The Data Link Layer

Jasleen Kaur

Fall 2016

Project Clarifications

- Milestones:
  - Project Proposals (to be presented) – end-Sept / beg-Oct
  - Mid-project Progress Updates – end-Oct
  - Final Project Presentations – end-Nov
  - Final Project Report – last day of exams

- Grading (flexible by ± 5%):
  - 2 early presentations – 10%
  - Final presentation – 20%
  - Final Report – 10%
  - Mid-project progress – 20%
  - Final progress – 40%
Today’s Overview

- Review of what you’ve read
  - Layered architecture, encoding, framing, BDP
- Error Detection
  - Checksums
- Media Access Control

Review Questions

- What is the Internet architecture’s layer model?
- What does layering mean? How is it implemented?
- What are the following encoding schemes?
  - NRZ, NRZI, Manchester, 4B/5B?
- What approaches are used to identify individual frames, if they can be of variable-length?
- What is statistical multiplexing?
- What is the delay-bandwidth product?
Protocol Layering in the Internet

- Application layer
  - Supporting network applications
    - ftp, SMTP, HTTP
- Transport layer
  - Host-host data transfer
    - TCP, UDP
- Network layer
  - Routing of packets from source to destination
    - IP, routing protocols
- Link layer
  - Data transfer between directly connected network elements
    - Ethernet, 802.11, SONET, ...
- Physical layer
  - The insertion of individual bits “on the wire”

Encapsulation Flow in Network Layers
The Data Link Layer

- Simplest way to create a network of nodes
  - Connect the nodes directly with a physical medium

- Point-to-point links:
  - Choice of Media
  - Encoding
  - Framing
  - Error detection
  - Error recovery

- Shared Media:
  - Media access control

ERROR DETECTION

Checksums and CRCs
What Causes Errors?

- Bit errors occur due to electrical interference or thermal noise
  - Detect whether errors have occurred in received data frames
    - Notify sender (for retransmission)
    - Or correct errors (based on error-correcting codes)
    - Or simply drop packet (to avoid wasting processing resources)

- Basic idea of error detection:
  - Use $k$ redundant bits to enable receiver to detect errors in a packet of size $n$

- Goals:
  - Strong error detection properties (detect different types of errors)
  - Small overhead ($k$ vs. $n$)
  - Efficient to implement

Naïve Approach: Packet Duplication

- Just append a duplicate copy of the frame
  0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1

- Error detection:
  - Errors that corrupt same positions in both frames will go undetected

- Overhead:
  - 100% overhead ($k = n$)

- Efficiency?
Smarter Scheme: One-dimensional Parity

- Insert an extra bit to every set of 7 bits to balance the number of “1”s
  - \[ 0 1 0 1 0 0 1 \mid 1 \]
- Error detection:
  - Catches all 1-bit errors
  - Can it detect 2-bit errors?
- Overhead:
  - 14 % (k = 1, n = 7)

Improve Efficiency: Two-dimensional Parity

- Compute parity also for each bit position across each of the bytes in a frame:
  - Add parity bit for parity byte also
    - \[ 0 1 0 1 0 0 1 \mid 1 \]
    - \[ 1 1 0 1 0 0 1 \mid 0 \]
    - \[ 1 0 1 1 1 0 0 \mid 1 \]
    - \[ 0 0 0 1 1 1 0 \mid 1 \]
    - \[ 0 1 0 1 0 0 1 \mid 1 \]
    - \[ 1 0 1 1 1 1 1 \mid 0 \]
    - \[ 0 1 0 1 1 0 0 \mid 1 \]
- Error detection:
  - Catches all 1/2-bit errors
  - Can it detect 3-bit errors?
  - Can it detect 4-bit errors?
  - Can 2-D parity be used to correct all 1-bit errors?
  - Can it be used to correct 2-bit errors?
- Overhead:
  - 30 % (k = 15, n = 49)
**Internet Checksum – Low Cost Detection**

- Add up all 16-bit words and send the sum
  - Use 16-bit ones complement arithmetic
    
    \[
    (-5) + (-3) = -(0101) + (-(0011)) = 1010 + 1100 = 0110 \text{ (+1 carry)} = 0111 = -(1000) = -8
    \]

- Error detection:
  - Weak (eg: one increment, one decrement)
  - Not used at data link layer (used by UDP, TCP, IP)

- Overhead:
  - Only \( k = 16 \) redundant bits for any size \( n \)

- Easy to implement in software

---

**Cyclic Redundancy Check (CRC)**

- 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, \( C(x) \)
  - e.g., \( C(x) = x^3 + x^2 + 1 \), \( k = 3 \)

- Transmit polynomial \( P(x) \) that is evenly divisible by \( C(x) \)
  - shift left \( k \) bits, i.e., \( M(x)x^k \)
  - subtract remainder of \( M(x)x^k \) / \( C(x) \) from \( M(x)x^k \)

- Receiver polynomial \( P(x) + E(x) \)
  - \( E(x) = 0 \) if 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 \( C(x) \)
**CRC: Selecting \( C(x) \)**

- **Can catch:**
  - 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 3 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.

- **CRC algorithm is easily implemented in hardware**
  - CRC-32 commonly used by link layer protocols \((k = 32)\)

---

**Error Detection: Summary**

- **Goals:**
  - Strong error detection properties (detect different types of errors)
  - Small overhead \((k \text{ vs. } n)\)
  - Efficient to implement

- **Ideas used:**
  - Redundancy: error detection implemented at multiple layers
    - Why?
  - Software-easy implementations at higher layers
    - Checksum
  - Hardware-easy implementations at lower layers
    - CRC-32