CS 336 Lecture 7– 2/23/98

Announcements

- Exercise 3 is now available on the web (due Friday, Feb. 27).

The Cyclic Redundancy Check

1. A significant role of the Data Link layer is to convert the potentially unreliable physical link between two machines into an apparently very reliable link.

2. This is done by including redundant information in each transmitted frame. Depending on the nature of the link and the data one can either:

   - include just enough redundancy to make it possible to detect errors and then arrange for the retransmission of damaged frames, or
   - include enough redundancy to enable the receiver to correct any errors produced during transmission.

   Most current networks take the former approach.

3. One widely used parity bit based error detection scheme is the cyclic redundancy check or CRC.

   - The CRC is based on some fairly impressive looking mathematics. It is helpful as you deal with its mathematical description that you recall that it is ultimately just a way to use parity bits.
   - The presentation of the CRC is based on two simple but not quite "every-day" bits of mathematics:
     - polynomial division
     - arithmetic over the field of integers mod 2.
   - Arithmetic over the field of integers mod 2 is simply arithmetic on single bit binary numbers with all carries (overflows) ignored. So 1 + 1 = 0 and so does 1 - 1. In fact, addition and subtraction are equivalent in this form of arithmetic.
   - Polynomial division isn't too bad either. There is an algorithm for performing polynomial division that looks a lot like the standard algorithm for integer division. More interestingly from the point of view of understanding the CRC, the definition of division (i.e. the definition of the quotient and remainder) are parallel.
     - When one says "dividing a by b produces quotient q with remainder r" where all the quantities involved are positive integers one really means that

       $$a = qb + r$$

       and that $0 \leq r < b$
     - When one says "dividing a by b produces quotient q with remainder r" where all the quantities are polynomials, one really means the same thing as when working with integers except that the meaning of "less than" is a bit different. For polynomials, less than means of lesser degree. So, the remainder of a polynomial division must be a polynomial of degree less than the divisor.
   - Now, we can put this all together to explain the idea behind the CRC.
     - Any particular use of the CRC scheme is based on selecting a generator polynomial G(x) whose coefficients are all either 0 or 1.

       Just to be different from the book, we will use

       $$x^3 + x^2 + 1$$

       as our example of a generator polynomial.
     - Given a message to be transmitted:

       $$b_n b_{n-1} b_{n-2} ... b_2 b_1 b_0$$

       view the bits of the message as the coefficients of a polynomial

       $$B(x) = b_n x^n + b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + ... b_2 x^2 + b_1 x + b_0$$
     - Multiply the polynomial corresponding to the message by x^k where k is the degree of the generator polynomial and then divide this product by the generator to obtain polynomials Q(x) and R(x) such that:

       $$x^k B(x) = Q(x)G(x) + R(x)$$

       Treating all the coefficients not as integers but as integers modulo 2.
     - Finally, treat the coefficients of the remainder polynomial, R(X) as "parity bits". That is, append them to the message before actually transmitting it.
     - Since the degree of R(x) is less than k, the bits of the transmitted message will correspond to the polynomial:

       $$x^k B(x) + R(x)$$

- Since addition and subtraction are identical in the field of integers mod 2, this is the same as

$$x^k B(x) - R(x)$$

- From the equation that defines division, however, we can conclude that:

$$x^k B(x) - R(x) = Q(x)G(x)$$

  In other words, if the transmitted message's bits are viewed as the coeeficients of a polynomial, then that polynomial will be divisible by G(X).

- This last fact is the basis of error checking using the CRC. When a message is received the corresponding polynomial is divided by G(x). If the remainder is non-zero, an error is detected. Otherwise, the message is assumed to be correct.

4. How about an example:

- Suppose we want to send a nice short message like 11010111 using the CRC with the polynomial $x^3 + x^2 + 1$ as our generator.
- The message corresponds to the polynomial:

$$x^7 + x^6 + x^4 + x^2 + x + 1$$

- Given G(x) is of degree 3, we need to multiply this polynomial by x^3 and then divide the result by G(x).
- I'll have to think about how to get this formatted better, but basically we have:

$$
\begin{array}{rllllll}
 & & x^7+ & & & x^2+ & 1 \\ \cline{2-7}
x^3+x^2+1 \;) & x^{10}+x^9+ & x^7+ & x^5+x^4+ & x^3 & & \\
 & x^{10}+x^9+ & x^7 & & & & \\ \cline{2-5}
 & & & x^5+x^4+ & x^3 & & \\
 & & & x^5+x^4+ & & x^2 & \\ \cline{3-6}
 & & & & x^3+ & x^2 & \\
 & & & & x^3+ & x^2+ & 1 \\ \cline{5-7}
 & & & & & & 1
\end{array}
$$

- The punchline is that the remainder is just "1". So, the parity bits added in this case would be 001.

5. I hope this is all strange enough that you feel compelled to ask "Why bother?". If so, the answer comes in two parts:

- While the computation of parity bits through polynomial division may seem rather complicated, with a little reflection on how the division algorithm works we will see how to implement a very simple algorithm for computing CRC bits and/or verifying the correctness of the CRC associated with a received message.
- The relationship between the bits and the polynomials will give us some mathematical leverage that will make it possible to prove facts about the sorts of errors the CRC associated with a given generator will detect.

6. Let's start by seeing how the mathematics underlying the CRC can be used to investigate its ability to detect errors.

- Suppose that we transmit the message corresponding to some polynomial B(x) after adding CRC bits. In this case, the transmitted bits will correspond to some polynomial, T(x), where

$$T(x) = B(x)x^k - R(x)$$

  where k is the degree of the generator polynomial and R(x) is the remainder obtained when B(x) is divided by the generator.
- Remember, the key property of T(x) is that it is divisible by G(x) (i.e. G(x) is a factor of T(x)).
- Now, if during transmission some of the bits of the message are damaged, the actual bits received will correspond to a different polynomial, T'(x). If we imagine computing

$$E(x) = T(x) - T'(x)$$

  then the coefficients of E(x) will correspond to a bit string with a one in each position where T(x) differed from T'(x) and zeroes everywhere else. Thus, E(x) corresponds to a bitmap of the positions at which errors occurred.
- As long as T'(x) is not divisible by G(x), our CRC bits will enable us to detect errors. SO, the cases we are really interesting are those where T'(x) is divisible by G(x).
- Given that we already know that T(x) is divisible by G(x), T'(x) must be divisible by G(x) if and only if E(x) is divisible by G(x). So, we can investigate the forms of errors that will go undetected by investigating polynomials, E(x), that are divisible by G(x).

  Better yet, one might prefer to say we can design good parity bit schemes by looking for polynomial, G(x), that do not evenly divide examples of E(x) that correspond to anticipated error patterns.

- Well, at the very least, it would be nice to make sure that the CRC did as well as adding a single parity bit. That is, we would like to avoid using any G(x) that did not guarantee we could detect all instances of errors that change an odd number of bits.
  - If a received message T'(x) contains an odd number of inverted bits, then E(x) must contain an odd number of terms with coefficients equal to 1.
  - As a result, E(1) must equal to 1 (since if x = 1 then $x^i = 1$ for all i).
  - If G(x) is a factor of E(x), then G(1) would also have to be 1.
  - So, if we make sure that G(1) = 0, we can conclude that G(x) does not divide any E(x) corresponding to an odd number of error bits. In this case, a CRC based on G(x) will detect any odd number of errors.
  - As long as G(x) has some factor of the form $x^i + 1$, G(1) will equal 0. So, it isn't hard to find such a polynomial.
- As a sanity check, consider the CRC associated with the simplest G(x) that contains a factor of the form $x^i + 1$, namely $x + 1$.
  - The CRC for any message consisting entirely of zeroes will be zero.
  - Given that the code is guaranteed to detect any error involving an odd number of bits, if we start with all zeroes and add 1's in various posisiton, the parity bit must flip each time we add a one.
  - Accordingly, the value of the parity bit will be 1 if and only if the number of 1's is odd.
  - In other words, when the generator is $x + 1$ the CRC is just a single even parity bit!
- Having discovered this amusing fact, let's make sure that the CRC does more than a single parity bit if we choose an appropriate polynomial of higher degree. Consider how the CRC behaves is G(x) is $x^k + 1$ for some k larger than one.
  - Obviously, this CRC will catch any error that changes an odd number of bits.
  - I argued last time, however, that one generally worries more about burst errors than isolated errors. So, consider the case where a burst error affects some subset of j consecutive bits for j ¡ k.
  - In this case, the error polynomial will look like

    $$E(x) = x^{n_1} + x^{n_2} + ...x^{n_r}$$

    where we assume that $n_i > n_{i+1}$ for all i and that $n_1 - n_r <= j$.

    Factoring out the lowest degree term in this polynomial gives:

    $$E(x) = x^{n_r}(x^{n_1-n_r} + x^{n_2-n_r} + ... + 1)$$

  - Now, $G(x) = x^k + 1$ can not divide x^{n_r}. So, the only way that G(x) can divide E(x) is if if divides $x^{n_1-n_r} + x^{n_2-n_r} + ... + 1$. However, G(x) can not possible divide a polynomial of degree less than k. So, it can not divide E(x).
  - Thus, we can conclude that the CRC based on our simple G(x) detects all burst errors of length less than its degree.
- There are many other examples of ways that properties of G(x) can be used to ensure that the associated CRC will detect certain errors. The text provides a few more examples.

7. The other aspect of the CRC that makes it valuable is the surprising fact that it is actually quite simple to calculate.

- In a transmission system, the ideal is to be able to calculate the checksum bit by bit as the data arrives (rather than having to wait for the whole frame to arrive before we can start the process). This is how the calculation of the CRC is performed.
- As a result, the best way to undertand the computation is to assume you have received a message and computed its CRC when to your suprise and extra bit of the message arrives. How could you calculate the CRC of the message with the added bit included based on the CRC you had computed before you knew about the extra bit?
- Well, let's call the partial message T(X) and assume that the partial CRC is R(x) such that for some Q(X)

  $$x^k T(X) = Q(x)G(x) + R(x)$$

  and lets call the extra bit "b".
- By multiplying both sides of the equation above by x, adding b to both sides and then regrouping a bit we obtain

  $$x^k(T(X)x + b) = (xQ(x))G(x) + (R(x)x + b)$$

- If $R(x)x$ is of degree less than $G(x)$ then we are done. In this case, by the definition of polynomial division, $R(x)x + b$ is the remainder of dividing x^k time the polynomial corresponding to the complete message by G(x). Since multiplying a polynomial by x just raises the power of all its terms, this means we can obtain the CRC for the polynomial with b added by throwing away the first bit of the old CRC (which must have been 0) and adding b as the last bit.

- If the first bit of the old CRC was 1, then $R(x)x$ will have degree k. Luckily, in this case

$$R(x)x + b - Q(x)$$

is guaranteed to have degree less than k since the coeffients of x^k in both Q(x) and $R(x)x$ must be 1 and 1+1 = 0.

In this case, regrouping the terms of the equation above to yield

$$x^k(T(X)x + b) = (xQ(x) + 1)G(x) + (R(x)x + b - G(x))$$

allows us to conclude that this time

$$R(x)x + b - Q(x)$$

must be the new remainder.

Accordingly, when the first bit of the CRC is 1, we can compute the new CRC by first shifting the old CRC left by one, then adding the new bit b and then adding G(x). Since addition mod 2 is just the exclusive or operation, we can just exclusive or the bits of the shifted CRC with the bits corresponding to the coefficients of G(x).

- As the text shows, these rules can be expressed by cute little diagrams that can ultimately be turned into simple and highly efficient circuits for computing and verifying CRCs.