Improving 802.11 Range with Forward Error Correction

Reina E. Riemann, Keith J. Winstein

{*riemann,keithw*}*@mit.edu*

Computer Science and Artificial Intelligence Laboratory, Massachussets Institute of Technology

Abstract

The ISO/IEC 8802-11:1999(E) specification[1] uses a 32-bit CRC for error detection and whole-packet retransmissions for recovery. In long-distance or high-interference links where the probability of a bit error is high, this strategy results in excessive losses, because any erroneous bit causes an entire packet to be discarded. By ignoring the CRC and adding redundancy to 802.11 payloads in software, we achieved substantially reduced loss rates on indoor and outdoor long-distance links and extended line-of-sight range outdoors by 70 percent.

1 Introduction

Bit errors are a rare phenomenon in most wired local area networks. Wired Ethernet[2] rarely operates close to the noise threshold — link distances are limited by restrictions on delay, not by signal power. See Figure 1. As such, the cause of misbehavior in wired Ethernet is typically congestion, not noise, and the type of error usually manifested is the loss of an entire packet, not a flipped bit.

Wireless networks are different. The range of a 32 mW 802.11 base station indoors is only a few tens of feet before bit errors begin to appear. Outdoors, there is much interest in stretching 802.11 links as far as possible, including MIT's own Roofnet project[3] to build a citywide 802.11 mesh network in Cambridge, Massachusetts.

[1]ANSI/IEEE Std 802.11, 1999 edition.

[2]Formally, ISO/IEC 8802-3:2000(E) or IEEE Std 802.3, 2000 edition.

[3]`http://www.pdos.lcs.mit.edu/roofnet/`

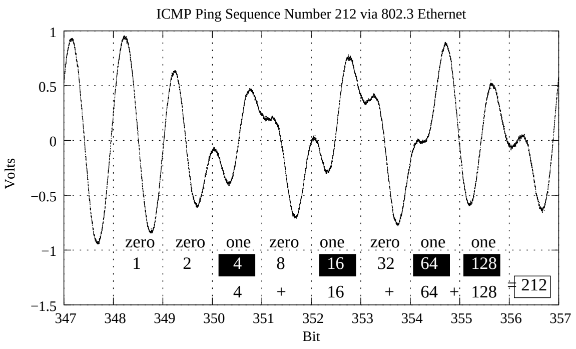

Figure 1: An example 802.3 waveform at 10 Mbit/sec (the sequence number "212" in an ICMP ping packet). Up-transitions in the middle of bit intervals indicate ones, down-transitions zeros. Note the almost complete lack of noise.

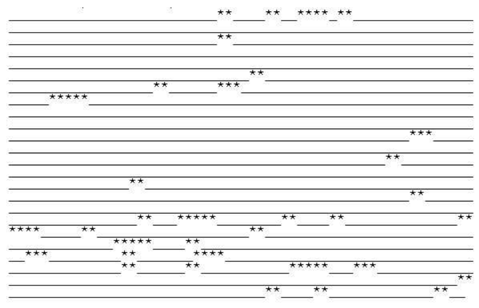

Figure 2: An example 802.11 packet at 1 Mbit/sec, received over a 0.55-mile outdoor line-of-sight link. This packet has 90 octets received incorrectly (indicated with *) out of 1,392.

The causes of misbehavior on an 802.11 link thus include, in addition to congestion, low signal-to-noise ratio and interference. These latter two produce incorrectly received bits, causing a failed CRC, a lack of acknowledgement, and the sender's retransmitting the entire packet. (Low signal-to-noise and interference can also cause a receiver not to hear the sender's preamble and miss the packet entirely.)

Figure 2 shows an example 802.11 payload after transmission 2,900 feet along an outdoor line-of-sight link. The payload was 1,392 octets, of which 1,301 were received correctly and 90 incorrectly (indicated with `*`). At this error rate, most 802.11 senders will give up on retransmissions before the receiver is able to receive an entire packet correctly.

With some fraction of each packet devoted to forward error correction, the receiver can recover from bit errors. By adding redundancy to each packet in software — essentially, increasing the noise margin of the transmission path by reducing throughput — we were able to extend the outdoor range of 802.11 by up to 70 percent, from 0.5 miles to 0.85 miles.

2 Related Work

Many papers have presented strategies to improve the performance of reliable transports, such as TCP, over wireless networks. In [1], Balakrishnan *et al.* compare and analyze several techniques for improving TCP throughput over a precursor to 802.11, including modifications to TCP endpoints such as selective acknowledgements, and making the link-layer protocol reliable and "TCP-aware," with immediate retransmission of lost packets. The authors were not able to obtain the contents of errored frames[4] and did not analyze forward error correction in this context.

Khayam *at al.* ([2]) measured the error characteristics of two indoor one-way 802.11 links across a hallway at 2, 5.5, and 11 Mbits/sec. Because their links were one-way (for multicast video broadcast), they used forward error correction to recover from dropped packets as well as bit errors.[5] The authors found that, for these two links operating at 5.5 Mbits/sec, giving the application layer access to frames received with bit errors reduced by about 40 percent the amount of redundant information necessary to achieve perfect transmission. At 11 Mbits/sec, there was no advantage to looking at frames received with bit errors, because the vast majority of errors were long strings of dropped packets, not frames received with bit errors. Their two links had almost no dropped packets or errored octets at 2 Mbits/sec. The Khayam authors did not try to extend 802.11's range or discuss two-way connections.

3 Measurements

In order to design a reasonable error-correction strategy for 802.11 packets with bit errors, we sought to observe the error characteristics of several indoor and outdoor environments. Unfortunately, most consumer 802.11 implementations do not advertise an interface that allows the host to receive frames with failed CRC. To our knowledge, only cards based on the Intersil[6] Prism 2/2.5/3 chipset have a publicly known method to access errored frames.

We modified the GNU/Linux Host AP driver[7] for these cards to ignore the card's report of a failed CRC and put the card in "`HFA384X_RID_PROMISCUOUSMODE`." These two changes were necessary to receive errored frames outside of the card's "monitor mode."

Because our goal is to improve 802.11's range, we conducted all measurements at 1 Mbit/sec, the slowest and most noise-resilient 802.11 modulation. We set up a laptop with a 32 mW Lucent Orinoco card on top of MIT's Green Building, about 310 feet above ground level. The laptop transmitted 1,400-byte ICMP ping packets to the Ethernet broadcast address. We wanted there to be no link-layer retransmissions, because they don't make sense when we are trying to salvage errored frames: the sender will keep

[4]Balakrishnan, personal communication, November 2003.

[5]When a two-way connection is available and latency is not important, forward error correction is not useful for dealing with dropped packets. The ideal strategy is acknowledgement and retransmission of the dropped packet, because the redundantly-transmitted information is exactly what was lost.

[6]Now sold by GlobespanVirata, Inc.

[7]By Jouni Malinen. `http://hostap.epitest.fi/`.

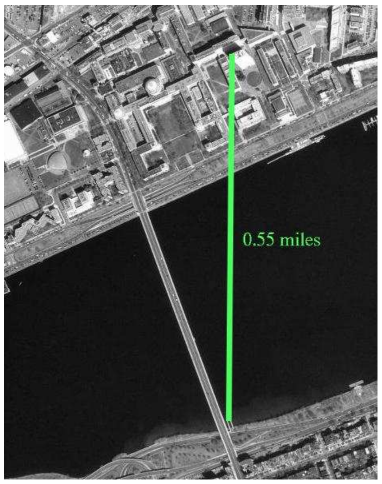

Figure 3: The 0.55-mile marginal link across the Charles River. Courtesy MassGIS and the MIT Department of Urban Studies and Planning.

trying to retransmit until the receiver gets the frame perfectly. This is not what we want.

We attempted to receive the pings at various distances with a Prism 2 card in another laptop, using a Global Positioning System receiver to mark our location. The links we tested were line-of-sight over the Charles River, which separates Boston and Cambridge.

We made three measurements: (1) *How far is a line-of-sight outdoor link functional without receiving bit errors?* (2) *How far out can a line-of-sight outdoor link consistently receive packets, with or without bit errors?* and (3) *What is the distribution of bit errors per packet on an example long-distance link?*[8]

The difference between (2) and (1) is essentially the "marginal zone": the increase in range we expect to receive by using forward error correction.

We made measurement (3) by taking just over three minutes of data on the Boston bank of the river, 0.55 miles away from the transmitter. See Figure 3.

[8]We acknowledge that these definitions are subjective, and should not substitute for rigorous link-quality statistics. How many successful transmissions must a link receive to be "functional"? What percentage of packets must be received before we can say our link does so "consistently"?

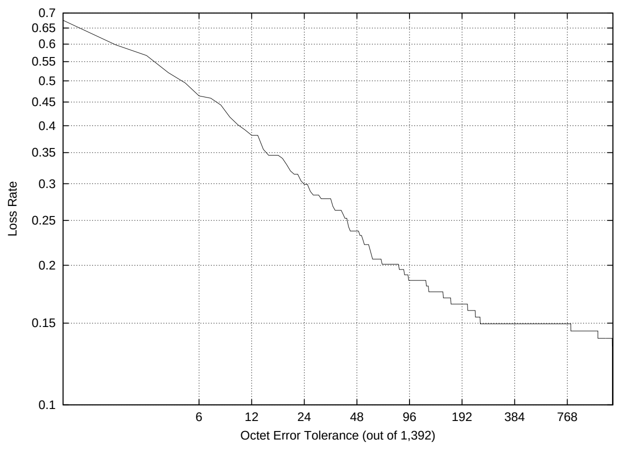

Figure 4: Fraction of packets lost versus error tolerance on the 0.55-mile one-way link. Note log-log scale.

4 Results

The distance in measurement (1) — the maximum distance for zero-error reception — was 0.50 miles.[9] The distance in measurement (2) — the maximum distance at which we could get packets at all — was 0.87 miles.

The "marginal zone" is about 0.37 miles, or a 70 percent increase in distance if we can recover from received bit errors.

The measurements on the 0.55-mile link were as follows:

1. 195 pings were transmitted, each with a 1,392-byte known payload.

2. 23 packets, or 12 percent of those sent, were lost entirely.

3. 63 received packets, or 32 percent of those sent, were received perfectly.

[9]More rigorous measurements are needed here, along with signal strength. The zone of perfect reception dropped out conclusively at 0.50 miles, but seemed to reappear briefly at one spot 0.70 miles away. Was this the result of a coherence of lucky bounces, a low-noise spot, or what? Unfortunately, the Prism 2 card does not produce any measurement of signal strength at these long distances.

4. 10 received packets, or 5 percent of those sent, had 192 or more octets incorrect.

5. The remaining 99 packets, or 51 percent of those sent, had between 0 and 192 octets incorrect.

Figure 4 presents these data in graphical form: the fraction of packets with more than a given number of octets incorrect. Normal 802.11, which can only make use of perfectly-received packets, is at the left-hand side of the graph — a loss rate of 100 - 32 = 68 percent.[10] If we can salvage packets with up to 192 incorrect octets, we would instead have a loss rate of 100 - (32 + 51) = 17 percent.[11]

These results — a 70 percent increase in distance, a 75 percent decrease in packet loss rate — convince us that adding redundancy so that the receiver can salvage errored packets can be a useful software-only addition to 802.11.

5 Implementation

Under GNU/Linux, we implemented an "RSenc" device that encodes outgoing packets by adding redundancy, and corrects errors in incoming packets. The implementation in user space, with the Linux TUN/TAP (user space network devices) module.

We use Reed-Solomon error correction at the byte level, with a transmitted codeword of 255 bytes. Of these, some are the "message bytes" (the input to the coder) and the rest are "redundancy bytes." Every two redundancy bytes we add decreases our message capacity by two bytes, but it adds one byte of error tolerance. That is, with two redundancy bytes and 253 message bytes, any one byte in the entire 255-byte received codeword may be corrupted arbitrarily in flight, and the decoder will still be able to recover the original message bytes.

[10]Note that most 802.11 implementations retransmit four times, so for non-broadcast traffic the loss rate that normal 802.11 users would experience in practice is closer to $0.68^4 = 0.21$.

[11]Why all the talk of incorrect octets, instead of bit errors? 802.11 at 1 Mbit/sec uses differential binary phase shift keying (DBPSK), where each bit gets its own modulated symbol, so there does not seem to be a theoretical reason to expect errors to come in octets. Nevertheless, the errors we observe in practice generally corrupt a whole octet or a string of octets, not just scattered bits. See Figure 2.

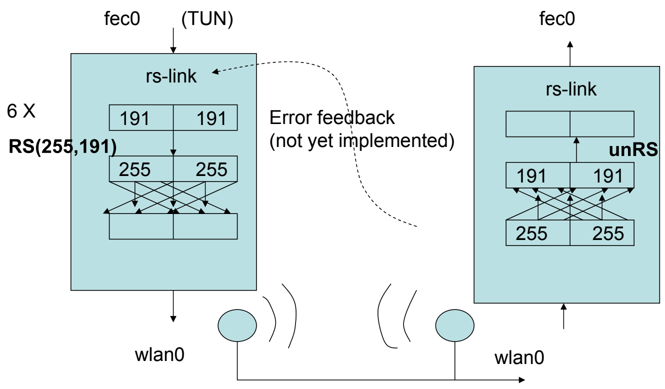

Figure 5: Structure of rs-link implementation, including a planned addition: feedback as to the appropriate amount of redundancy to add.

Based on our measurements of the 0.55-mile link, we chose a code that has 191 message bytes, and 64 redundancy bytes, in each 255-byte codeword (known as an $RS(255, 191)$ code).

The RSenc device accepts packets of up to 1,146 bytes ($= 6 \cdot 191$). It splits them into blocks of 191 bytes and appends the 64-byte redundancy information to each block, making a codeword of 255 bytes, for a total of $6 \cdot 255 = 1{,}530$ bytes in each output packet.[12]

Our measurements also indicated that errors tend to occur in bursts of several bytes. We must drop a packet if any one of its Reed-Solomon codewords has more than 32 bytes wrong, so it is to our advantage to try to spread those errors out. So before transmission, we scramble the message by interleaving each codeword in a round-robin manner.[13] We then send each 1,530-byte encoded packet to the 802.11 card.

On receipt, we reverse the procedure: unscramble, then rectify each 255-byte Reed-Solomon codeword, extract the 191 message bytes from each, and deliver

[12]We have to increase the MTU of the wireless card to this number, from the default of 1,500 bytes.

[13]We leave the Ethernet header unscrambled so the card knows we are sending to the broadcast address and turns off link-layer retransmissions.

the resulting packet to the operating system.

The 802.11 card itself must be in “ad hoc” mode and can only communicate with other “ad hoc” 802.11 stations. (When the card is in “managed” mode, it disassociates from its access point when it starts to receive errors.)

6 Tests

We tested the rs-link software indoors, on a link through several walls, between two laptops with Prism 2 cards. The link distance was picked to be in the “marginal zone” where each station could receive packets from the other, but with errors.[14]

Unlike the measurements in Figure 4, this was a two-way test over our error-corrected link. One laptop sent 105 ICMP ping queries to the Ethernet broadcast address and tallied the responses.

The measurements were as follows:

1. 105 pings were transmitted, each with a 1,146 known contents.
2. 31 pings, or 30 percent of those transmitted, did not receive a response.
3. 8 pings, or 8 percent of those transmitted, received a perfect response.
4. 10 pings, or 10 percent of those sent, received a response that the rs-link decoder could not salvage (i.e., more than 192 octets incorrect).
5. The remaining 66 pings, or 63 percent of those sent, received a an errored response that the rs-link decoder was able to salvage.

These data are presented graphically in Figure 6. Our two-way loss rate is at the right-hand side of the figure — 192 errored octets permitted. The two-way loss rate for normal 802.11 (without link-layer retransmissions) is at the left-hand side of the figure – no errored octets permitted.

[14]Most transmission problems were in one direction, because one card, the “access point,” was operating at 200 mW and the other at 32 mW.

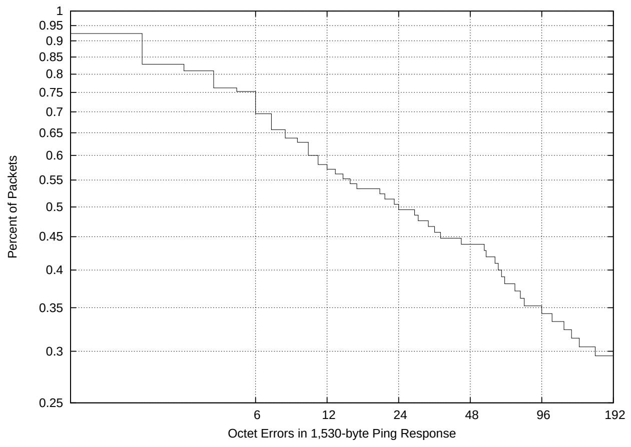

Figure 6: Fraction of two-way pings lost as a function of octet error tolerance, in a 1,530-byte ping response. Note log-log scale.

The CPU load is tolerable on modern PCs. A 200 MHz Pentium Pro is just able to saturate a 1 Mbit/sec link using all its processing power. A 2 GHz Pentium 4 has no trouble.

7 Conclusions and Extensions

By adding redundancy to transmitted 802.11 frames in software, we were able to extend the range of an outdoor line-of-sight link by 70 percent, or 0.37 miles, and reduce losses significantly on marginal indoor and outdoor links. But without link retransmissions for packets that fail error correction, users will still have difficulty making productive use of a marginal link, even with error coding. To be more usable, we will need to examine:

Retransmissions after error correction. To be able to use TCP over a marginal link, users of rs-link will have to duplicate the functionality of link-layer retransmissions at a higher level, after error correction. A loss rate of 17 percent, while certainly better than 68 percent, is still not good enough to run TCP. One way to add link-layer retransmissions above the error-correction layer is by using the LL-SMART-TCP-AWARE protocol described in [1]. Under LL-SMART-TCP-AWARE (or the simpler “snoop” protocol also discussed), link endpoints cache TCP segments as they cross the link, suppress duplicate acknowledgements, and retransmit segments that appear to have failed.

Error feedback. As implemented, rs-link uses 34 percent of the 802.11 link bandwidth for redundancy information. This is worth it if the result is a 75 percent reduction in loss rate, as in the case of our 0.55-mile link. But it's not worth it for a less-marginal link. rs-link should receive feedback on the amount of observed errors from the receiver and adjust the level of added redundancy as appropriate.

Why are Roofnet's results different? Preliminary efforts by the Roofnet group to record and salvage errored frames at 1 Mbit/sec have given disappointing results — errored frames end up completely corrupted. Why are their results so different from ours? Are urban links with multipath fundamentally different from the line-of-sight outdoor and through-walls indoor links we have examined?

8 Acknowledgments

The authors are deeply indebted to Austin Roach and Sanjit Biswas for lending equipment and much time to help us throughout this project. Phil Karn's free-software Reed Solomon library was very helpful, and we appreciate the assistance of Professor Hari Balakrishnan, Daniel Aguayo, John Barnett, Waseem Daher, Allie Brownell, Marissa Cheng, John Hawkinson, and the MIT Radio Society.

References

[1] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[2] S.A. Khayam, S.S. Karande, H. Radha, and D. Loguinov. Performance analysis and modeling of errors and losses over 802.11b lans for high-bitrate real-time multimedia. *Signal Processing: Image Communication*, 18(7), 2003.