

# Sniffing out the correct Physical Layer Capture model in 802.11b

## CS-TR-4583

### UMIACS-TR-2004-26

Andrzej Kochut, Arunchandar Vasani, A. Udaya Shankar, Ashok Agrawala  
Department of Computer Science, University of Maryland, College Park, Maryland 20742  
{kochut, arun, shankar, agrawala}@cs.umd.edu

## Abstract

*Physical layer capture (PLC) in 802.11b refers to the successful reception of the stronger (higher signal strength at receiver) frame in a collision. PLC causes significant imbalance in the throughputs of sources. Existing 802.11b simulators, including NS-2 and Qualnet, assume that PLC occurs only if the reception of stronger frame starts first at the receiver. We show empirically that in reality PLC occurs even if the stronger frame arrives later (but within the physical layer preamble of the first frame). We have modified the NS-2 simulator to account for this and Qualnet will be incorporating a fix in their next release. Simulations using the latter PLC model generate significantly (up to 15%) higher, and more realistic, throughput unfairness than with the former PLC model. We also show that time-instants of start of colliding frames routinely differ by as much as 20  $\mu$ s due to inherent uncertainties of 802.11b firmware clock synchronization and rx/tx turnaround delays, and that the frame to arrive first can be either the stronger or the weaker with equal likelihood. To identify which frames were involved in collisions, when their transmissions started, and which of them were retrieved, we have devised a novel technique using multiple sniffers and instrumented device drivers to reconstruct from the air interface all tx/rx events in a WLAN to within 4  $\mu$ s accuracy. This allows us to quantify the causal links from the PHY layer through the MAC layer to the observed application layer imbalance, which may be as high as 25 % with two sources and 75 % with four sources as seen in experiments spanning ad-hoc and infrastructure modes with both UDP and TCP workloads.*

## 1 Introduction

Consider two 802.11b stations, A and B, transmitting data to a wired sink S through access point C, in the DCF (Distributed Coordination Function) mode under the fol-

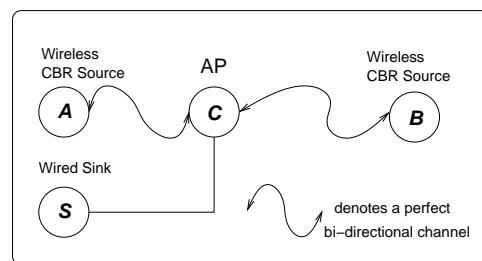


Figure 1: Scenario showing imbalance in throughputs obtained by sources though their channel to the AP is perfect.

lowing natural conditions: (1) each of A and B have a “perfect” channel to C when the other is not transmitting, i.e., gets the maximum throughput at the highest bitrate; (2) A’s signal at C is stronger than B’s signal at C by a few dB; and (3) A, B, and C are all in range of each other, i.e., no hidden terminal. The workload considered is a UDP source on each station with a constant rate sufficient to saturate the available capacity. The scenario is shown in Figure 1.

Our experiments, done with a variety of 802.11 cards, show that A’s goodput *as well as throughput* are consistently higher than B’s in the long term (over intervals longer than, say, 10 seconds). Here, goodput refers to the number of successful transmissions per second and throughput to the number of (re)transmissions per second, including lost tries.

Because A and B achieve equal performance when the other is not transmitting, the imbalance can only be because of interaction between A and B when both are transmitting simultaneously. Indeed, conditions 1-3 above allow us to conclude that the imbalance can only be because collisions are resolved in favor of A, rather than, say, both frames being lost. The phenomenon of the stronger frame in a collision being received successfully is known as *physical layer capture (PLC)* [1].

Note that the imbalance cannot be caused by B’s frames having a poor signal-to-noise ratio, because B experiences

a perfect channel when A is not transmitting. The imbalance also cannot be caused by B switching to a lower transmission bitrate, as that would only reduce each station's throughput by same amount. This is because, as explained in detail in [2], the 802.11 MAC guarantees fair access to the media (in terms of the number of transmissions) in the long term and does not abort a transmission in progress (no matter what the bitrate).

Here are possible models for what happens when frames of different signal strength collide at a receiver:

- I Both frames are lost.
- II The stronger frame is received correctly provided it is sufficiently stronger and it is the first to arrive.
- III The stronger frame is received correctly provided it is sufficiently stronger, regardless of whether it arrives before or after the weaker frame.

Clearly, we can rule out model I because it does not generate the observed imbalance. Both models II and III generate imbalance in favor of A, model III more so than model II. To the best of our knowledge, existing analytical models for 802.11 MAC throughput [3, 4, 5] assume model I. Simulators such as NS-2 [6] and Qualnet [7] use (deterministic or probabilistic versions of) model II. **We show that in reality, collisions are resolved according to model III.**

The 802.11b standard [8] does not preclude this kind of capture (see PHY receive process in pages 202-203). Each frame starts with 192 bits of Physical Link Control Protocol (PLCP) preamble and header, consisting of 128 sync bits, 16-bit Start Frame Delimiter (SFD), and 48 bits indicating, among other things, the bitrate and length of the MPDU (data) that follows. These 192 bits are always transmitted at 1 Mbps. If a stronger frame arrives while a receiver is receiving a frame, the stronger frame can be captured provided it starts before the SFD of the first frame.

Naturally, the difference between models II and III with respect to throughput imbalance is most significant when the stronger frame almost always starts second; we demonstrate such a simulation scenario in Section 4, showing throughput imbalance of 15% with model III as opposed to 0% with model II. The PLC imbalance is typically magnified by protocol control mechanisms as one goes higher up in the protocol stack: the MAC's Binary Exponential Backoff (BEB) algorithm magnifies the raw PLC imbalance by increasing the backoff window of the losing station; the TCP congestion control magnifies the imbalance at the MAC layer by throttling the send rate; and so on.

We also show that the time instants of start of reception of colliding frames at the receiver routinely differ by as much as 20 microseconds, and that the frame to start reception first can be either the stronger or the weaker frame with equal likelihood. This happens even when all stations

are in range of each other, due to the rx/tx turnaround delay at stations and inherent uncertainties in 802.11 firmware clock synchronization. The time differences can be more if hidden terminals exist.

To demonstrate these behaviors in real-life 802.11 networks, we have developed a technique for reconstructing from the air interface the timeline of all transmission (including retransmission) and reception events in an 802.11 WLAN to an accuracy of  $4\mu\text{s}$ . Hence we can identify *all* the frames that collided, their arrival times, and whether any of them were received (due to physical layer capture). Combining this with an instrumentation of the WLAN device driver and the TCP/UDP/IP subsystem, we are able to trace the impact of physical layer capture through the protocol stack, and thereby explain the observed imbalance in the application layer throughputs.

## 1.1 Impact of our findings on simulators

NS-2, probably the most detailed network simulator in the open source domain, uses a deterministic version of model-II; if the earlier of two colliding frames is stronger than the later by 1 dB, it is captured. Qualnet, a state-of-the-art commercial network simulator, uses a probabilistic version of model-II; if the earlier frame is stronger, it treats the later frame's signal as noise for the earlier frame, computes the resulting bit error rate (this depends on the earlier frame's bitrate and modulation), and appropriately flips bits of the earlier frame.

Neither simulator allows the stronger frame to be captured if it starts later than the weaker frame, and it seems reasonable to expect the same of other 802.11-capable network simulators. Consequently, the throughput imbalance predicted by simulation can differ significantly from the imbalance observed in reality, for example, the imbalances can differ by 15% with NS-2. The difference can be greatly reduced by changing the simulator's collision handling to that of model III, and we have shown this for NS-2. We discuss this further in Section 4. We have informed the maintainers of NS-2 and the Qualnet support staff about this issue, and Qualnet will incorporate a fix in their next release [9].

## 1.2 Identifying collisions and their resolution

How does one determine what happens in a collision when one or both frames may be lost? To identify the correct collision model (I, II, or III), we need to identify in each collision precisely which frames were involved in the collision, which frame arrived first, which frame was stronger, and how the collision was resolved (were both frames lost or was one of them received).

Clearly, we need to generate a global timeline of all transmissions (including retransmissions) and all receptions

in the WLAN. This is not trivial for several reasons. The information needed for this is available only at the firmware level or the air interface (e.g., the number of retransmissions a frame underwent is not available at the device driver). Access to the firmware code is not feasible (in our case, because of the stiff license fee). Hence, we need to *sniff* packets in the air.

However, a single sniffer cannot capture everything because it too is subjected to PLC (so the frame retrieved by it may, in fact, be the one lost by the intended receiver). So we need many sniffers, which then leads to the problem of merging the logs of different sniffers on a common timeline, i.e., synchronizing the timestamps from the various sniffers to an accuracy adequate to resolve collisions. We solve this problem, achieving timing accuracy of better than  $4\mu\text{s}$ , by using timestamps generated by the firmware and synchronizing them via receptions of beacon frames. We explain this technique, which can be used in itself as a comprehensive WLAN monitor, in Section 3.

### 1.3 Causal link between PLC and higher-layer throughput imbalance

Several papers in the literature (e.g., [10, 11]) have pointed out that difference in signal strengths at the receiver is accompanied by unfairness in application layer throughput. They correlate application layer throughput to signal strength, but they do not explain *why* it happens. They do not examine or quantify the causal links from the PHY layer through the MAC layer to the application layer.

We instrument wireless stations at the lowest layer available to us – the device drivers of the WLAN cards and the TCP subsystem in the Linux kernel. We first isolate the effect of PLC by itself by examining an ad-hoc network in the broadcast mode, *thereby disabling the MAC backoffs and retransmissions*. Then, we consider how the MAC magnifies the PHY layer imbalance. We explicitly identify the frames which collided and how they were resolved at the receiver, and relate the observed difference in this to the throughputs obtained by different stations.

### 1.4 Summary of contributions

Our main contributions are as follows:

- We identify from real-life measurements that model III is the correct model for physical layer capture. Because the throughput and goodput imbalance caused by model III is significantly higher than that caused by models II and I, this has a practical consequence to simulators, including NS-2 and Qualnet, which use model II. Incorporating collision model III in NS-2 makes it more realistic. Qualnet will incorporate this fix in their next release.

- We develop a novel technique for reconstructing from the air interface a time line of all transmission (including retransmission) and reception events in an 802.11 WLAN to an accuracy of  $4\mu\text{s}$ . This allows one to identify all the frames that collided, their arrival times, and how collisions were resolved at a receiver.
- We present a detailed analysis of the causal link between PLC and application layer throughput imbalance. This explains why a station with a perfectly fine channel to a sink suffers an unfair degradation in service when another station whose frames have stronger signal strength at the sink starts to compete with it.

### 1.5 Roadmap

Section 2 surveys related work. Section 3 describes our method to construct the global timeline of WLAN events. Section 4 identifies issues with simulator models and suggests fixes in light of our findings. Section 5 describes our kernel space instrumentation and equipment. Sections 6, 7, and 8 describe our experiments and our inferences. Section 9 concludes.

## 2 Related work

Related work falls into two groups. The first group considers models for physical layer capture. A survey of communications literature shows that existing analytical models for obtaining probability of frame capture typically consider the stronger frame starting first (e.g., [12, 13, 14]). However, there are patents, including one on the WaveLan precursor of 802.11 [15], that also consider the stronger frame starting later.

To the best of our knowledge, reference [16] is the only work which tries to identify what capture model is used in 802.11. This work simulates different capture models [12, 13, 14, 15] to identify which of these explain the extent of unfairness observed. Thus, it indirectly infers the appropriate capture model, but does not provide any real-life evidence. Our work, on the other hand, provides concrete evidence of the correct capture model by the use of a novel technique that reconstructs all collisions and their resolutions.

The second group of related work is real-life measurement based studies which consider unfairness in 802.11 throughput. References [10] and [11] do not consider the impact of PLC, but focus on MAC-layer performance. Reference [17] analyzes location-dependent throughput performance, but does not explain the mechanism leading to this. Reference [18] is closest to our work. It correlates the TCP throughput to signal strength at the macroscopic level in presence of a hidden terminal effect only. Specifically, the

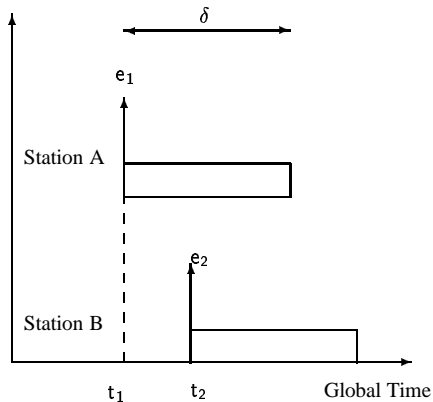


Figure 2: A collision between frames sent in Tx events  $e_1$  and  $e_2$

authors neither identify collisions and PLC nor provide evidence of how collisions are resolved and how this impacts higher-layer protocols. We are able to precisely identify and quantify the impact of PLC by itself and how it is magnified by higher layers.

### 3 Reconstructing all Tx/Rx events in a WLAN

How does one determine whether a collision took place, if one (or both) of the colliding frames is lost? Recall that the firmware of a transmitting host does not expose the time instants of start of transmission or number of retransmissions a frame underwent. More generally, how does one develop a comprehensive WLAN monitor? Unlike a wired network interface, a single vantage point is insufficient because the signal strengths, channel conditions, etc., observed by different hosts on the WLAN vary significantly (depending on their relative physical locations). This also means that different hosts resolve a collision differently, depending on which of the colliding frames had a higher signal strength at their receivers. We exploit this fact to develop a comprehensive monitoring algorithm which provides a global timeline of all events in a WLAN, and consequently, identifies which transmissions collided and how they were resolved at a receiver.

We do this by having several stations – one for each traffic source – acting as *sniffers*. A sniffer is a station with a WLAN card in *monitor* mode, i.e, it passively captures all frames in the channel. Observe that a sniffer is also subjected to PLC. We locate each sniffer close enough to its associated source so that it resolves collisions involving transmissions of its associated source in favor of that source.

Define a Tx event to be the start of a frame transmission and an Rx event to be the start of a frame reception. Given a global timeline of all Tx events in the WLAN, we can identify transmissions that overlap in time and thereby

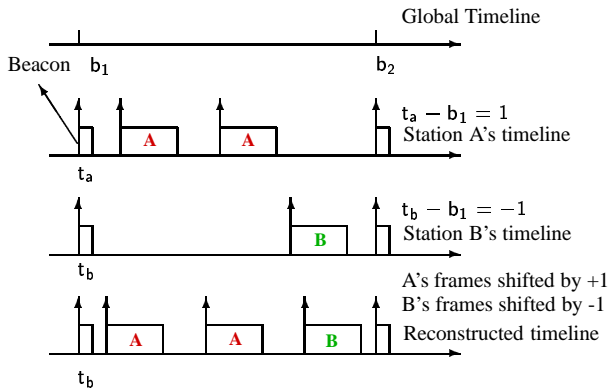


Figure 3: Global timeline construction

detect collisions. For instance, Figure 2 shows Tx events  $e_1$  at station A and  $e_2$  at station B at time instants  $t_1$  and  $t_2$ , respectively, with  $t_1 < t_2$ . This pair of events constitutes a collision if  $t_2 - t_1$  is less than  $\delta$ , the duration of the frame transmission that started at  $t_1$ .

Neglecting propagation delay in the WLAN (which is limited by the standard to  $1\mu s$  but is in the order of nanoseconds due to the range of 802.11b), each transmitter's Tx event generates an Rx event at its associated sniffer instantaneously.

Because every transmitter X has an associated sniffer that receives (practically) all frames transmitted by X, all the sniffers together have a record of all Tx events in the WLAN, with each Tx event record timestamped by the receive time according to the associated sniffer's WLAN card clock.

Hence to merge the sniffers logs into a global timeline of Tx events, all that is needed is to adjust the timestamps of different sniffers so that they are all synchronized to a common clock.

To do this, we exploit the fact that 802.11 cards in normal operation synchronize their clocks to within  $4\mu s$  of a "global" WLAN time that is disseminated in beacon frames and computed either by the AP (if infrastructure mode) or by a distributed algorithm involving all nodes (if adhoc mode) [8]. Sniffer cards, which are in monitor mode passively listening to all traffic, are *not* in normal operation and hence do not synchronize their clocks in this way. But the difference between a sniffer's clock and the global time is sampled by the difference between the receive timestamp of a sniffed beacon and the global timestamp within the beacon.

Therefore, we "synchronize" the receive timestamps in the sniffer logs by post-processing the logs as follows: Let  $t_1$  and  $t_2$  be time instants of two consecutive beacon Rx events as logged by a sniffer card's clock. Let  $b_1$  and  $b_2$  be the (WLAN's global) timestamps in the beacons. Thus  $t_1 - b_1$  is the offset between the global clock and the sniffer card's clock at time  $b_1$ . For all records in the interval

$[t_1, t_2]$  in the sniffer log, compensate the record’s timestamp by subtracting the offset  $t_1 - b_1$ . The drift in the firmware clock for the inter-beacon duration is small enough ( $< 1\mu s$ ) to be ignored.

Figure 3 illustrates the algorithm for two sniffers, A and B. Two successive beacons with global timestamps  $b_1$  and  $b_2$ , respectively, are received by both A and B. The local clock of A’s card is 1 unit ahead of the global time, i.e.,  $t_a = b_1 + 1$ . Likewise, the local clock of B’s card is 1 unit behind of the global time, i.e.,  $t_b = b_2 - 1$ . Therefore, A’s frames from the first beacon to the second are shifted back by 1 unit, and B’s frames are shifted forward by 1 unit to get the global timeline. In general the offsets of different WLAN cards’ clocks are different.

The accuracy of our synchronization technique with respect to any two sniffers can be measured by the difference between the compensated receive timestamps for each frame received by both sniffers. A histogram of these differences for a typical experiment is presented in Figure 4. The x-axis is the difference between compensated receive timestamps of two sniffers, and the y-axis is the normalized frequency distribution. The maximum absolute value of the difference is  $4\mu s$ , which is in line with the time synchronization accuracy of the 802.11b cards (as described in the 802.11b standard).

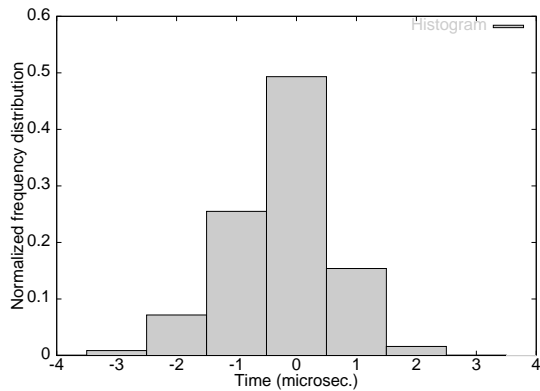


Figure 4: Histogram depicting accuracy of the time synchronization in a typical experiment. The maximum discrepancy between sniffer clocks after compensation is  $4\mu s$ .

This accuracy is sufficient because the transmission time of any 802.11 frame at any transmission bit-rate is more than  $192\mu s$  (because the 192-bit PLCP preamble and header is always transmitted at 1Mbps).

Thus, we construct a global timeline of events by composing different sniffer logs. Henceforth, we refer to this as “global timeline construction” and identifying collisions from the global timeline as “collision analysis”

#### 4 A better PLC model for simulators

In this section we first make some observations from our empirical measurements. Then we use these to point out the scenario where there is a significant difference between predicted throughput imbalances with capture models II and III.

Consider a collision of RTS frames transmitted by two stations A and B at a receiver. Let  $T_A$  denote the time instant when the reception of the RTS frame transmitted by source A starts, and  $T_B$  denote that of source B. Figure 5 shows a histogram of  $T_A - T_B$  for all collisions at the receiver obtained from a typical real-life experiment (described in Section 7). Source A is the stronger, and almost all collisions (99 %) are resolved in its favour. From these results, we make the following observations:

- The stronger frame is almost always retrieved even if it starts second.
- The arrival times of colliding frames routinely differ by as much as  $20\mu s$ , and the frame to arrive first can be either the stronger or the weaker with approximately equal probability.
- RTS frames are also successfully captured.

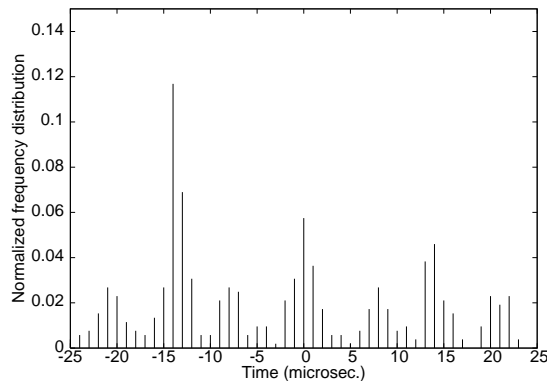


Figure 5: Histogram of the difference between starting time instants of two colliding RTS frames obtained from a typical experiment.

In light of these observations, we identify issues with the PLC model commonly used in simulators. We restrict citing source code to the NS-2 simulator (version 2.26) due to licensing concerns, and briefly discuss Qualnet at the end of this section. Our experimental results have helped uncover two issues in the current modeling of physical layer capture in NS-2.

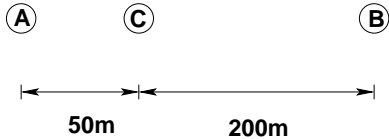


Figure 6: Topology used for NS Experiment

```

// pktRx_ is the first frame
// p is the second frame
if (pktRx_>rxPower/p->rxPower > CPTreshold)
    capture(p);
else
    if (p->rxPower/pktRx_>rxPower > CPTreshold)
        capture(pktRx_);
else
    collision(p);

```

Figure 7: Original code deciding capture in NS-2 simulator with our suggested improvement (in box).

#### 4.1 Issue I: Capture model II vs. capture model III

In a collision of data frames, NS-2 does not display capture of the stronger frame if it starts second, contrary to experimental results. The pseudo-code for NS-2 capture (paraphrased from the file `mac/mac-802.11.cc`) is outlined in Figure 7. Here, `pktRx_` refers to the first frame in the collision and `p` refers to the second frame. If the ratio of `pktRx_`'s signal strength to `p`'s signal strength is greater than `CPTresh` (which is set to 10, or equivalently, a difference of 1 dB), then the routine `capture` is executed.

We now identify a scenario where using the existing model causes significant deviation from reality. We simulate in NS-2 the topology shown in Figure 6. We disabled RTS/CTS to avoid another problem described below. The distances AC and BC are 50 meters and 200 meters, respectively. The workload consists of two equal rate CBR sources, one on each station, with a total rate enough to saturate the available capacity.

Because transmissions are slotted and NS-2 uses one global clock, all the frames in a collision start transmission at the same time. Hence, the time difference between the receptions of the two frames is determined solely by the propagation delay. In our scenario, A's frame arrives at C before B's frame.

**Case 1:** Suppose A and B transmit with the same power. Then A's signal is stronger at C than B's signal at C by more than 1 dB (in NS-2, the signal strength is inversely proportional to a power of the distance, either  $1/d^2$  or  $1/d^4$ ). So the stronger signal arrives before the weaker signal and the existing NS-2 code resolves all collisions in favor of A.

The resulting difference in their long-term throughputs is 14.45%.

**Case 2:** Suppose B transmits with high enough power such that the ratio of signal strength of B at C to that of A at C is higher than the capture threshold of 1dB. Now the stronger signal starts second at C, and therefore, the existing NS-2 code loses both signals in a collision. *NS-2 shows no unfairness in this scenario, contrary to what is observed in reality. It would be more realistic to observe the same unfairness as in case 1, but in B's favor.*

Further, the histogram in Figure 5 shows that it not the propagation delay (which is of the order of nanoseconds) which decides which frame arrives first at the receiver, but rather the inherent uncertainties of clock synchronization (drift, missed beacons, etc.) and rx/tx turnaround time. Therefore, if one were simulating a wireless technology that has capture model II, then our results show that it is crucial to accurately model these timing jitters.

#### 4.2 Issue II : lack of RTS capture

In a collision of RTS frames, NS-2 does not display capture behavior at all (i.e., even if stronger starts first), contrary to experimental results. Suppose two RTS frames collide in NS-2. If the stronger frame is second, then both frames are lost (as discussed in Issue I). If the stronger frame is first, then it is captured (as it should be). However, the NS-2 capture sub-routine (paraphrased in Figure 8) wrongly marks the medium as busy for the duration of the *second frame plus an additional EIFS duration*. Consequently, the CTS response to the first RTS is never sent, effectively resulting in the first RTS frame being dropped. *However, our experiments described in the Section 7 show that when RTS/CTS frames are used, the stronger RTS frame always receives a CTS frame in response.*

```

set_nav(txtime(p) + eifs_);
free(p);

```

Figure 8: Capture sub-routine in NS-2 simulator

Issue I can be fixed by minor modifications to the existing code and this results in an unfairness of 15%, which reflects reality better. Issue II has been taken up with NS-2 maintainers for further debugging.

As mentioned before, Qualnet uses a probabilistic capture model if the stronger frame starts first, and therefore is affected by Issue I. Subsequent to our communication, their support staff have scheduled a fix to their simulator in a future release.

## 5 Equipment and Modifications to Kernel-Space Software

To observe and quantify PHY layer phenomena on a microscopic timescale, it is necessary to record events as they happen at the layers closest to the PHY layer (as opposed to indirect inferences from macroscopic application layer measurements). To this end, we instrumented the lowest layers we had access to: wireless device drivers and the kernel. We first summarize the hardware and software we used and then describe our kernel-space instrumentation in this section.

**Source stations:** For generating traffic, we use up to four identical IBM Thinkpad R-32 laptops, each with 256Mb RAM, 1.2GHz P4 processor, RedHat Linux with kernel 2.4.19, Compaq WL100 WLAN card with firmware version 0.8.0, and an instrumented `linux-wlan` [19] driver.

**Access point and sink:** The Prism2 chipset supports a *HostAP* mode, which allows a laptop with a WLAN card to act as an access point. Time-critical tasks like ACKing and sending beacon messages are implemented in the firmware, and management tasks like client association are implemented in the device driver. We use an IBM R-32 laptop as an Access Point (AP), with an instrumented version of the `host-ap` [20] device driver.

The AP is connected to a sink machine via a 100Mbps Ethernet switch. The sink machine is a P3 750Mhz 512Mb desktop running RedHat Linux with kernel 2.4.19.

**Sniffers:** We use another five IBM Thinkpad R-32 laptops as sniffers. For each source (including the AP), we have a dedicated sniffer placed in such a way that the signal strength of that source at the sniffer is larger than that of all other transmitters at the sniffer. So, in a collision involving a frame transmitted by the source, the sniffer captures the source's frame.

We used a variety of Prism2 [21] chipset cards, namely, LinkSys WPC11, Compaq WL100, and Demartec. Prism2 cards support a *monitor* mode of operation, in which the card captures all frames sent in that frequency. The Prism2 chipset prepends a special header to all sniffed frames, containing fields indicating signal strength, noise level, transmission bit rate, and a timestamp generated by the firmware (using the clock of the WLAN card) at the start of frame reception. The Prism2 firmware also implements a firmware level queue.

We use the `linux-wlan` driver and `tethereal` [22] as the user level process for sniffing. At peak data rates in the WLAN, the socket receive buffer of the `tethereal` program occasionally overflowed when the rate of flushing to the disk wasn't sufficient. So, we tuned the kernel to increase

the limit for the maximum memory limit for the receive buffer of the `SOCK_RAW` socket used by the sniffer program.

**Driver instrumentation:** The `linux-wlan` and `hostap` drivers are instrumented to log events such as packet submission to firmware and firmware's notification of successful transmission or exception (frame dropped after maximum retries). We use the `rdtsc` instruction available in the Pentium family of processors to timestamp events. This instruction returns the number of CPU clock ticks since the machine was last bootstrapped, thereby giving a resolution of inverse of the CPU frequency (1/1.2GHz); it also completely avoids the inaccuracies of using system calls.

We implemented a kernel module to efficiently log driver events in specially allocated kernel memory, thus avoiding the problem of synchronization with other kernel processes. The data is moved from kernel-space to the user-space by using our extension to the Linux `/proc` filesystem. We do not use the standard kernel log buffer because of the number and frequency of events occurring at the driver level and the problem of synchronization with other processes, like `syslogd` and `klogd` daemons, which use the standard kernel buffer.

**TCP tracing:** We tracked the time evolution of the RTT, loss, and congestion window of our TCP sources by using the `NF_HOOKs` [23] provided in the linux kernel. Every time a packet enters or leaves the IP subsystem, a call-back function registered by our module is executed. This routine looks up the parameters of the TCP connection in the associated kernel data structure and logs it in a special area of kernel memory. Again, we use the `/proc` sub-system to move the data from kernel-space to user-space at the end of each experiment.

Logging during experiments is done using in-memory buffer only. The data is written to the disk only after the end of experiment, thus our instrumentation is not intrusive.

## 6 Isolating the unfairness caused by PLC

The goal is to analyze the impact of physical-layer capture effect, all by itself, on 802.11b performance. For this, we consider WLAN networks in adhoc mode in which stations *broadcast* data, i.e., with destination MAC address `FF:FF:FF:FF:FF:FF`. Because there is no unicast, there are no ACKs and hence no timeouts, retransmissions, or binary exponential backoff, and the transmission bit rate is fixed at 2Mbps. Thus all MAC-layer mechanisms are eliminated except for carrier sensing (and the initial backoff). (Adhoc mode is needed because in infrastructure mode, the AP retransmits broadcasts from clients.)

We use two stations, A and B, broadcasting in the adhoc

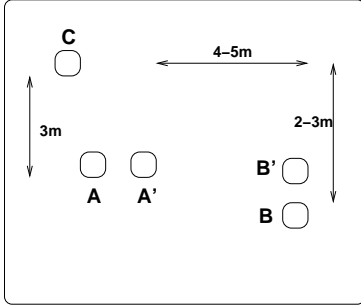


Figure 9: Approximate locations of nodes A, A', B, B', and C. The drawing is not to scale.

mode, and two sniffers, A' and B', one placed close to each source. A third sniffer, C, acts as a sink. We explore two physical configurations. We first look at a configuration designed to maximize the difference between signal strengths of the two stations at the receiver. Later we look at a configuration designed to explore the scenario in which the signal strengths distributions of A and B partially overlap.

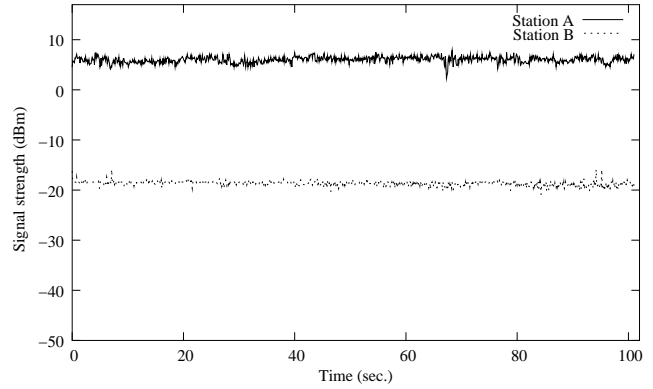
Figure 9 shows positions of sources and sniffers in an indoor setting for the first scenario. We use UDP sources at A and B (because TCP sources cannot broadcast). Each UDP source broadcasts 10,000 data packets at a constant sending rate chosen such that the firmware queue is always non-empty. Each UDP packet is 1472 bytes, this includes a sequence number. This results in a 802.11 frame payload of 1500 bytes (after adding 20-byte IP header and 8-byte UDP header), which is the Ethernet-based MTU used by current WLAN device drivers (although the WLAN standard allows 2304 bytes). Sniffer C logs all of the packets that were successfully received.

To verify that both sources are in range of the sink, we performed experiments in which only one source was sending at a time. In all cases, the sink received all 10,000 packets sent by the source correctly, thus verifying that each source has a perfect channel to the sink. Likewise, we verified that the sources are in range of each other by pinging one from another. Experiments were repeated with several cards, and we observed consistent results.

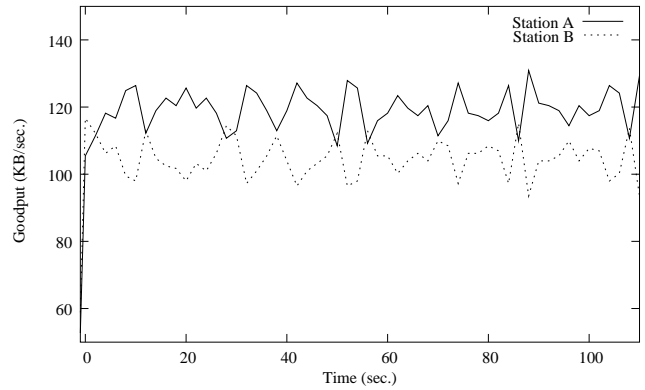
**Metrics:** We use two sets of metrics, one quantifying the extent of the physical-layer capture, and the other which quantifying the impact of the effect on higher-layer performance. For each source, let  $N$  denote the total number of frames transmitted by the source, let  $S$  denote the number of these frames that were not received at the sniffer, and let  $L_i$ , for  $i = 1, \dots, N$ , be an indicator variable that is 1 if frame  $i$  was involved in a collision (according to our collision analysis).

Our metrics that quantify *extent of capture* for each source are:

- **Fraction of transmitted packets that collided:**



(a)



(b)

Figure 10: Received signal strength (in dBm) vs. time (in sec.) (a), and goodput (in KBps) vs. time (in sec.) (b) for a typical ad-hoc mode experiment.

$$F_C = (\sum_i L_i) / N$$

- **Fraction of transmitted packets that were lost:**

$$F_L = S / N$$

Our metrics for estimating the *impact on higher-layer performance* are:

- **Throughput:** the number of “transmission-complete” interrupts over time.

(These interrupts are generated by the WLAN card and logged by the source driver. Because there are no ACKs, a “transmission-complete” interrupt does not imply successful transmission. Hence the throughput metric really quantifies the source’s share of the media access.)

- **Goodput:** the number of frames of the source received by the sink over time.

Station	Signal (dBm)	Num. of packets that			Fraction of pkts that			Goodput (KBps)	Service time (msec.)	Throughput (KBps)
		were sent	were lost	collided	collided ( $F_C$ )	were lost ( $F_L$ )	collided and lost			
A	8.64	10,000	6.23	862.12	8.62	0.06	0.64	<b>116.68</b>	12.54	117.73
B	-19.51	10,000	870.67	862.12	8.62	8.70	100.00	104.20	12.80	<b>117.91</b>

Table 1: Results of ad-hoc mode experiments. Each number is an average computed over several experiments. Fractions have been multiplied by 100. Observe that the average goodput of station A is larger even though its average throughput is nearly the same as that of station B.

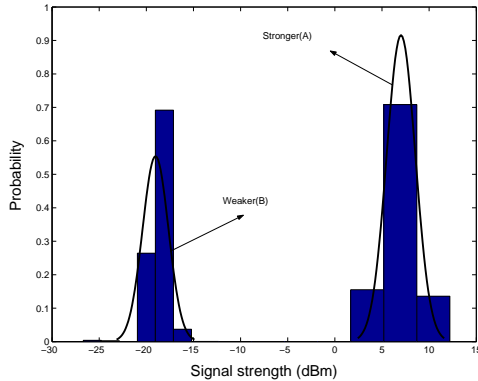


Figure 11: Non-overlapping distributions of signal strengths of A and B at C.

- **MAC Service Time:** The average time between successive transmission-complete interrupts.

(Because the firmware queue is never empty, the MAC service time of a frame is the time from the previous frame's transmission-complete interrupt to this frame's transmission-complete interrupt.)

**Results of the experiments:** Figure 10a represents the time evolution of the signal strength of the two source stations as perceived by the sink sniffer in a typical experiment. Each point in the figure is the averaged value of signal strengths of 50 frames around the frame received at that time instant. Figure 11 shows the corresponding histogram.

Figure 10b depicts the goodputs achieved by both sources during one of the experiments over time. The stronger source, A, has higher goodput than the weaker source, B. The overall goodput (computed over the entire experiment) for the stronger source was typically **12% higher** than that of the weaker source.

The time evolution of the instantaneous throughputs is shown in Figure 12. The difference in the instantaneous throughputs varies on the short-term between the two sources, while the overall throughputs differ by **less than 1%**. The service times for the two sources (Table 1) differ by **lesser than 2%**.

Because there are no ACKs and retransmissions, the throughput seen by the source represents the rate of media-

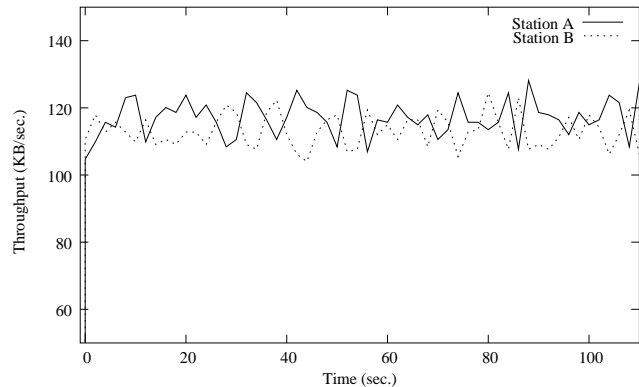


Figure 12: Throughput (KB/s) vs. time (sec.). Note, that throughput of both stations is nearly equal.

access. Because there is PLC and subsequent loss for the weaker station, the goodputs seen at the sink sniffer differ.

Table 1 presents the results of our collision and loss analysis of the experiments.  $F_C$  and  $F_L$  are very close (differ by 0.9%) for the weaker source showing that most of the frames that collided were lost by the weaker source, while  $F_L$  of the stronger source is close to zero. Nearly all of the collisions (99%) were resolved in favor of the stronger source. The average goodput of station A is larger, even though its average throughput is nearly the same as that of station B.

About 44% of the cases the stronger source started second. The packets of the weaker source that were lost are almost always (99%) the ones for which our global-timeline analysis detected collision.

### Impact of overlapping signal strengths:

In order to observe what happens when the distributions of received signal strengths are not clearly separated, we adjusted the positions of nodes A and B such that the ranges over which the signals vary overlap partially at the receiver C. We then repeated the experiments described above. Figure 13 shows the resulting distribution of the signal strengths of the two stations at the sink C and a Gaussian curve fitted using MATLAB. For comparison, Figure 11

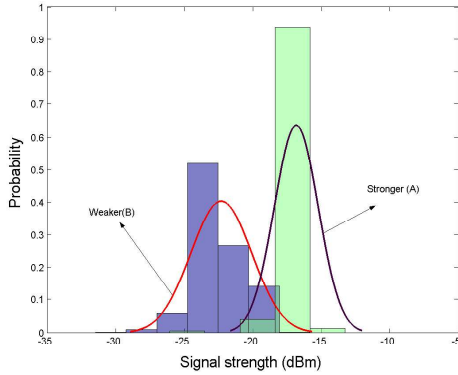


Figure 13: Overlapping distributions of signal strengths of A and B at C

shows the signal strength distributions which do not overlap (obtained from one of the previous ad-hoc mode experiments).

From loss analysis and global collision analysis, we find that 81% of the frames which collided were resolved in favor of the stronger source and both sources lost close to 20% of frames that collided. This shows that PLC happens even when signal strengths distributions from two sources are not completely separated from each other provided there are some collisions that happen with frames having higher strength than those that collide with them.

### Summary:

- In about 44% of the collisions, the stronger frame started second but was still captured by the receiver.
- Nearly 100% of collisions were resolved in favor of the station with the stronger signal.
- The resulting long-term goodput imbalance between the two sources was as high as 12%.
- The resulting long-term throughput imbalance between the two sources was practically zero.
- In case of overlapping signal strengths, 81% of the colliding frames were resolved in favor of the stronger source, with remaining 20% being lost.

## 7 How higher layers magnify the unfairness

The second round of experiments was designed to quantify the amount by which higher layers magnify unfairness caused by PLC. We present detailed results and analysis for two sources (A and B), and summarize the results for more sources.

**Experiments conducted:** All experiments were conducted

in a typical indoor WLAN environment, with access point AP, sources A, B, C, D, and sniffers AP', A', B', C', D', positioned approximately as shown in Figure 14. There were two sources close to the AP in the same room, and two other sources farther away in different rooms. There was no other RF transmission before, after, and during the experiment (this was confirmed by sniffing on the channel).

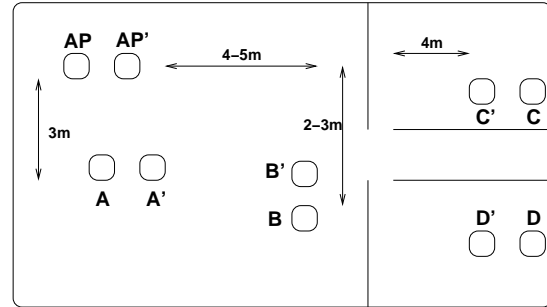


Figure 14: Approximate locations of nodes in infrastructure mode experiments. X' is the sniffer for station X. The drawing is not to scale.

We verified that all stations could hear each other by pinging them in ad-hoc mode. We confirmed that a perfect bi-directional channel was available from each station to the AP by running `netperf` on that station alone. Each station attained the same application throughput of 3.6 Mbps. For the UDP source experiments, each station sent 10,000 MTU-sized packets (not including MAC-layer retransmissions). The sending rate was chosen such that the firmware-level queue on each station was never empty during the course of experiment. For the TCP experiments, each station ran `netperf` [24] for 100 seconds.

**Metrics for UDP experiments:** For a source, let  $N$  be the total number of RTS frames transmitted by the source, and let  $M$  be the total number of CTS frames received by the source. Each CTS frame corresponds to an RTS frame that was received. Thus  $N \geq M$ , and  $N - M$  is the number of RTS frames that were transmitted and lost.  $N$  and  $M$  can be obtained from the log of the sniffer associated with the source.

For a source, our metrics for measuring the *extent of capture* are:

- **Fraction of RTS losses:**

$$F_L = (N - M)/N$$

- **Fraction of RTS collisions:** Let  $L_i$ , for  $i = 1, \dots, N$ , be an indicator variable, which is 1 if the RTS frame  $i$  is detected as having collided by our global collision analysis and 0 otherwise.

$$F_C = \sum_i L_i / N$$

Protocol	Station	Signal (dBm)	Fraction of packets that				RTT (msec.)	Goodput (KBps)				Goodput difference (%)
			collided ( $F_C$ )		were lost ( $F_L$ )			link layer		appl. layer		
			DATA	RTS	DATA	RTS		mean	variance	mean	variance	
UDP	A	8.72	0.01	4.78	0.01	0.48	N/A	347.23	3.88	330.45	3.72	<b>18.70</b>
	B	-19.53	0.01	4.82	0.02	4.92	N/A	292.45	1.67	278.56	1.57	
TCP	A	8.82	0.01	4.69	0.01	0.38	106.76	245.52	3.26	234.89	3.12	<b>25.13</b>
	B	-19.44	0.02	4.78	0.02	4.79	136.65	195.37	2.85	187.53	2.73	

Table 2: Analysis of collisions, losses and achieved goodputs during UDP and TCP experiments in the infrastructure mode with RTS/CTS enabled. Each value is an average over several experiments and fractions have been multiplied by 100. Signal strength is measured at the access point.

Protocol	Transmission 1	Transmission 2	% lost by both	% won by transmission 1	% won by transmission 2
UDP	A → AP	B → AP	4.54	95.46	0.00
TCP	AP → A	A → AP	100.00	0.00	0.00
	AP → A	B → AP	5.16	94.84	0.00
	AP → B	A → AP	15.59	84.41	0.00
	AP → B	B → AP	100.00	0.00	0.00
	A → AP	B → AP	5.68	93.82	0.50

Table 3: Analysis of collision resolution for UDP and TCP experiments in infrastructure mode with RTS/CTS. Each value is an average over several experiments. A→B denotes A sends to B.

Our metrics for measuring the *impact on higher layer performance* are:

- **Link Layer Goodput:** The number of unique IP-level frames successfully transmitted per unit time. This is obtained from the device-driver log of the source by counting the number of “successful transmission” interrupts over time. Because ACKs are used, an interrupt showing “successful transmission” indicates that the frame got through.
- **MAC Service Time:** This is obtained from the device-driver in the same manner as that for the ad-hoc mode experiments. In this case however, this includes the time durations due to retransmissions and binary exponential backoff.

**Results of UDP experiments:** The time evolution of the signal strength remains the same as the Figure 10a for ad-hoc experiments. Figure 15 shows the link-layer goodput of the stations over time.

The stronger source finishes first for the same number of packets. The overall goodput of the stronger source is **18.7%** higher. The average MAC service time of the weaker source is 4.96 ms and that of the stronger is 4.19ms, a difference of **18.37%**.

Tables 2 and 3 present the detailed results of the experiments. The values in the tables were obtained by averaging results of multiple experiments. The variation of each metric is very small. Nearly 5% of the RTS frames sent by each

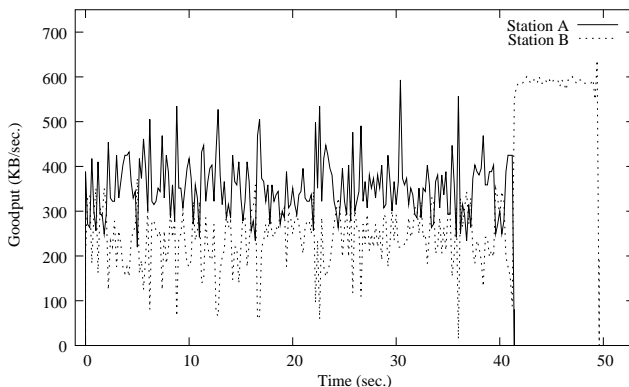


Figure 15: Link-layer goodput (in KBps) vs time (sec) of two sources in a typical UDP expt in infrastructure mode with RTS/CTS.

source collided at the AP. 95.46% of the collisions were resolved in favor of station A. In about 53% of the collisions, the stronger station, A, started sending later than B. The data from one of these experiments was presented as a histogram on Figure 5 in Section 4.  $F_C$  was close to  $F_L$  for the weaker station, showing that it lost most of the collisions.

**Metrics for TCP experiments:** We use the following to quantify the impact of PLC on TCP performance:

- Congestion Window
- RTT as observed by TCP

- TCP level loss
- TCP level throughput, as reported by `net-perf`.
- Firmware Service Time (instead of MAC Service time used for UDP)

Firmware service time is the time between submission of frame to the firmware and the successful completion indication. This includes the queuing delay involved in the firmware and is more representative of the end-to-end delay that affects TCP performance.

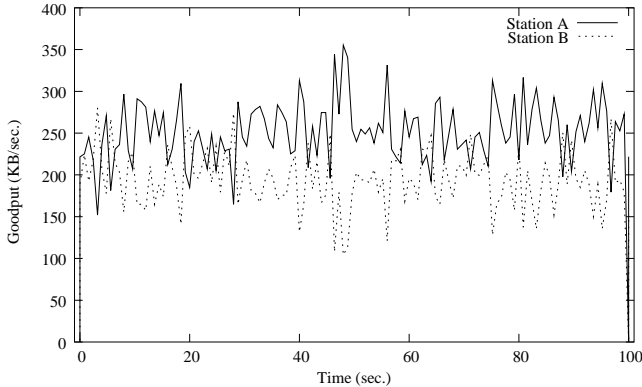


Figure 16: Link-layer goodput vs time of two stations during a typical TCP experiment in the infrastructure mode with RTS/CTS.

All our metrics involving TCP state were obtained by the kernel instrumentation summarized in Section 5.

**Results of TCP experiments:** The signal strengths of the sources at the AP is the same as that in Figure 10a. The time evolution of the link-layer goodputs is shown in Figure 16. All numbers are averages over several experiments. The difference in the long-term link-layer goodput is **29%** and the application layer throughput (reported by `netperf`) difference is **28.57%**.

After slow start, the congestion windows of both TCP sources source were at 23 frames (corresponding to 32K advertised receiver buffer) throughout the experiments, except for the very few losses (0.02%) for the weaker station. This implies that the throughputs obtained by the application is effectively determined by the RTT.

The firmware service times of the sources are shown in Figure 17. The X-coordinate of a point in the graph represents the transmission completion time instant of a frame and the Y-coordinate represents the firmware service time of that frame.

The weaker source has higher instantaneous service time (most of the time) and the difference in the average per-frame firmware service time is **27%**.

Figure 18 shows the time-variation of the RTT values seen by TCP. A point in the graph denotes the RTT value

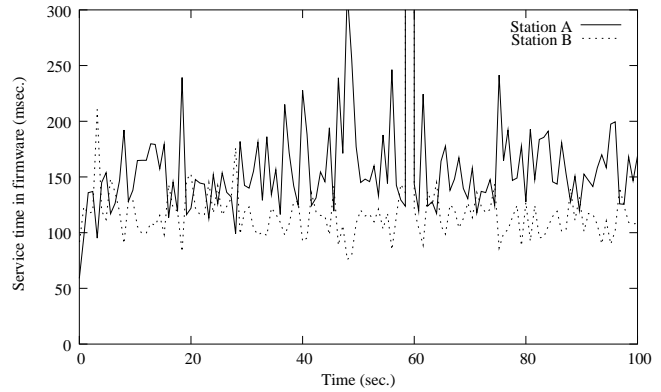


Figure 17: Time evolution of the firmware service time (including queuing) for a TCP experiment (with RTS/CTS enabled).

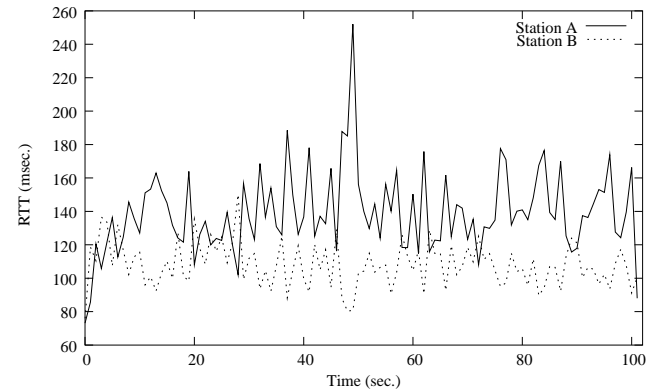
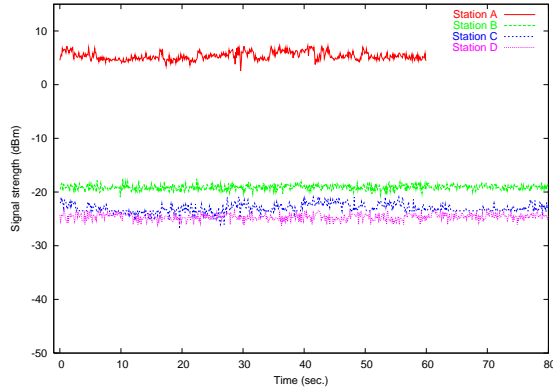


Figure 18: TCP's RTT of both stations in the experiment with RTS/CTS.

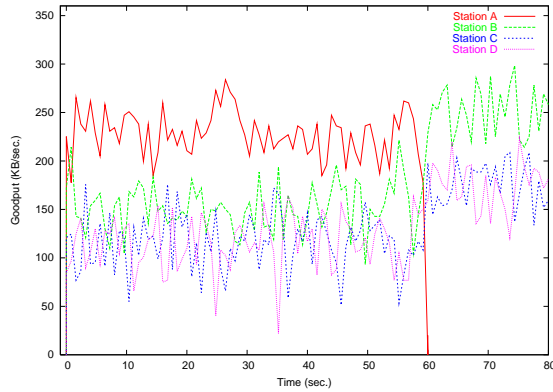
in the Y-axis and the time instant the information was obtained in the X-axis. The weaker source has higher RTT in general and the average RTT of the weaker source is higher by 27.9%. Table 2 summarizes the results.

Table 3 presents the detailed collision analysis of all pairs of simultaneous transmissions at various receivers. The results show that collisions are resolved in favor of the stronger source 93.82% of the time at the AP. The stronger source also collides with transmissions from AP to the weaker source and affect the weaker source's reception 15.59% of the time, while the weaker affects the AP's transmission to the stronger only 5.16% of the time.

**Statistical analysis:** While we expect a frame which was retransmitted to have a higher service time, it is not guaranteed due to the random choices of back-off durations. Therefore, we quantify the relation between the service time and the number of transmissions using Pearson's correlation co-efficient [25]. The result gives an average correlation co-efficient of **0.41**. This shows that the service time is *positively* correlated with the number of transmissions the frame



(a)



(b)

Figure 19: Time evolution of signal strengths at AP (a), and link-layer goodput of stations for a typical 4 station UDP experiment (b) (with RTS/CTS enabled).

underwent. The correlation coefficient is not very close to 1 due to the inherent randomness involved in the queuing delay and choice of backoff intervals. This implies that, in most of the cases, frames that experienced higher service time in the firmware are the ones that collided and thus were retransmitted.

**Results with multiple sources:** The time evolution of the goodputs with 4 UDP sources is shown in Figure 19b and the signal strengths are shown in Figure 19a.

(These figures have been shown in color to aid distinguishing between the curves.) The time evolution of link-layer goodputs with 4 TCP sources is shown in Figure 20.

The UDP curves show the strongest source finishing first for the same number of packets. The TCP curves also show the strongest source's goodput being higher than those of the rest. The results are summarized in Table 5.

In order to quantify the extent of unfairness, we compute

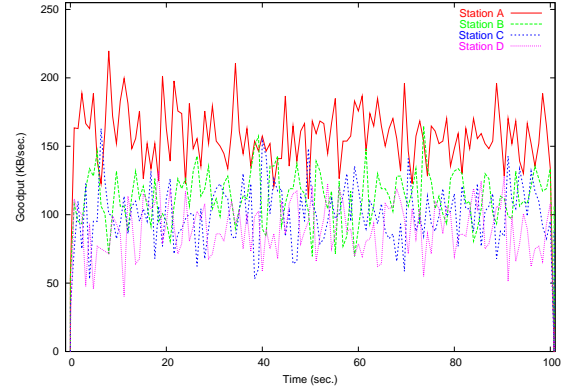


Figure 20: Time evolution of the link-layer goodput of all stations for a typical 4 station TCP experiment (with RTS/CTS enabled).

three metrics, namely Jain's Fairness Index [26], the ratio between the maximum and minimum goodput, and coefficient of variance (i.e., variance/mean). The first two metrics increase with increasing fairness, while the last decreases. (Let  $x_i$  denote the goodput obtained by source  $i$  and  $n$  be the number of sources. Jain's Fairness Index  $F$  is defined as  $\frac{(\sum_i x_i)^2}{n \times \sum_i x_i^2}$ . If all sources achieve the same goodput, then  $F = 1$ , else  $0 < F < 1$ , higher value indicating greater fairness.) The results are shown in Table 4. The low min/max values shows severe unfairness between the best and the worst stations. Other metrics are also significantly away from the ideal (1 for Jain and 0 for Coefficient of variance) values showing the extent of unfairness. The results without the use of RTS/CTS are discussed in the next section.

**Summary:** The results of the experiments presented above confirm that in the infrastructure mode (with binary exponential backoff, ACKs, and retransmissions):

- Collisions are resolved in favor of the stronger station.
- RTS frames are successfully captured; the stronger station gets the CTS frame.
- In approximately 50% of the cases the stronger RTS frame starts second but is still received properly.
- MAC layer magnifies the discrepancy (due to binary exponential backoff) by approximately 10%.
- TCP may additionally magnify the difference. In our case it does not, because there are no TCP-layer losses as link-layer losses are masked by retransmissions. Therefore, unfairness seen at the MAC-layer propagates to the application-layer unchanged.
- Observed throughput difference at the application layer may be as high as 25% for two sources and up to 75% with four sources.

Fairness index	with RTS/CTS		w/o RTS/CTS	
	UDP	TCP	UDP	TCP
Jain's index	0.91	0.95	0.77	0.90
Min/Max ratio	0.50	0.57	0.28	0.44
Coeff. of variance	0.30	0.22	0.53	0.32

Table 4: Metrics for quantifying fairness in goodputs for 4 sources

Protocol	Station	Goodput (KBps) with RTS/CTS			
		enabled		disabled	
		mean	variance	mean	variance
UDP	A	228.91	3.98	373.06	9.32
	B	150.06	2.67	207.39	6.81
	C	116.53	2.32	117.49	2.10
	D	115.21	3.40	105.08	2.11
TCP	A	158.27	3.16	226.57	5.19
	B	113.26	3.39	184.58	3.68
	C	101.08	5.05	120.75	2.78
	D	90.20	5.38	100.99	4.87

Table 5: Goodputs for multi-source experiments with UDP and TCP with RTS/CTS and with RTS/CTS disabled.

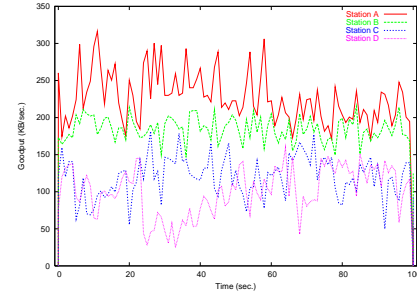
## 8 Infrastructure mode experiments without RTS/CTS

The final round of experiments was designed to quantify the impact of PLC on WLANs that do not use RTS/CTS. We expect the number of data frame collisions to increase without the use of RTS/CTS, and therefore, the effect of unfairness to be more pronounced. (Existing 802.11 WLAN cards and APs do not use RTS/CTS by default; it has to be specially enabled by the user/administrator.)

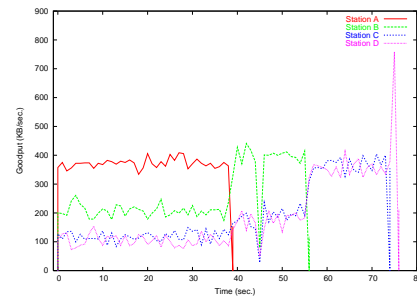
Experiments conducted involve four mobile sources generating UDP and TCP traffic. The experimental setup is identical to the one used in four source experiments with RTS/CTS enabled. Owing to lack of space, we briefly summarize our results for four sources.

Figure 21b shows the goodputs of 4 UDP sources. The sources finish sending the same number of packets (10,000) in order determined by the signal strength, and the strongest source has the highest goodput. Figure 21a shows the goodputs attained when each TCP source sent data for 100s.

Table 4 shows metrics for fairness with 4 sources. Table 5 summarizes the goodputs attained by each source. The metrics used show much bigger unfairness for workloads which didn't use RTS/CTS. Thus, the use of RTS/CTS increases fairness, but does not bring it close to ideal because of PLC.



(a)



(b)

Figure 21: Goodput vs. time for a typical TCP experiment with 4 sources and no RTS/CTS (a), and a typical UDP experiment (b).

## 9 Conclusions and future work

Existing models for modeling collisions in 802.11 networks typically assume that either both colliding frames are lost or that the stronger frame is received successfully only if it starts first. In this work, we showed experimentally that in reality the stronger frame is retrieved first irrespective of whether its reception starts first or second.

This result has significant impact on the modeling of the physical layer in discrete event simulators. Specifically, we have uncovered two issues in the NS-2 simulator. The Qualnet simulator will also be modified to account for our results.

Finally, we examined the causal link between physical layer capture and application layer unfairness in great detail. This quantifies the impact of the physical layer capture on the perceived application layer unfairness by itself and examined how it is magnified by the MAC and higher layer control mechanisms.

Directions for future work include the design of control mechanisms at the MAC layer to compensate for the unfairness induced by the physical layer.

## References

- [1] M.Soroushnejad and E. Geraniotis, "Probability of Capture and Rejection of Primary Multiple Access Interference in Spread Spectrum Networks," *IEEE Transactions on Communications*, vol. 39, no. 6, pp 986-994, 1991.
- [2] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proceedings of IEEE INFOCOM 2003*, San Francisco, USA, March-April 2003.
- [3] J. Bianchi, "Performance analysis of the IEEE 802.11 Distributed Coordination Function.," in *IEEE Journal On Selected Areas in Communications*, March 2000.
- [4] K.C. Tay, Y.C. and Chua, "A Capacity analysis for the IEEE 802.11 MAC protocol," in *Wireless Networks*, January 2001.
- [5] Cali I., Conti M., and Gregori E., "IEEE 802.11 wireless lan: Capacity analysis and protocol enhancement," in *Proceedings of the IEEE INFOCOM*, 1998.
- [6] "NS-2 network simulator," <http://www.isi.edu/nsnam/ns>.
- [7] "Qualnet network simulator," <http://www.scalable-networks.com>.
- [8] The Institute of Electrical and Electronic Engineers Inc., "IEEE 802.11, 1999 edition (ISO/IEC 8802-11:1999)," <http://standards.ieee.org/getieee802/802.11.html>.
- [9] *Personal communication with Qualnet support staff*.
- [10] B. Bing, "Measured performance of the ieee 802.11 wireless lan," in *24th Conference on Local Computer Networks*, October 1999.
- [11] G. Xylomenos and G. Polyzos, "Tcp and udp performance over a wireless lan," in *IEEE Infocom*, 1999.
- [12] K. Cheun and S. Kim, "Joint delay-power capture in spread-spectrum packet radio networks," in *IEEE Transactions on Communications*, 1998.
- [13] D.H. Davis and S. Gronemeyer, "Performance of slotted aloha random access with delay capture and randomized time of arrival," in *IEEE Transactions on Communications*, 1980.
- [14] J.C. Arnbak, "Capacity of slotted aloha in rayleigh fading channels," in *IEEE Journal On Selected Areas in Communications*, February 1987.
- [15] US Patent 5987033, *Wireless LAN with Enhanced Capture Provision*, <http://www.uspto.gov>, November 1999.
- [16] C.Ware, J.F.Chicharo, and T.Wysocki, "Modelling of capture behaviour in ieee 802.11 radio modems," in *Invited Paper, IEEE International Conference on Telecommunications*, June 2001.
- [17] B. Rathke, M. Schlager, and A. Wolisz, "Systematic measurement of tcp performance over wireless lans," in *Technical Report, Technical University Berlin TKN01BR98*, 1998.
- [18] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz, "Unfairness and capture behaviour in 802.11 adhoc networks," in *IEEE International Conference on Communications*, 2000.
- [19] "Linux Wlan driver," <http://www.linux-wlan.com>.
- [20] "HostAP driver," <http://hostap.epitest.fi>.
- [21] "Prism chipset documentation," <http://www.intersil.com/design/prism>.
- [22] "Ethereal packet analyzer," <http://www.ethereal.com>.
- [23] "Netfilter documentation," <http://www.netfilter.org>.
- [24] "Netperf performance evaluation tool," <http://www.netperf.org>.
- [25] John E. Freund, *Mathematical Statistics*, Prentice Hall, 1999.
- [26] R.Jain, *The Art of Computer System Performance Analysis*, John Wiley and Sons, 1991.