Lecture 9:
Plan: CRC

Review: we talked about the basic functionality of data link layer (framing, error control and flow control) and framing. The topic for today is CRC.

Error Detection/Correction:

- Basic Idea: adding redundancy to the transmitted message
  - detection: the receiver knows that something is wrong with the received data. Ask the sender to re-send
  - correction: the receiver not only knows if something is wrong but also is able to correct the mistakes.
    Q:/ Shouldn't we always use correction scheme?
    A:/ correction needs more information included. Thus more redundancy → use more bandwidth.
    Q:/ When to use correction?
    A:/ in wireless network, the error rate is high. To avoid large number of retransmission, correction is preferred.

1). Error Detection
CRC (Cyclic Redundancy Coding)

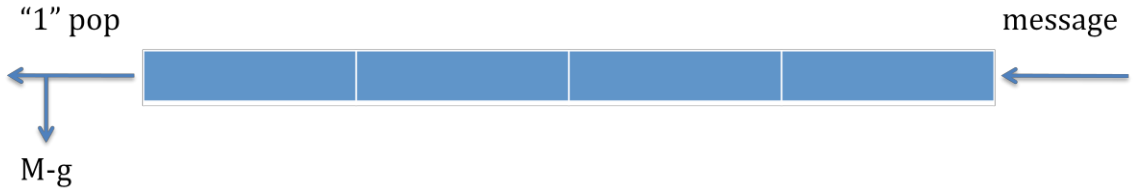
- Basic Idea: Based on the arithmetic of polynomials with coefficients taken from Z_2 (ring of integers mod 2). In this domain, 2 is the same as 0.
  - 1=-1, so addition is the same as subtraction
  - carries and borrows are never necessary. They are both the same as exclusive-or
    e.g. multiply $x^3 + x^2 + 1$ with x^7+x^5+x+1.
    result is $x^{10}+x^9+x^8+x^5+x^4+x^2+x+1$
    e.g. divide 1011101 by 1101
    result is 1100 with remainder 1
  - CRC scheme: define a generating polynomial (known by both the sender and the receiver) g(x) of order r (the highest degree is r). Suppose we want to send a message of m bits, represented by a polynomial m(x) of degree m
    - Append r zeros to the message (the resulting polynomial is $x^r m(x)$)
    - Divide $x^r m(x)$by g(x) and determine the remainder r(x)
    - The binary representation of r(x) is the CRC code. The sender sends the message followed by the CRC code (corresponds to $x^r m(x)$-r(x))
    - When the message is received, the sender divides it by g(x). If the remainder is not zero, error has occurred.
      e.g. g=10011, message = 11001011010
      - the g(x) is of order 4, so append 4 zeros to the message
      - divide it by g. the remainder is 1011.
      - The sender sends the original message followed by the 4 bit remainder.
        110010110101011
      - When the message arrives, the receiver divides the received message by g. If the message is changed, the remainder will not be zero.
  - Why does it work?

P166

g(x) should divide the $x^r m(x)$-r(x)completed since the remainder is already subtracted.

- Division is not fun in calculations. → we can use a shift register to compute.
  Receiver algorithm:

```
M = 0; // it is the length of the remainder, i.e. CRC code length
while more bits {
   lshift(M);
   M = M | next bit;
   if(M_high == 1)
     M = M xor g;
}
if(M!=0) error
```



- This might look a bit messy, but all we are really doing is "subtracting" various powers (i.e. shiftings) of the poly from the message until there is nothing left but the remainder. demonstrate the previous example if time allows.
- In the real engineering world, IEEE 802 uses a generating polynomial of degree 32.
  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
  It can catch all single or double bit errors; all erros with an odd number of bits; all burst erros of length 32 or less