Chapter 5: The Data Link Layer

Our goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control: done!

- instantiation and implementation of various link layer technologies
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Hubs and switches
- 5.7 PPP
Link Layer: Introduction

Some terminology:
- hosts and routers are nodes
- communication channels that connect adjacent nodes along communication path are links
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a frame, encapsulates datagram

**data-link layer** has responsibility of transferring datagram from one node to adjacent node over a link
Link layer: context

- Datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
  - e.g., may or may not provide rdt over link

**Transportation analogy**

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- travel agent = routing algorithm
- transportation access = link layer protocol

Traffic lights, Airport control, Platform scheduling,
Link Layer Services

- **Framing, link access:**
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, dest
    - different from IP address!

- **Reliable delivery between adjacent nodes**
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?
Link Layer Services (more)

- **Flow Control:**
  - pacing between adjacent sending and receiving nodes

- **Error Detection:**
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame

- **Error Correction:**
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission

- **Half-duplex and full-duplex**
  - with half duplex, nodes at both ends of link can transmit, but not at same time
Adaptors Communicating

- Link layer implemented in "adaptor" (aka NIC)
  - Ethernet card, PCMCI card, 802.11 card

- Sending side:
  - Encapsulates datagram in a frame
  - Adds error checking bits, rdt, flow control, etc.

- Receiving side
  - Looks for errors, rdt, flow control, etc
  - Extracts datagram, passes to rcving node

- Adapter is semi-autonomous

- Link & physical layers
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Hubs and switches
- 5.7 PPP
- 5.8 Link Virtualization: ATM
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:**
Detect single bit errors

- \[ \text{d data bits} \rightarrow \text{parity bit} \]
- \[ 01110001101010110 \]

**Two Dimensional Bit Parity:**
Detect and correct single bit errors

- \[ \begin{array}{cccc}
  d_{1,1} & \cdots & d_{1,j} \\
  d_{2,1} & \cdots & d_{2,j} \\
  \vdots & \ddots & \vdots \\
  d_{i,1} & \cdots & d_{i,j} \\
  d_{i+1,1} & \cdots & d_{i+1,j}
\end{array} \]

- \[ \text{row parity} \]
- \[ \text{column parity} \]

- \[ \frac{\text{101011}}{\text{111100}} \]
- \[ \frac{\text{011101}}{\text{001010}} \]

- **no errors**

- \[ \frac{\text{101011}}{\text{101010}} \]
- \[ \frac{\text{011100}}{\text{010101}} \]

- **correctable single bit error**
Internet checksum

**Goal:** detect “errors” (e.g., flipped bits) in transmitted segment (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? More later ....