Overview

- Direct link networks
  - Error detection - Section 2.4
  - Reliable transmission - Section 2.5

Error Detection

- Problem of detecting errors introduced into frames
- Specific mechanism that introduces errors depends on the network technology - thermal, radio interference etc.
- Sender introduces additional error detecting codes that are based on the actual data. Receiver recomputes the code for the received data. If the computed code mis-matches the error code computed by the sender; there was an error in the transmission. Sender and receiver use the same algorithm to create these codes.

Two dimensional parity

- 7 bit data, add one parity bit$_{\text{Parity}}$
- Compute parity of all bit positions to create a single parity byte for the entire frame
- Catches 1, 2, and 3 bit errors (most 4-bit errors)

Internet Checksum Algorithm

- View message as a sequence of 16-bit integers; sum using 16-bit ones-complement arithmetic; take ones-complement of the result.
- Simple, last line of defense (e2e argument)

```
u_short cksum(u_short *buf, int count) {
   register u_long sum = 0;
   while (count--){
      sum += *buf++;
      if (sum & 0xFFFF0000){
         /* carry occurred, so wrap around */
         sum &= 0xFFFF;
         sum++;
      }
   }
   return ~(sum & 0xFFFF);
}
```


Cyclic Redundancy Check

- Theoretical foundation lies in finite fields
- Offers stronger protection

- Represent n-bit message as n-1 degree polynomial
  - e.g., MSG=10011010 as $M(x) = x^7 + x^4 + x^3 + x^1$
- Let k be the degree of some divisor polynomial
  - e.g., $C(x) = x^3 + x^2 + 1$ (k = 3 here)
- C(x) is chosen apriori
- Add k bits of redundant data to an n-bit message
  - want k << n
  - e.g., in Ethernet, k = 32 and n = 12,000 (1500 bytes)
  - Transmitted message is P(x)

CRC (cont)

- Transmit polynomial P(x) that is evenly divisible by C(x)
  - shift left k bits, i.e., $M(x).x^k$
  - Divide by C(x) and find the remainder
  - subtract remainder of $M(x).x^k/C(x)$ from $M(x).x^k$
- Receiver polynomial P(x) + E(x)
  - E(x) = 0 implies no errors
- Divide (P(x) + E(x)) by C(x); remainder zero if:
  - E(x) was zero (no error), or
  - E(x) is exactly divisible by C(x)

Selecting C(x)

- All single-bit errors, as long as the x^k and x^0 terms have non-zero coefficients.
- All double-bit errors, as long as C(x) contains 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
- See Table 2.5 on page 96 for common C(x)

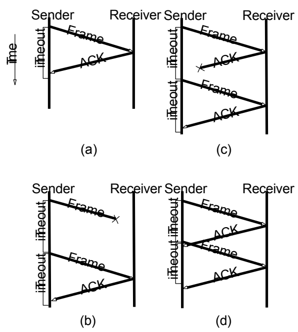
CRC polynomials

- CRC-8: $x^8+x^2+x^1+1$
- CRC-10: $x^{10}+x^9+x^5+x^4+x^1+1$
- CRC-12: $x^{12}+x^{11}+x^3+x^2+1$
- CRC-16: $x^{16}+x^{15}+x^2+1$
- CRC-CCITT: $x^{16}+x^{12}+x^5+1$
- CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+ x^{16}+x^{12}+x^{11}+x^{10}+x^8$ $x^7+x^5+x^4+x^2+1$
- Ethernet uses CRC-32
- HDLC: CRC-CCITT
- ATM: CRC-8, CRC-10, and CRC-32

Reliable Transmission

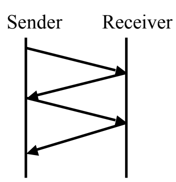
- When corrupt frames are received and discarded, want the network to recover from these errors
- Two fundamental mechanisms:
  - Acknowledgement: A control message sent back to the sender to notify correct receipt
  - Timeout: If sender does not receive and Ack after timeout interval, it should retransmit the original frame
- This general strategy is called ARQ: Automatic Repeat Request

Acknowledgements & Timeouts



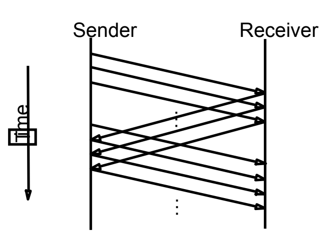
Stop-and-Wait

- Problem: keeping the pipe full
  - (bandwidth delay product)
- Example
  - 1.5Mbps link x 45ms RTT = 67.5Kb (8KB)
  - 1 KB frames imples 1/8th link utilization



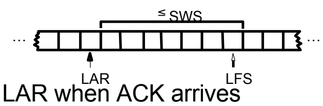
Sliding Window

- Allow multiple outstanding (un-ACKed) frames
- Upper bound on un-ACKed frames, called window



SW: Sender

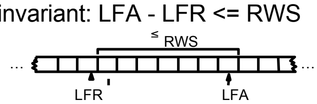
- Assign sequence number to each frame (SeqNum)
- Maintain three state variables:
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame sent (LFS)
- Maintain invariant: LFS - LAR <= SWS



- Advance LAR when ACK arrives
- Buffer up to SWS frames

SW: Receiver

- Maintain three state variables
  - receive window size (RWS)
  - largest frame acceptable (LFA)
  - last frame received (LFR)
- Maintain invariant: LFA - LFR <= RWS



- Frame SeqNum arrives:
  - if LFR < SeqNum < = LFA → accept
  - if SeqNum < = LFR or SeqNum > LFA → discarded
- Send cumulative ACKs

Acknowledgements

- Negative acknowledgment (NAK)
  - When receiver receives frame which has a sequence number higher than the next frame expected, receiver proactively informs the sender to resend the missing frame
- Selective ACK
  - Acknowledge frames that it has received, not just the last frame received

Sequence Number Space

- SeqNum field is finite; sequence numbers wrap around
- Sequence number space must be larger then number of outstanding frames
- SWS <= MaxSeqNum-1 is not sufficient
  - suppose 3-bit SeqNum field (0..7)
  - SWS=RWS=7
  - sender transmit frames 0..6
  - arrive successfully, but ACKs lost
  - sender retransmits 0..6
  - receiver expecting 7, 0..5, but receives second incarnation of 0..5
- SWS < (MaxSeqNum+1)/2 is correct rule
- Intuitively, SeqNum "slides" between two halves of sequence number space

Concurrent Logical Channels

- Multiplex 8 logical channels over a single link
- Run stop-and-wait on each logical channel
- Maintain three state bits per channel
  - channel busy
  - current sequence number out
  - next sequence number in
- Header: 3-bit channel num, 1-bit sequence num
  - 4-bits total
  - same as sliding window protocol
- Separates reliability from order