Assignment 2: The Data Link Layer
Due the week of September 19/20, 2012

In the OSI reference model, the layer above the physical layer is the data link layer. The task of this layer is to provide a reliable means of communication between two hosts directly connected by a transmission channel. In the readings and exercises this week, you will spend time looking at how the data link and physical layers recognize when errors occur (a prerequisite to correcting them) and at algorithms and/or protocols that ensure reliable communication by retransmitting data when errors are detected.

Reading §2.3 through 2.5 of Peterson and Davie will provide an overview of the material for the week. Additional readings (described below) will be made available online.

A first step toward detecting and correcting errors is to delineate the stream of data flowing between two machines into distinct units. This task is called *framing*. Framing provides basic units to which error detecting codes can be applied. The first reading I would like you to do beyond the text this week delves more deeply into one framing technique, byte stuffing. The paper is "Consistent Overhead Byte Stuffing," by Cheshire and Baker. It both presents an alternative to the more traditional schemes for byte stuffing and bit stuffing discussed in the text and does a very nice job describing the traditional schemes. To save a little time, you can skip or just skim much of the "Theoretical Analysis" section of the paper.

Peterson and Davie include a discussion of an error detection scheme known as the cyclic redundancy check or CRC. The cyclic redundancy check is employed by some of the most widely used network protocols (Ethernet and WiFi). To gain some additional background on this scheme I ask you to read a "classic" paper by (a different) Peterson and Brown entitled "Cyclic Codes for Error Detection".[1] One of this week's problems is designed to make you recognize that despite all the polynomials, CRC codes are in some sense just parity bit codes. This reading is intended to counter that by giving you some sense of what the polynomials are good for.

Exercises

1. Below you will find the contents of an actual Internet data packet I captured on my home network. It is displayed in hexadecimal notation. The packet was 52 bytes long.

```
4500 0034 257e 4000 4006 88c7 0a00 0105
3f6f 420b f9f7 0050 66c2 5add 422b bdf1
8010 7ef6 c3f5 0000 0101 080a a6e5 6326
2094 c0ad
```

   (a) How long would the packet become if framed using the standard byte stuffing technique used in PPP?

[1] I would guess that the Peterson who authored of our textbook was about 4 years old when this paper about the CRC was published. As far as I know, these two Petersons are not related.

Another historical note of some (?) interest is that the CRC paper appeared in a publication entitled *Proceedings of the IRE*. The abbreviation IRE is short for "Institute of Radio Engineers". A few years after the paper was published, the IRE merged with another professional organization for engineers and became the Institute for Electrical and Electronics Engineers or IEEE. If you don't recognized the initials IEEE now, you will by the end of the semester. Many current networking standards are set by the IEEE. For example, the wireless networks we use are known as IEEE 802.11 networks and Ethernet is IEEE 802.3.

(b) How long would the packet become if framed using consistent overhead byte stuffing without zero pair elimination? with zero pair elimination?

2. The Chesire and Baker paper seems to provide a significant improvement over standard byte stuffing. If you think about the techniques used however, they should remind you of another framing technique discussed in Peterson and Davie, the "Byte-Counting Approach". As Peterson and Davie observe, this is a technique that "every Computer Science 101 student knows". Did Chesire and Baker just reinvent the wheel? Obviously, their proposal differs from the byte counting scheme described in Peterson and Davie. Rather than putting one byte count at the beginning of a frame, they spread counts throughout the frame in such a way that they never increase the size of the frame by more than one byte per "chunk". However, most practical networks limit packet size to less than 2^{16} bytes. The limit on the radio network with which they are concerned is even stricter. This means that if a single byte count was placed at the beginning of the packet, two bytes would be sufficient to hold it. This amount of overhead is both highly consistent and relatively modest. In their own discussion of fragmentation, Cheshire and Baker emphasize that it is undesirable to include features that add complexity to the implementation of network protocols. Is the added complexity of the Cheshire and Baker scheme justified if all it does is save a byte or two for small packets? Defend your position.

3. Do problem 2.18 from Peterson and Davie.

4. It strikes me as interesting that the material on error detecting codes provided by most texts jumps from the trivial (parity bits) to the mysterious (cyclic redundancy checks) or beyond. This problem attempts to show that this jump isn't as large as it may appear. In particular, the conclusion of your work on this problem should show you that the CRC is nothing but a fancy way of associating parity bits with sub-parts of a message.

   (a) Compute CRC's for each of the 4 bit messages 0001, 0010, 0100, and 1000 using the generator polynomial: $G(x) = x^3 + x + 1$.

   (b) For a given generator polynomial $G(x)$ of degree r and a given data length m, let

   $$C^{(i)}(x) = \sum_{j=0}^{r-1} c_j^{(i)} x^j$$

   be the remainder obtained by dividing $G(x)$ into x^{i+r} ($0 <= i < m$). For an arbitrary data polynomial

   $$M(x) = \sum_{i=0}^{m-1} m_i x^i$$

   show that the remainder polynomial obtained by dividing $x^r M(x)$ by $G(x)$ is

   $$C(x) = \sum_{i=0}^{m-1} m_i C^{(i)}(x)$$

   (using modulo 2 arithmetic).
   Be careful. It is easy to have the right intuition for this problem but to get the details of the math wrong. Make sure that you are not making unjustified assumptions about the operation of determining the remainder of a polynomial division. In particular, it is important to realize that the term "remainder" has a precise definition. In general, we say that q is the quotient and r is the remainder of dividing p by d if

- $p = qd + r$, and
- $0 \leq r < d$.

In the case of division involving polynomials, the "<" relationship in the second condition is interpreted as meaning of smaller degree.

(c) In completing the previous step, you have shown that the CRC is an example of a *linear code*. That is, if you add together the check bits associated with two messages, A and B, then the result will equal the check bits that should be associated with the message obtained by adding together the contents of A and B (assuming in both cases "add" means to add the binary strings together bit-wise using modulo 2 arithmetic which is the same as exclusive-oring the bit strings together).

Use this fact and the $C^{(i)}$'s computed in part (a) to compute the CRC for the message 1001 using the generator $G(x)$. Check your result by computing the CRC using the division procedure presented by Peterson and Davie.

(d) Letting

$$C(x) = \sum_{j=0}^{r-1} c_j x^j$$

show that

$$c_j = \sum_{i=0}^{m-1} m_i c_j^{(i)}$$

for all j, $0 \leq j < r$.

This shows that each c_j is really just a parity check on a subset of the message bits determined by G and the length of the message.

(e) Determine which message bits are "checked" by each parity bit in the CRC using $G(x)$ as described above for messages of length 4.

5. Now that you know that the CRC is in some sense just fancy parity bits, it is worth asking what all the polynomials are for. Suppose that you were given the task of choosing the generator polynomial to use in a CRC-based error detection system. Assume that the plan was to add only three bits to each message to detect errors. (No, this isn't very realistic, but it will make this problem easier by limiting your attention to at CRC polynomials that produce exactly 3 error bits.)

   How would you go about picking the "right" polynomial to use? Are there any that can be completely rejected? Why? How many are still left? What (if any) guidance can you gain by applying Theorems 1 through 4 in the Peterson and Brown paper to the polynomials that you can't completely reject? Are the codes associated with any of the polynomials Hamming codes? What is a Hamming code anyway?

   In one sense, this question is fairly limited. Because all the coefficients are just 0s and 1s, there are only 8 polynomials of degree 3 you need to consider. At the same time, this can be a dangerously open ended problem. Don't go overboard. I don't expect any of you to positively identify the "right" polynomial (I can't). My main hope is that you will explore the implications of the theorems in the paper enough to see how much (or little) guidance they really provide.

6. In their description of the sliding window protocol, Peterson and Davie suggest that the appropriate value for SWS is always the bandwidth-delay product while the RWS might be any value less

than or equal to SWS. For this problem I would like you to explore situations in which smaller values for RWS might result in situations where setting SWS to a value smaller than the bandwidth delay product might make sense.

One common value for RWS is 1. That is, the receiver buffers no packets, and, at any point, the only packet it will accept is a packet whose sequence number is one larger than the last packet received. When used in this way, the sliding window protocol is sometimes called GO-BACK-N because if a single packet is lost, the received will reject all subsequent packets until the sender times out and retransmits the lost packet. At this point, SWS packets (otherwise known as N packets) will have to be retransmitted.

Assume that node A is sending packets to node B using GO-BACK-N over a link that fails to deliver packets (and acknowledgments) with probability p. Suppose that A has transmitted packets with sequence numbers 1 through N where N is small enough that A would not yet expect to have received ACKs for any of the packets sent. At this point, if SWS was large enough, the algorithm described in the text would just send packet $N + 1$. It is not clear that this is the best choice. Obviously (I hope), A's optimal strategy would be to send the packet that has the highest probability of being received and accepted by B. So,

(a) Determine the probability that packet number $N + 1$ will be received and accepted by B if transmitted at this point (By "received and accepted" I mean that it gets there intact and that B doesn't discard it as a packet outside the range of sequence numbers it is currently willing to accept.).

(b) Determine the probability that a repeated copy of packet number M (where $1 <= M <= N$) will be received and accepted by B if transmitted at this point.

(c) Either argue that the value of the probability you determined for part a will always exceed that for part b (in which case A should never repeat packets before timing out) or give as simple a formula as you can expressing the relationship that must exist between M and N to justify sending packet M rather than $N + 1$.

(d) Now, suppose that

$$1 < N < RWS < \text{bandwidth-delay product measured in packets} <= SWS$$

Would it ever make sense to retransmit packet number M (where $1 <= M <= N$) rather than sending $N + 1$ in this case? Assume that if a packet loss occurs, the receiver uses its buffers to hold the first RWS packets send after the missing packet.

7. Do problem 2.33 from Peterson and Davie.