Networks and Distributed Systems

Lecture 4 – Error detection and reliable transmission

Outline

- Perspectives on Connecting nodes
- Encoding
- Framing
- Error Detection
- Reliable Transmission
- Ethernet and Multiple Access Networks
- Wireless Networks

Cyclic Redundancy Check (CRC)

- Reduce the number of extra bits and maximize protection
- Given a bit string 110001 we can associate a polynomial on a single variable *x* for it.

  $1.x^5+1.x^4+0.x^3+0.x^2+0.x^1+1.x^0 = x^5+x^4+1$ and the degree is 5.

  A *k*-bit frame has a maximum degree of *k*-1
- Let M(x) be a message polynomial and C(x) be a generator polynomial.

Cyclic Redundancy Check (CRC)

- Let M(x)/C(x) leave a remainder of 0.
- When M(x) is sent and M' (x) is received we have M' (x) = M(x)+E(x)
- The receiver computes M' (x)/C(x) and if the remainder is nonzero, then an error has occurred.
- The only thing the sender and the receiver should know is C(x).

Cyclic Redundancy Check (CRC)

Polynomial Arithmetic Modulo 2

- Any polynomial B(x) can be divided by a divisor polynomial C(x) if B(x) is of higher degree than C(x).
- Any polynomial B(x) can be divided once by a divisor polynomial C(x) if B(x) is of the same degree as C(x).
- The remainder obtained when B(x) is divided by C(x) is obtained by subtracting C(x) from B(x).
- To subtract C(x) from B(x), we simply perform the exclusive-OR (XOR) operation on each pair of matching coefficients.

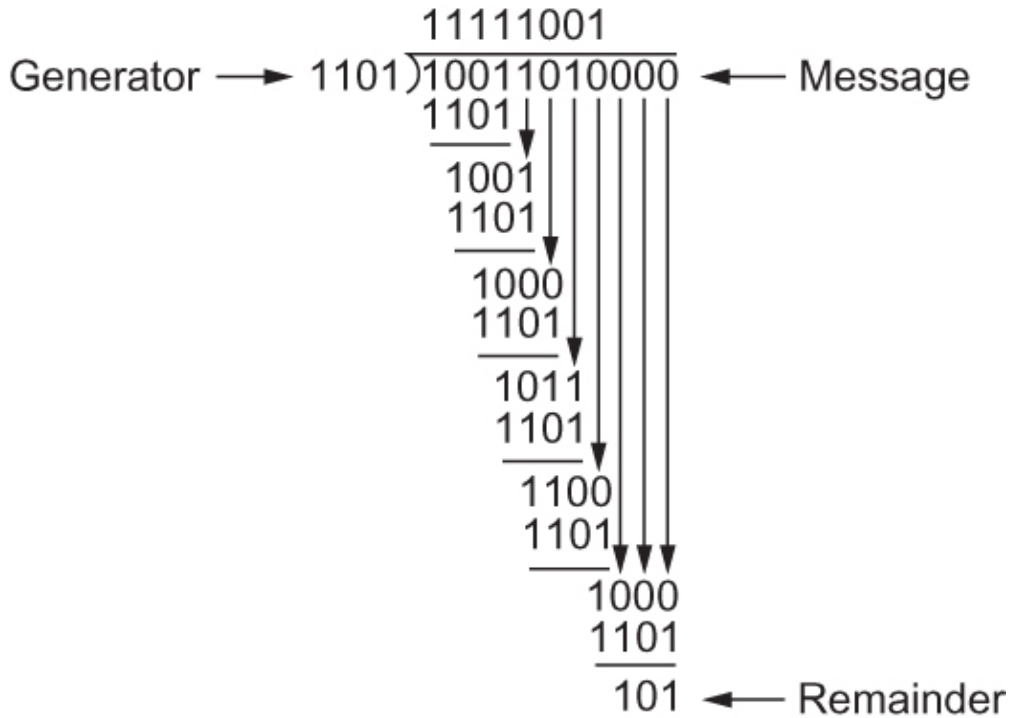
Cyclic Redundancy Check (CRC)

- Let M(x) be a frame with *m* bits and let the generator polynomial have less than *m* bits say equal to *r*.
- Let *r* be the degree of C(x). Append *r* zero bits to the low-order end of the frame, so it now contains *m*+*r* bits and corresponds to the polynomial $x^r M(x)$.

Cyclic Redundancy Check (CRC)

- Divide the bit string corresponding to $x^r M(x)$ by the bit string corresponding to $C(x)$ using modulo 2 division.
- Subtract the remainder (which is always *r* or fewer bits) from the string corresponding to $x^r M(x)$ using modulo 2 subtraction (addition and subtraction are the same in modulo 2).
- The result is the checksummed frame to be transmitted. Call it polynomial $M'(x)$.

Cyclic Redundancy Check (CRC)



CRC Calculation using Polynomial Long Division

Cyclic Redundancy Check (CRC)

- Properties of Generator Polynomial
  - Let P(x) represent what the sender sent and P(x) + E(x) is the received string. A 1 in E(x) represents that in the corresponding position in P(x) of the message the bit is flipped.
  - We know that P(x)/C(x) leaves a remainder of 0, but if E(x)/C(x) leaves a remainder of 0, then either E(x) = 0 or C(x) is factor of E(x).
  - When C(x) is a factor of E(x) we have problem; errors go unnoticed.
  - If there is a single bit error then E(x) = x^i, where *i* determines the bit in error. If C(x) contains two or more terms it will never divide E(x), so all single bit errors will be detected.

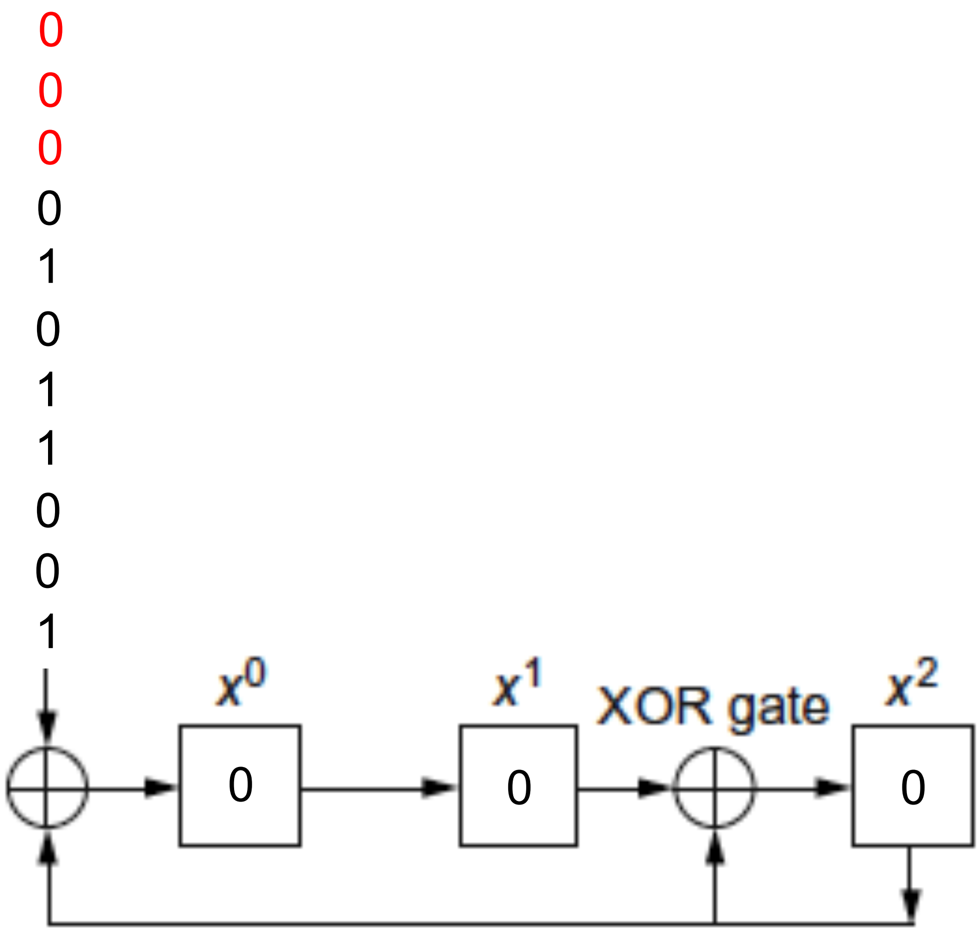
Cyclic Redundancy Check (CRC)

- Properties of Generator Polynomial
  - In general, it is possible to prove that the following types of errors can be detected by a C(x) with the stated properties
    - All single-bit errors, as long as the x^k and x^0 terms have nonzero coefficients.
    - All double-bit errors, as long as C(x) has a factor with at least three terms.
    - Any odd number of errors, as long as C(x) contains the factor (x+1).
    - Any “burst” error (i.e., sequence of consecutive error bits) for which the length of the burst is less than *k* bits. (Most burst errors of larger than *k* bits can also be detected.)

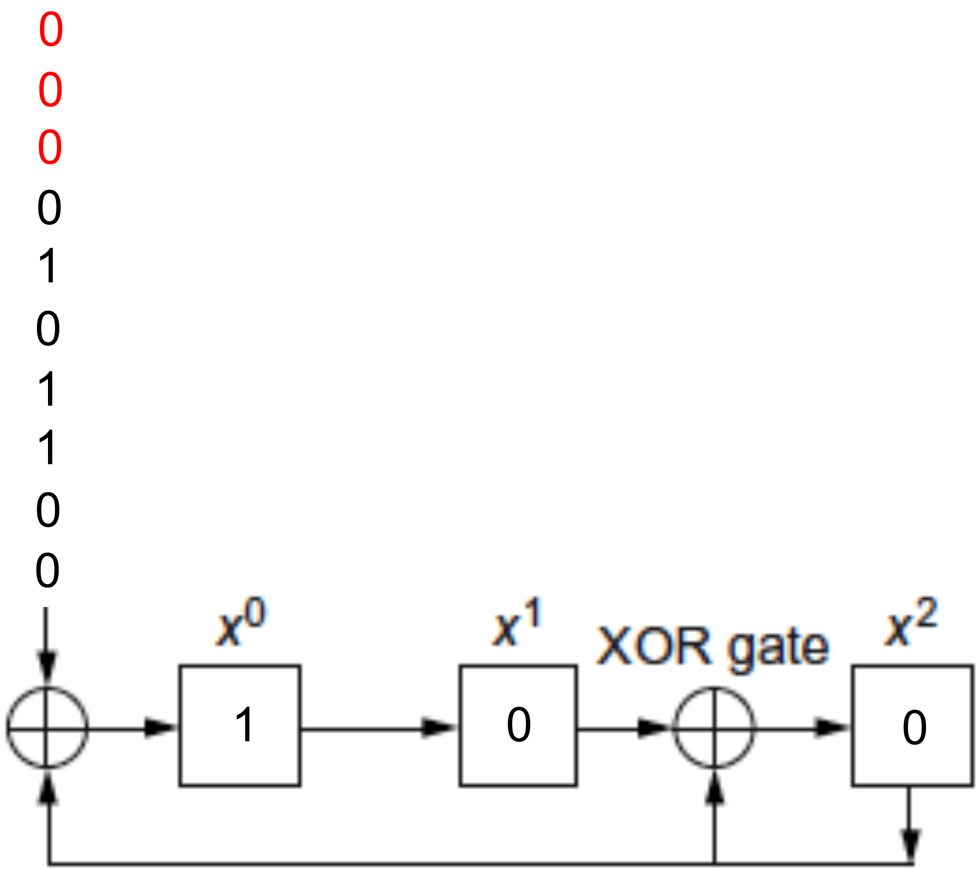
Cyclic Redundancy Check (CRC)

- Six generator polynomials that have become international standards are:
  - CRC-8 = x^8+x^2+x+1
  - CRC-10 = $x^{10}+x^9+x^5+x^4+x+1$
  - CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$
  - CRC-16 = $x^{16}+x^{15}+x^2+1$
  - CRC-CCITT = $x^{16}+x^{12}+x^5+1$
  - CRC-32 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

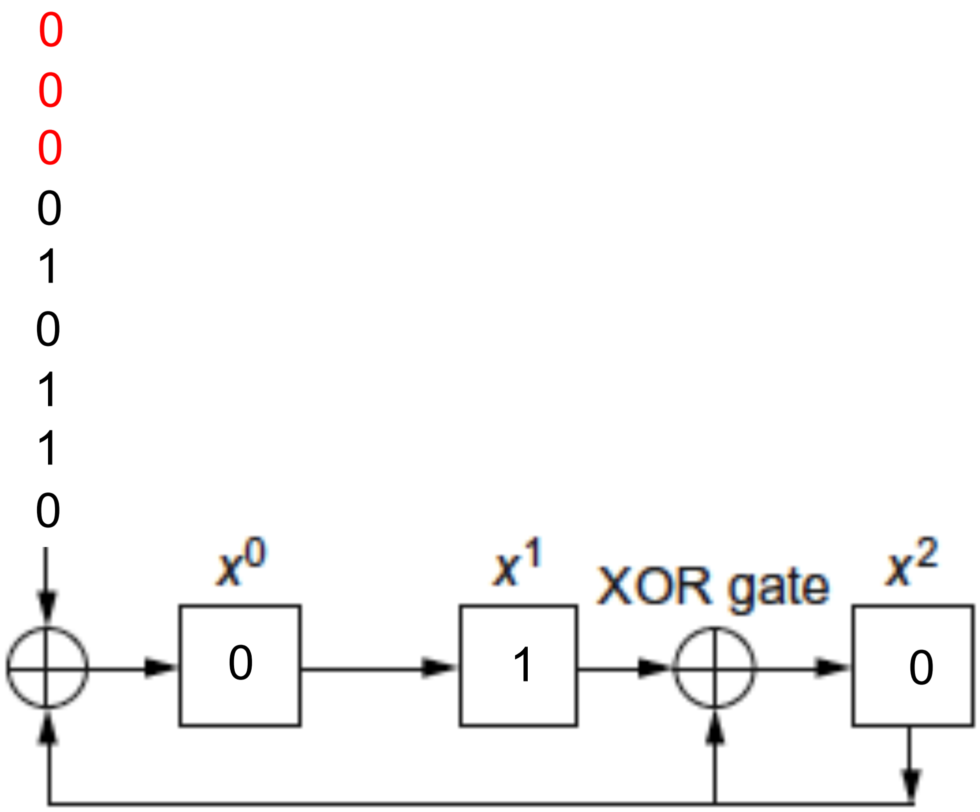
Cyclic Redundancy Check (CRC)



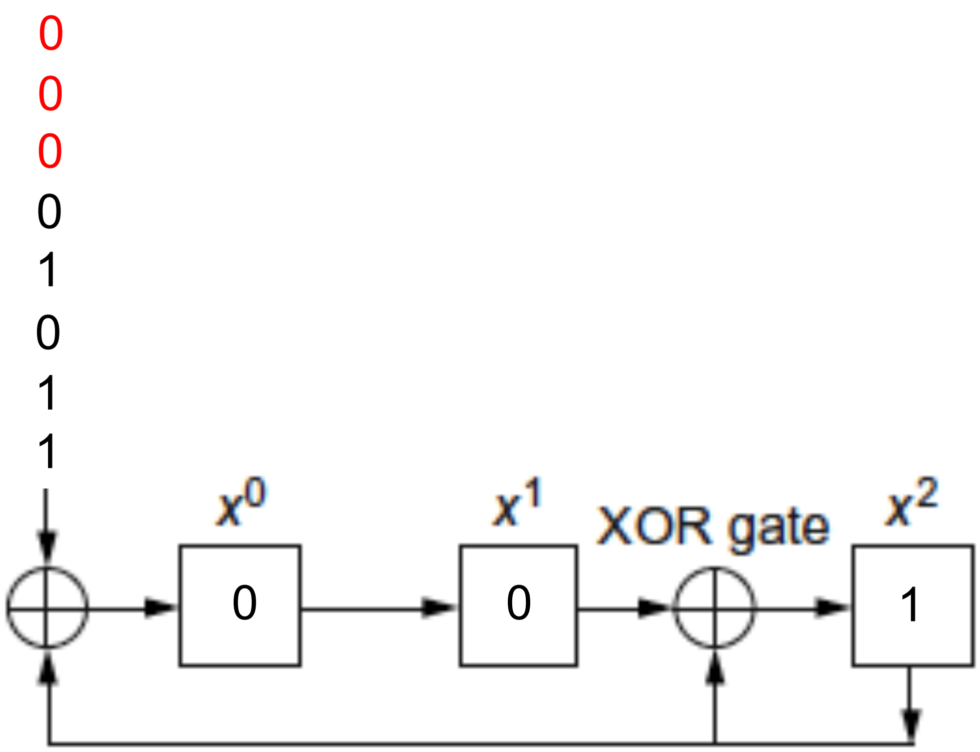
Cyclic Redundancy Check (CRC)



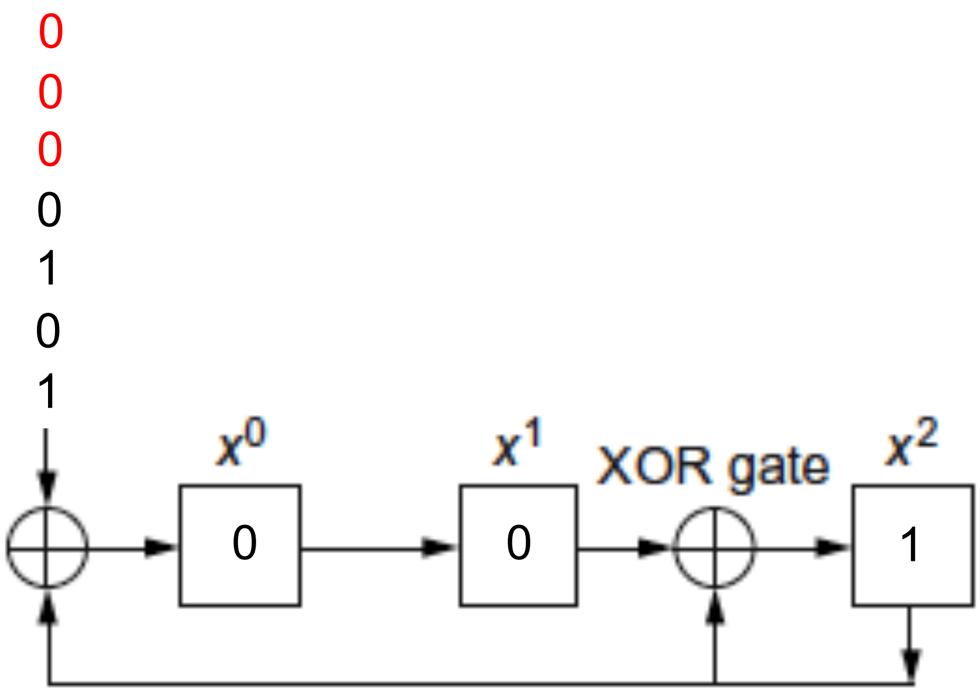
Cyclic Redundancy Check (CRC)



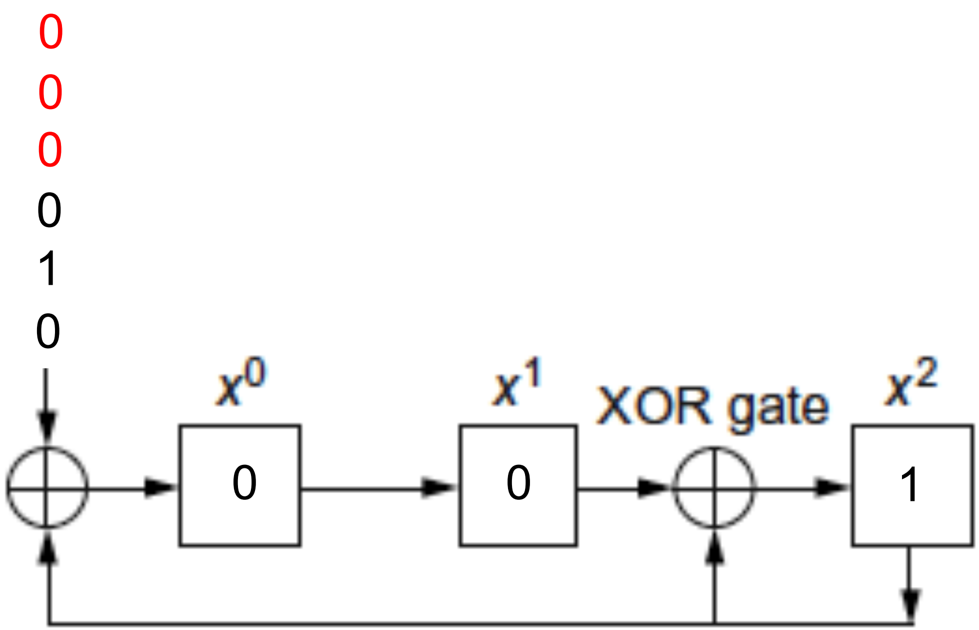
Cyclic Redundancy Check (CRC)



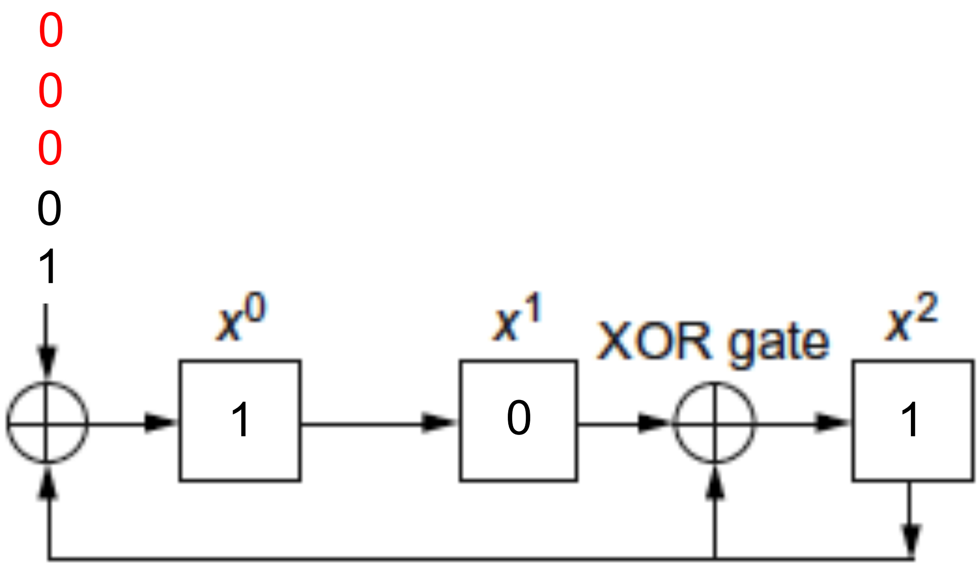
Cyclic Redundancy Check (CRC)



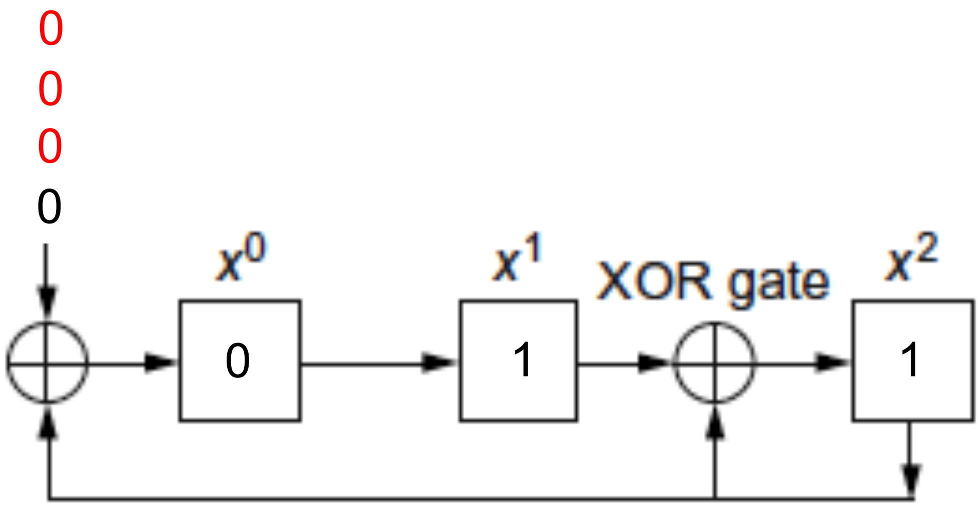
Cyclic Redundancy Check (CRC)



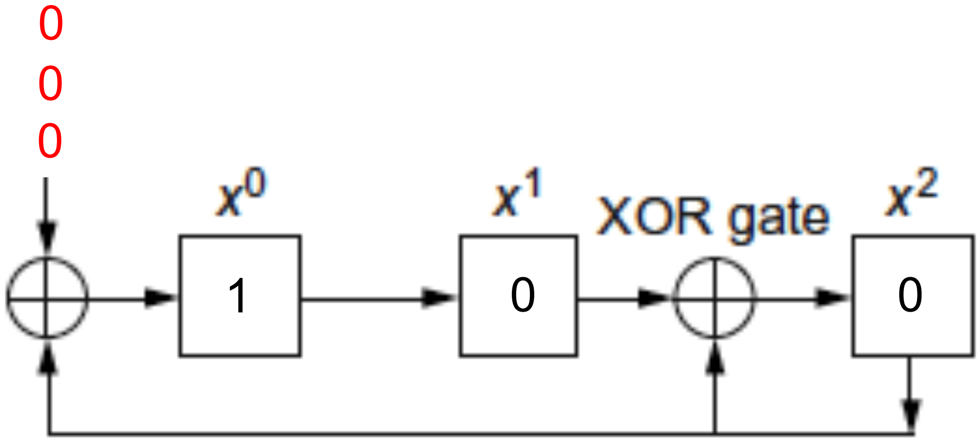
Cyclic Redundancy Check (CRC)



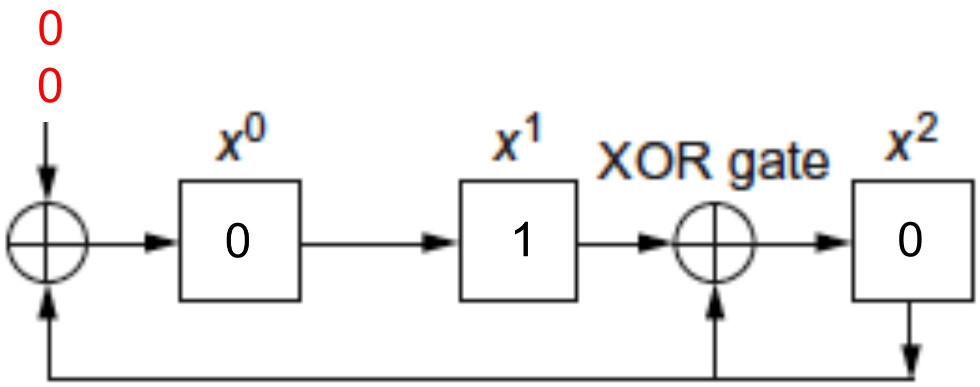
Cyclic Redundancy Check (CRC)



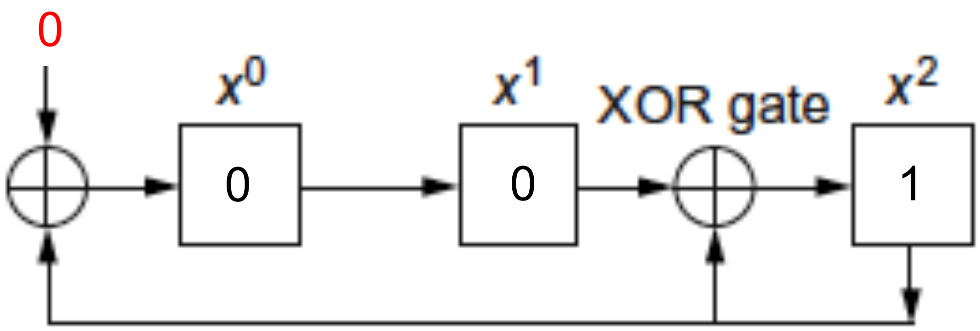
Cyclic Redundancy Check (CRC)



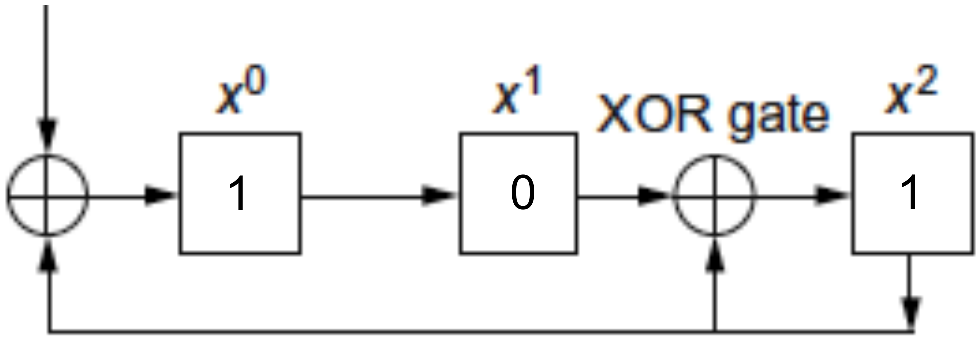
Cyclic Redundancy Check (CRC)



Cyclic Redundancy Check (CRC)



Cyclic Redundancy Check (CRC)



Reliable Transmission

- CRC is used to detect errors.
- Some error codes are strong enough to correct errors.
- The overhead is typically too high.
- Corrupt frames must be discarded.
- A link-level protocol that wants to deliver frames reliably must recover from these discarded frames.
- This is accomplished using a combination of two fundamental mechanisms
  - Acknowledgements and Timeouts

Reliable Transmission

- An *acknowledgement* (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received the earlier frame.
  - A control frame is a frame with header only (no data).
- The receipt of an *acknowledgement* indicates to the sender of the original frame that its frame was successfully delivered.

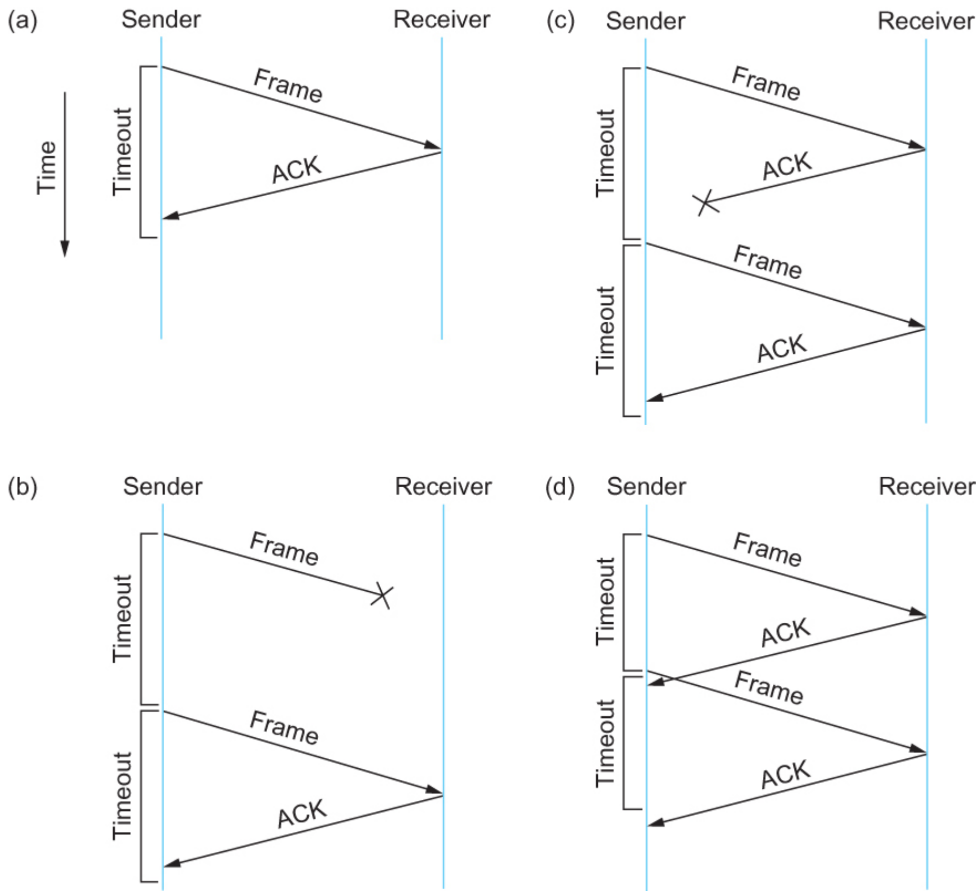
Reliable Transmission

- If the sender does not receive an *acknowledgment* after a reasonable amount of time, then it retransmits the original frame.
- The action of waiting a reasonable amount of time is called a *timeout*.
- The general strategy of using *acknowledgements* and *timeouts* to implement reliable delivery is sometimes called Automatic Repeat reQuest (ARQ).

Stop and Wait Protocol

- Idea of stop-and-wait protocol is straightforward
  - After transmitting one frame, the sender waits for an acknowledgement before transmitting the next frame.
  - If the acknowledgement does not arrive after a certain period of time, the sender times out and retransmits the original frame

Stop and Wait Protocol



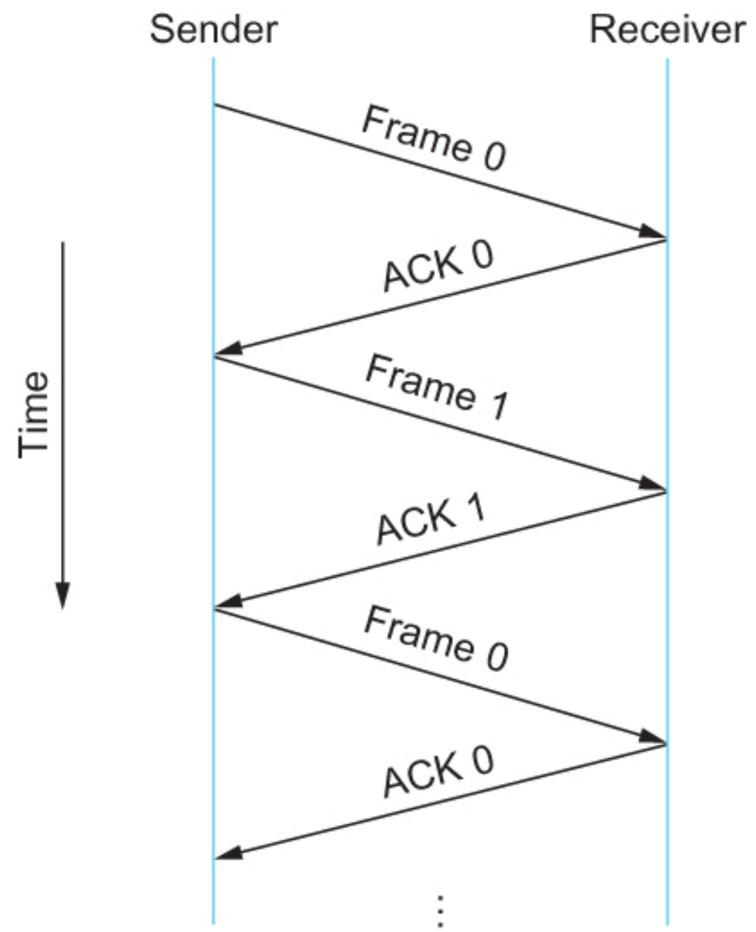
Timeline showing four different scenarios for the stop-and-wait algorithm.
(a) The ACK is received before the timer expires; (b) the original frame is lost; (c) the ACK is lost; (d) the timeout fires too soon

Stop and Wait Protocol

- If the acknowledgment is lost or delayed in arriving
  - The sender times out and retransmits the original frame, but the receiver will think that it is the next frame since it has correctly received and acknowledged the first frame
  - As a result, duplicate copies of frames will be delivered

- How to solve this
  - Use 1 bit sequence number (0 or 1)
  - When the sender retransmits frame 0, the receiver can determine that it is seeing a second copy of frame 0 rather than the first copy of frame 1 and therefore can ignore it (the receiver still acknowledges it, in case the first acknowledgement was lost)

Stop and Wait Protocol



Timeline for stop-and-wait with 1-bit sequence number

Stop and Wait Protocol

- The sender has only one outstanding frame on the link at a time
  - This may be far below the link's capacity
- Consider a 1.5 Mbps link with a 45 ms RTT
  - The link has a delay × bandwidth product of 67.5 Kb or approximately 8 KB
  - Since the sender can send only one frame per RTT and assuming a frame size of 1 KB
  - Maximum Sending rate
    - Bits per frame ÷ Time per frame = 1024 × 8 ÷ 0.045 = 182 Kbps
      Or about one-eighth of the link's capacity
  - To use the link fully, then sender should transmit up to eight frames before having to wait for an acknowledgement