Data Link Layer

Abusayeed Saifullah

CS 5600 Computer Networks

These slides are adapted from Kurose and Ross
Error detection

EDC = Error Detection and Correction bits (redundancy)
D   = Data protected by error checking, may include header fields

• Error detection not 100% reliable!
  • protocol may miss some errors, but rarely
  • larger EDC field yields better detection and correction
**Parity checking**

*single bit parity:*

- Even scheme: choose \((d+1)\)-th bit to make an even number of 1's

- Receiving an odd # of 1's $\rightarrow$ odd # of bit error

- Can we detect if there is an even number of bit errors?
**Parity checking**

**single bit parity:**
- Even scheme: choose \((d+1)\)-th bit to make an even number of 1’s
- Receiving an odd # of 1’s → odd # of bit error
- Receiving an even # of 1’s → ??????????

**two-dimensional bit parity:**
- detect and correct single bit errors

<table>
<thead>
<tr>
<th>Parity bit</th>
<th>0111000110101011</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>d data bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccccccc}
& d_1,1 & \cdots & d_{1,j} & d_1, j+1 \\
& d_2,1 & \cdots & d_{2,j} & d_{2,j+1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
& d_{i,1} & \cdots & d_{i,j} & d_{i,j+1} \\
& d_{i+1,1} & \cdots & d_{i+1,j} & d_{i+1,j+1} \\
\end{array}
\]

\[
\begin{array}{c}
1010111 \\
111100 \\
011101 \\
001010 \\
\end{array}
\]

\[
\begin{array}{c}
1010111 \\
101100 \\
011101 \\
001010 \\
\end{array}
\]

*parity error*  
*correctable single bit error*
**Internet checksum (Transport layer)**

**Goal:** detect “errors” (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

**Sender:**
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1’s complement sum) of segment contents
- sender puts checksum value into UDP checksum field

**Receiver:**
- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*
CRC (Cyclic Redundancy Check)

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (using modulo 2 arithmetic)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- widely used in practice (Ethernet, 802.11 WiFi, 802.15.4)
CRC (Cyclic Redundancy Check)

- more powerful error-detection coding
- view data bits, $D$, as a binary number
- choose $r+1$ bit pattern (generator), $G$
- goal: choose $r$ CRC bits, $R$, such that
  - $<D,R>$ exactly divisible by $G$ (using modulo 2 arithmetic)
  - receiver knows $G$, divides $<D,R>$ by $G$. If non-zero remainder: error detected!
- widely used in practice (Ethernet, 802.11 WiFi, 802.15.4)

$d$ bits $\rightarrow r$ bits $\rightarrow$

$D$: data bits to be sent  $R$: CRC bits

$D \times 2^r$ XOR $R$  [Mathematical formula]
How do we calculate $R$?

want:
$$D \cdot 2^r \text{ XOR } R = nG$$

equivalently:
$$D \cdot 2^r = nG \text{ XOR } R$$

equivalently:
$$R = \text{remainder}\left(\frac{D \cdot 2^r}{G}\right)$$

Who defines $G$?
Elementary Data Link Protocols

- An Unrestricted Simplex Protocol
- A Simplex Stop-and-Wait Protocol
- A Simplex Protocol for a Noisy Channel
Unrestricted Simplex Protocol

- Simplex: transmission in one direction
- Infinite buffer: receiver always ready to receive the next frame
- Assumption: error-free channel
- No acknowledgments (ACK) or retransmissions used.
- Channel utilization = ??

Diagram:

Sender

Data Frame 1

Data Frame 2

Data Frame 3

Data Frame 4

Data Frame 5

Data Frame 6

... ...

Receiver

Frame transmission time

One way Latency
Unrestricted Simplex Protocol

- Simplex: transmission in one direction
- Infinite buffer: receiver always ready to receive the next frame
- Assumption: error-free channel
- No acknowledgments (ACK) or retransmissions used.
- Channel utilization = 100%
Sending Frames Across

Transmission Delay

Propagation Delay

Latency
Sending Frames Across

100% utilization of channel

Throughput: bits / s
Simplex Stop-and-Wait Protocol

- Finite buffer: receiver may not be always ready to receive.
- Receiver sends ACK to sender to transmit the next data frame.
- Error-free channel assumed: no retransmissions

Channel utilization

\[ U = \frac{1}{1 + 2a} \]

where \( a = \frac{t_{\text{prop}}}{t_{\text{trans}}} \)
Stop-and-Wait: Channel utilization

- **Example**
  - channel capacity: 50 kbps, frame size: 1kb
  - round-trip propagation delay: 500 msec
  - Time: $t=0$ start to send 1st bit in frame
    - $t=20$ msec frame sent completely
    - $t=270$ msec frame arrives
    - $t=520$ msec best case of ack. Received

- Sender blocked ?? % of time
- Channel utilization ?? %
Stop-and-Wait: Channel utilization

- **Example**
  - channel capacity: 50 kbps, frame size: 1kb
  - round-trip propagation delay: 500 msec
  - Time: $t=0$ start to send 1st bit in frame
    - $t=20$ msec frame sent completely
    - $t=270$ msec frame arrives
    - $t=520$ msec best case of ack. Received
  - Sender blocked $500/520 \approx 96\%$ of time
  - Channel utilization $20/520 \approx 4\%$
Simplex Positive ACK with Retransmission (PAR)

- Finite buffer
- Noisy channel: frames may be damaged or lost.
- Positive acknowledgment to transmit the next data frame.
- Any frame has a sequence number, either 0 or 1
Simplex PAR: Effect of Errors

- The sender starts a timer when transmitting a data frame.
- If data frame is lost or damaged:
  - Receiver does not send an ACK
  - Sender times out and retransmits the data frame
Stop-and-Wait, lost ACK frame

- If the sender receives a damaged ACK, it discards it.
- When the timer of the sender expires, the sender retransmits frame 1.
- Receiver has already received frame 1 and expecting to receive frame 0 (R=0). Therefore it discards the second copy of frame 1.
Stop-and-Wait, delayed ACK frame

- The ACK can be delayed at the receiver or due to some problem.
- It is received after the timer for frame 0 has expired.
- Sender retransmitted a copy of frame 0. However, R = 1 means receiver expects to see frame 1. Receiver discards the duplicate frame 0.
- Sender receives 2 ACKs, it discards the second ACK.