.8 Mbps), after subtracting UDP/IP and MAC overhead, higher than previously reported with slower hosts [4], [5], older WLAN versions (900 MHz instead of 2.4 GHz) and OS (BSD instead of Linux). Most of the gains are due to increased processing power, which was shown to be a decisive factor in previous work. Since our considerably asymmetric hosts achieved nearly identical throughputs, it seems that we have reached the peak capacity of the WLAN. Another observation from Fig. 3 is that TCP throughput is not only below UDP, it actually *drops* with large packet sizes. In 100/500 byte tests the slower sender achieves higher throughputs as there are no losses (and retransmissions). In 1000/1500 byte tests however, throughput drops symmetrically to only about 30% of UDP, despite different loss rates.

This phenomenon is explained in Fig. 4, showing mean, minimum and maximum number of data and acknowledgment packets sent/received for IOS to MYKONOS transfers (across repetitions). The gaps between sent and received curves for both packet types show considerable loss on the link, growing with packet size. Since the gaps are roughly the same, we conclude that their magnitude represents the number of undetected collisions of CSMA/CA. Results for transfers in the reverse direction are similar, hence the symmetric TCP throughput results. Retransmissions due to these losses are clear in the 1500 byte tests where the total number of packets *sent* exceeds 10,000. Queued segments are combined into larger packets in the 1000 byte tests, as explained above, where the total number of packets *received* is less than 10,000, despite retransmissions.

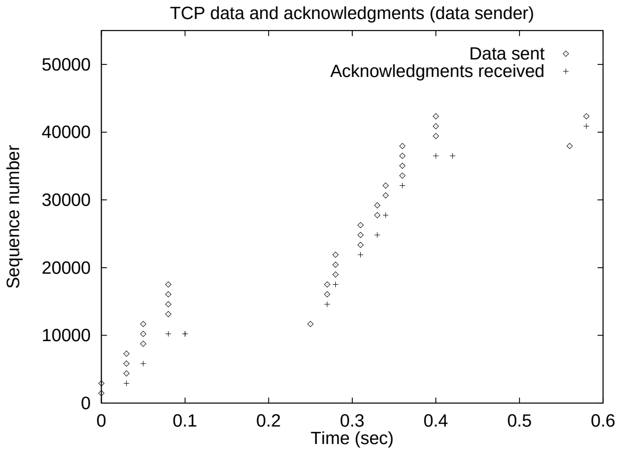

Fig. 5. Scenario 1, TCP trace snapshot (1500 bytes, sender's view)

To explain how a 10% collision loss rate causes so much throughput degradation, we turn to `tcpdump` TCP traces. When data and acknowledgments collide, some data packets after the lost one are retransmitted needlessly. In addition, when some of the duplicate acknowledgments that TCP returns to initiate *fast retransmission* [7] are lost, if less than three duplicates arrive the sender stalls until a timeout occurs. Fig. 5 and Fig. 6 show a `tcpdump` trace for the beginning of a 1500 byte packet TCP test from MYKONOS to IOS. Fig. 5 shows the packets seen by the data sender and Fig. 6 shows the viewpoint of the data receiver. A collision occurs when one data and one acknowledgment packet are shown on the sender but not on the receiver. At time 0.05 three data packets with consecutive sequence numbers are sent (Fig. 5) and the third is lost (Fig. 6): it has collided with the acknowledgment for the first of these data packets (it appears on Fig. 6 but not on Fig. 5). Right after the (undetected) collision, four new data packets are sent. The receiver replies with three duplicate acknowledgments to signal the loss, but the second one collides with a new data packet.

Since only two duplicates were received, the sender stalls having exhausted its transmission window, until a timeout occurs at time 0.25 triggering retransmission of the first lost data packet, 200 ms after the loss. Linux uses a 10 ms granular-

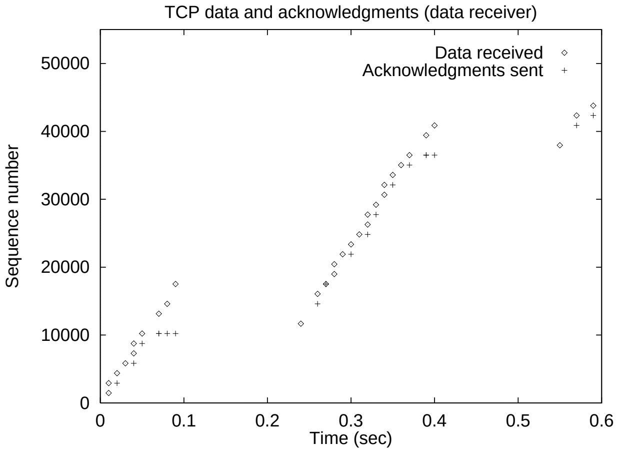

Fig. 6. Scenario 1, TCP trace snapshot (1500 bytes, receiver's view)

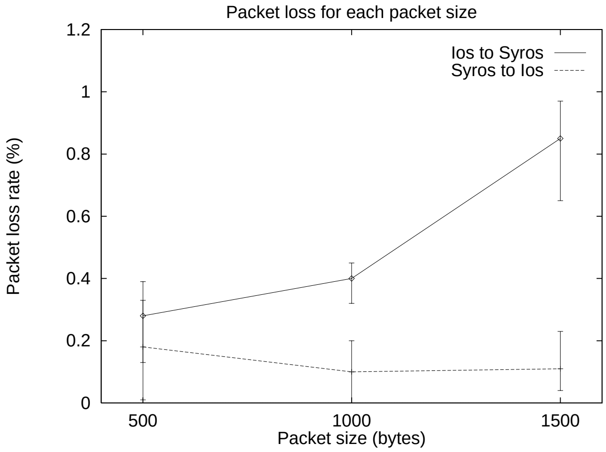

Fig. 8. Scenario 2, UDP packet loss

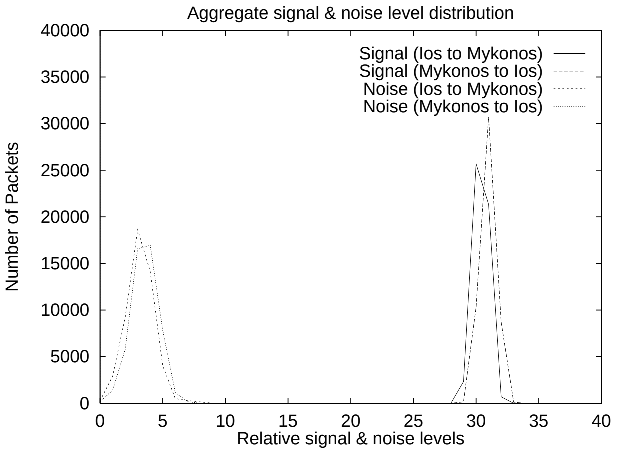

Fig. 7. Scenario 1, signal and noise level histogram (TCP, 1500 bytes)

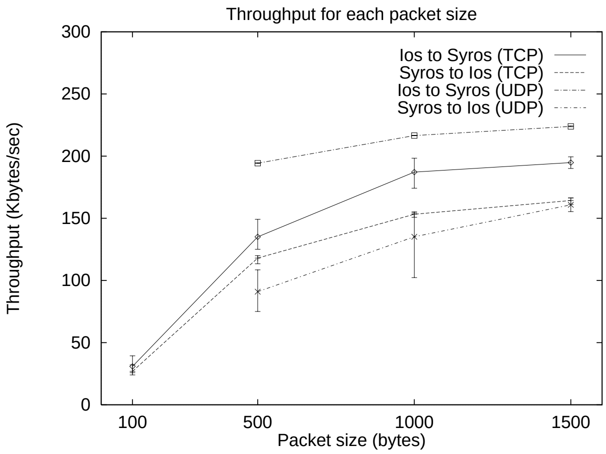

Fig. 9. Scenario 2, UDP and TCP throughput

ity clock with a 200 ms lower limit for TCP timeouts, during which the channel is mostly idle in the trace. The timeout reduces the congestion window to one segment and the host goes in *slow start* mode [7], retransmitting some already received data. Around time 0.4 another collision occurs, but this time only two duplicates are returned, as another subsequent data packet is lost (not to a collision). Frequent losses have caused the *ACK clock* property of TCP to be lost, i.e. the sender exhausted its window before transmitting enough data after the loss to trigger three duplicate acknowledgments [8]. The idle periods between timeouts are thus the main reason for low TCP throughput. Note that the situation would be much worse with the 500 ms granularity timers used in many BSD derived systems, since the idle periods would be correspondingly longer.

Fig. 7 shows the received signal and noise level distributions (in both directions) aggregated across all 1500 byte TCP tests. These distributions were practically identical across protocols and packet sizes within any given scenario. Signal and noise metrics are comparable as they are expressed in the same (hardware defined) units. Both histograms are nearly symmetric since the peer interfaces were exactly the same and host processing power does not influence the radios. Since hosts were next to each other, these are the best case distributions.

C. *Scenario 2*

The second scenario employs one ISA and one PCMCIA host, again placed next to each other, to establish a performance baseline for mixed interface tests. The processing power of SYROS and MYKONOS is roughly equivalent, thus comparisons with the first scenario are direct. Indeed, the UDP loss rate shown in Fig. 8 is similar (0.3–0.8%) when the faster host is sending, implying that both ISA and PCMCIA receivers are overwhelmed by faster senders. In the reverse direction, the perceived losses (0.1–0.2%) are due to packets never leaving the sending interface, according to `tcpdump`. This is caused by the single transmit buffer of PCMCIA cards (ISA cards have multiple buffers) which is easy to overrun, especially with smaller, faster generated, packets. `ettcp` loss traces show, as expected, single packet losses every about 1000 packets in the ISA (faster) to PCMCIA (slower) direction. In the reverse direction loss periods are less frequent but longer (1–5 packets).

Fig. 9 shows TCP and UDP throughput for all tests. In the ISA to PCMCIA direction UDP is faster than TCP, due to less header overhead and the absence of TCP retransmissions and acknowledgments. TCP throughput in the reverse direction is slightly lower, verifying previous claims that PCMCIA cards

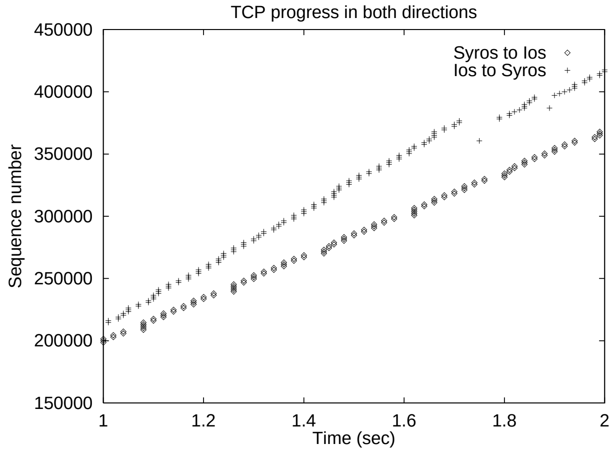

Fig. 10. Scenario 2, TCP data (1500 byte packets)

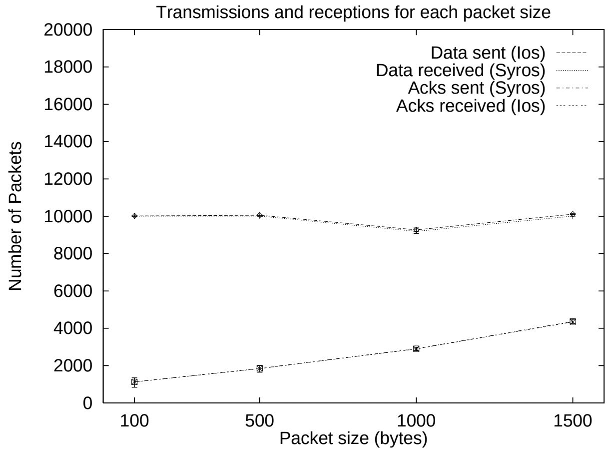

Fig. 12. Scenario 2, TCP data and acknowledgments, from IOS to SYROS

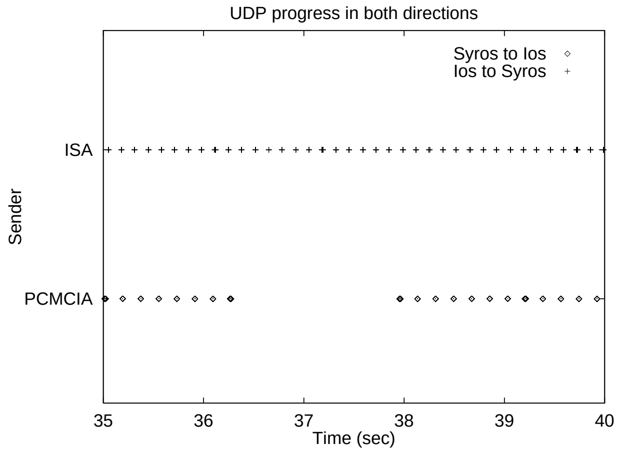

Fig. 11. Scenario 2, UDP data (1500 byte packets)

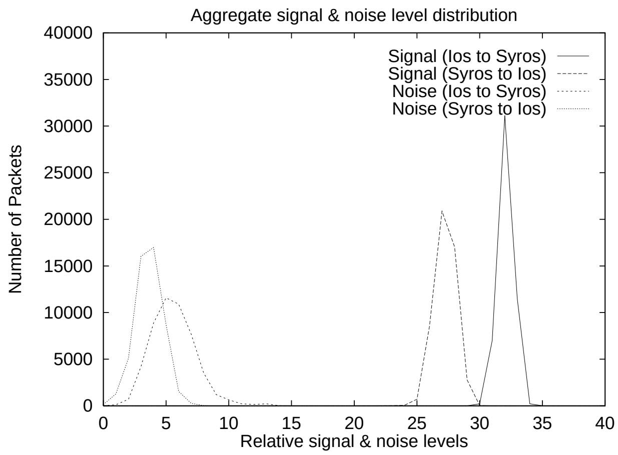

Fig. 13. Scenario 2, signal and noise level histogram (TCP, 1500 bytes)

are slower senders [6]. This is clear in Fig. 10, showing two snapshots of TCP progress: sequence numbers increase faster with an ISA sender despite occasional retransmissions. The PCMCIA sender leaves short gaps between transmission bursts, due to the transmit buffer shortages mentioned above. The reduced slope is due to longer interframe gaps (96 bits) than ISA senders (32 bits), as shown in Fig. 11 where two snapshots of UDP progress are exhibited. In BSD systems the interframe gap is the same for both cards, but the contention window is larger for PCMCIA cards (it is the same for both cards in Linux), again making PCMCIA senders slower [6].

Fig. 9 also shows that, unexpectedly, UDP is slower than TCP in the PCMCIA to ISA direction. Although segment aggregation helps TCP for the 1000 byte packet tests, retransmissions reduce TCP performance. The real cause for the low UDP performance is shown in Fig. 11: occasional long PCMCIA sender pauses despite the lack of contention with UDP. The `tcpdump` traces reveal that packet losses occur exactly at the beginning of these pauses. The most likely explanation is that severe transmit buffer overruns cause interface controller resets. Such overruns are avoided with TCP due to its flow control mechanisms. This problem reduces peak UDP throughput to 160 KBps (1.28 Mbps) with a PCMCIA sender. In the reverse direction (ISA sender) throughput reaches 223 KBps (1.78 Mbps), i.e. it is similar with both ISA and PCMCIA receivers.

Fig. 12 shows only a small discrepancy between sent and received data packets on the ISA to PCMCIA direction (close to the loss rate), with practically overlapping acknowledgment curves. This implies that collisions are practically zero and justifies the improved TCP throughput compared to the first scenario. The reverse direction is similar. We conclude that the slight differences in interface implementations and timing eliminate the synchronization phenomenon that causes excessive collisions among ISA cards. Note the aggregation of data segments in 1000 byte tests. The signal and noise level histograms shown in Fig. 13 are asymmetric, with the ISA receiver detecting lower signal levels and the PCMCIA receiver detecting higher noise levels. This may either be an artifact of implementation differences or an indication that PCMCIA radios and antennas are weaker and less tolerant to noise. These are the best case distributions for mixed ISA/PCMCIA tests.

D. Scenarios 3–6

The remaining four scenarios did not contribute as many new issues as the first two, but confirmed our hypotheses about the causes of already seen problems by exhibiting predictable changes due to variations in the testing environment. Scenario 3 was the same as scenario 1 but with a 45 feet distance and two intervening walls between the peers (Fig. 1). The loss rates and distributions were similar, indicating again that a faster sender overruns a slower receiver. Also, throughput curves were similar due to excessive CSMA/CA collisions in TCP tests. The main difference with scenario 1 was a signal level distribution with uniformly lower receiver metrics (10–15 units), due to increased distance and obstacles. Noise levels were nearly the same though, maintaining an adequate margin between signal and noise, leading to practically identical performance. Scenario 4 differs from scenario 2 in the same way, i.e. it mixed one ISA and one PCMCIA host separated by 45 feet. Loss rates and distributions, as well as throughput curves are nearly the same, due to the reasons discussed in scenario 2, namely less aggressive PCMCIA sending behavior, with occasional stalls and even resets in UDP. Signal levels are uniformly lower as in scenario 3, with the same noise level. We conclude that this distance and obstacles do not have measurable effects on performance in both ISA/ISA and ISA/PCMCIA tests.

In scenario 5 the *slower* ISA host communicates with the PCMCIA host over a distance of 45 feet, making this the only scenario where hosts have comparable processing power. The main difference with scenarios 2 and 4 (also mixed ISA/PCMCIA) is a nearly zero loss rate in the ISA to PCMCIA direction. This shows that hosts matched in processing power avoid losses due to receiver overruns. It also causes TCP throughput in this direction to reach 213 Kbps (1.7 Mbps), the highest in our tests, achieved by the slower ISA host. Signal metrics are slightly lower than scenario 4 despite a similar distance, due to differences in intervening obstacles. Scenario 6 is another variation on scenario 4, with a higher distance (60 feet) between two (asymmetric) ISA and PCMCIA hosts. Apart from the expected lower signal level metrics (still far from noise levels), the only difference with scenario 4 is a slightly higher loss rate, both due to the increased distance and obstacles.

VI. Discussion of Test Results

Our measurements uncovered numerous problems with TCP and UDP performance over our WLAN due to parameters first examined in our work, such as bidirectional traffic, host and interface heterogeneity, and different OS and device drivers. The 2.4 GHz DSSS WaveLAN system performed very well overall in our office environment. Our faster hosts, compared to previous tests, apparently achieved the peak UDP throughputs for this system: 1.8 Mbps between ISA cards, 1.78 Mbps from ISA to PCMCIA and 1.28 Mbps from PCMCIA to ISA. Since our asymmetric ISA hosts exhibited similar performance, we conclude that peak throughputs can be achieved by a 166 MHz Pentium system. We also confirmed that ISA cards are faster senders than PCMCIA cards due to more aggressive timing, even under a different OS and device driver settings [6]. Although our throughput results follow a trend in reported results [4], [5], [6] of increasing data rates with faster hosts, they could also be partly due to our newer WaveLAN version.

Fast processors can also overwhelm the Linux networking code when sending short UDP packets at peak speed, a problem that affects to a lesser extent wired Ethernets. It does not arise when the sender is throttled back by flow control, as with TCP, which is normally used for large transfers. The main problem with faster ISA senders is that they can overrun slower receivers (both ISA and PCMCIA) causing sporadic packet losses. PCMCIA senders are inherently slower, so they avoid this problem. Losses cause frequent retransmissions and invocation of congestion control procedures for TCP, thus reducing throughput. Pacing the sender in software is infeasible with Linux as even its 10 ms granularity timers are too coarse to be used to introduce pauses before each transmission.

Another difference between ISA and PCMCIA cards appears in their signal and noise level metrics, which probably diverged due to dissimilar card and antenna implementations, but could also imply that PCMCIA cards have a shorter range. An important distinction is that PCMCIA senders always dropped a few packets *before* transmission. This phenomenon is due to PCMCIA transmit buffer limitations that cause the sender to stall periodically with TCP, increasing the performance gap with ISA senders, and even reset itself completely under UDP, dropping packets and pausing for a long period of time. TCP flow control prohibits it from overwhelming the transmitter to the extent of causing a reset. In contrast, UDP suffers so much from resets that its throughput is below TCP. ISA senders do not face these problems because ISA cards use multiple transmit buffers instead of a single one in PCMCIA cards.

Since our WLAN does *not* detect collisions [3], an important issue is how successful the CSMA/CA MAC layer is in avoiding them, and TCP tests with their inherently bidirectional traffic help us examine exactly that. We ignored the obscure collision metric reported by the driver since the hardware does not support it. Instead, we estimated the actual (undetected) collision rate by comparing the number of sent and received data and acknowledgment packets seen at either side of a transfer. Collisions proved to be a significant problem in the ISA to ISA case, adding 10% packet loss and decreasing the throughput of TCP to only 30% of UDP, despite otherwise low loss rates. While data loss causes retransmissions of even already received packets, loss of duplicate acknowledgments or of the ACK clock property of TCP [8] can lead to timeouts, with the sender stalled for most of the timeout interval. These collisions only affect ISA to ISA transfers, not mixed ISA and PCMCIA scenarios, implying that identical interface timing can lead to undesirable synchronization. Although we have not tested it, it is possible that PCMCIA to PCMCIA transfers could avoid such problems due to their less aggressive timing that leaves more room for sending acknowledgments. Relaxing the timing to improve TCP performance would reduce UDP performance, as evidenced by the lower UDP performance of PCMCIA senders.

At the TCP level the adverse collision effects could be reduced by smarter retransmission policies such as *selective acknowledgments* (SACK) which handle frequent losses better [8] by returning more accurate feedback to the sender. At the MAC layer these problems could be avoided by MAC acknowledgments and retransmissions, as proposed by the IEEE 802.11 standard [1]. Even though collisions still waste bandwidth in the absence of immediate collision detection, MAC retransmissions would still improve TCP performance by avoiding the slower timeout based recovery of TCP. Long chains of retransmissions would cause TCP to timeout anyway and retransmit the same data as the MAC layer [9], a more likely event with short paths *and* tight TCP timers. MAC retransmissions are wasteful for protocols and applications that prefer sending new packets to retransmitting old ones, such as real time audio over UDP. This argues for offering limited and/or protocol dependent retransmission policies at the MAC layer.

One goal was to gather additional data for modeling the wireless channel with a two state process [6]. Although our results can be used to calculate *good* and *bad* state distributions for unidirectional (UDP) traffic, note that the resulting process does not model *only* wireless impairments. Losses were also caused by many implementation details, such as receiver overruns and transmit buffer limitations. However, since these factors are critical in determining perceived network performance, it is necessary to include them in a performance model. Another complication is that the multitude of factors affecting performance, such as packet size, interface types and host processors, does not allow direct generalizations for modeling purposes. This reinforces our original view that comprehensive WLAN measurements, as described in this paper, *cannot* be replaced by a simple mathematical model based on a few actual measurements and interpolation.

Given the severe MAC layer collision problems experienced in TCP transfers between ISA interfaces, it is questionable whether unidirectional traffic models can be used to simulate bidirectional transfers. For TCP in particular, accurate modeling requires simulating the loss distribution *and* the correlation between losses in both directions, i.e. how data and acknowledgment packets collide. Due to these synchronization problems with TCP traffic, the *complete* CSMA/CA protocol must also be simulated in order to get accurate results. In particular, it is *not* enough to simulate CSMA/CD with an extra penalty on collisions to compensate for not aborting garbled transmissions. The fact that CSMA/CD collisions are *not* losses inherently differentiates them from CSMA/CA collisions.

Finally, it is interesting to note the effect of timers with fine granularity on TCP. Linux uses 10 ms granularity timers, as opposed to 500 ms for many BSD derivatives, which may become unstable if measures, such as the 200 ms lower limit for timeouts, are not taken. Eliminating such safeguards can lead to very tight estimates of round trip time and variance, causing timeouts to occur after only minor delays. In our WLAN tests however, the TCP traces showed that when the sender was stalled due to multiple losses, the shorter 200 ms timeouts reduced the idle period between transmissions. With coarser (500 ms) timers these losses would cause TCP throughput to diminish. It is an open question whether fine timer granularity is beneficial for longer paths [10] or when MAC retransmissions (that may increase delay and its variance) are used, as in the IEEE 802.11 standard.

VII. Conclusions

Our measurements extend previous work on TCP and UDP performance over WLANs in many directions. In particular, for the first time we present results for (bidirectional) TCP traffic, asymmetric host processors and heterogeneous wireless interface implementations. We uncovered many performance problems and investigated their causes, which were variously attributed to network and host hardware, device drivers, network protocol implementation in the OS, and their interactions. We also discussed the effectiveness of various proposed improvements. While our performance measurements reflect a particular testing setup, a detailed investigation of the root causes of these problems is of value to both hardware and software designers in order to avoid such pitfalls in the future.

In summary, achievable UDP throughput reaches 1.8 Mbps for ISA senders and 1.28 Mbps for PCMCIA senders. It is influenced by host processing power but reaches a plateau with a 166 MHz Pentium host. Fast senders can overwhelm slower receivers (both ISA and PCMCIA), leading to semi-periodic packet loss. PCMCIA senders pause every few packets under TCP due to transmit buffer limitations, and are generally slower than ISA senders due to less aggressive timing. PCMCIA transmit buffer shortages become so severe under UDP, due to the absence of any flow control, that the interfaces suffer from lengthy communication pauses during resets.

When two ISA hosts communicate, many collisions occur under TCP, leading to very degraded performance because of slow timeout initiated recovery. Fine granularity timers speed up these timeouts, but may interfere with MAC retransmissions. As long as these synchronization problems are avoided, CSMA/CA performs well with bidirectional traffic, as evidenced by our mixed ISA and PCMCIA tests. Less aggressive MAC timing and/or better TCP recovery mechanisms could help CSMA/CA eliminate or reduce the effects of such synchronization. MAC layer retransmissions could also be beneficial for TCP but problematic for other protocols. UDP loss models can be easily formulated for simulations, but they should include more factors than wireless impairments. They should be based on actual measurements rather than simple interpolations. TCP simulation models should also include CSMA/CA collisions between data and acknowledgment packets for accuracy.

References

[1] B.P. Crow, I. Widjaja, J. Geun Kim, and P.T. Sakai, "IEEE 802.11 wireless local area networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, September 1997.

[2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *Proc. of the ACM SIGCOMM '96*, August 1996, pp. 256–267.

[3] B. Tuch, "Development of WaveLAN, an ISM band wireless LAN," *AT&T Technical Journal*, vol. 72, no. 4, pp. 27–37, July/August 1993.

[4] D. Duchamp and N.F. Reynolds, "Measured performance of a wireless LAN," in *Proc. of the 17th IEEE Conference on Local Computer Networks*, September 1992, pp. 494–499.

[5] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *Proc. of the ACM SIGCOMM '96*, August 1996, pp. 243–254.

[6] G.T. Nguyen, R.H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proc. of the Winter Simulation Conference*, 1996.

[7] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," Internet Request For Comments, January 1997, RFC 2001.

[8] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, pp. 5–21, July 1996.

[9] A. DeSimone, M.C. Chuah, and O.C. Yue, "Throughput performance of transport-layer protocols over wireless LANs," in *Proc. of the IEEE GLOBECOM '93*, December 1993, pp. 542–549.

[10] S. Floyd, "TCP and explicit congestion notification," *Computer Communications Review*, vol. 24, no. 5, pp. 8–23, October 1994.