Chapter 5: Link layer

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  ▪ addressing, ARP
  ▪ Ethernet
  ▪ switches
  ▪ VLANs
Link layer: introduction

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

**data-link layer** has responsibility of transferring datagram from one node to *physically 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
- transportation mode = link layer protocol
- travel agent = routing algorithm
**Link layer services**

- **framing, link access:**
  - encapsulate datagram into frame, adding header, tailer
  - 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
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
- even parity and odd parity scheme

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

---

<table>
<thead>
<tr>
<th>d data bits</th>
<th>parity bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111000110101011</td>
<td>0</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>row parity</th>
<th>column parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_1,1</td>
<td>d_1,j</td>
</tr>
<tr>
<td>d_2,1</td>
<td>d_2,j</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>d_i,1</td>
<td>d_i,j</td>
</tr>
<tr>
<td>d_{i+1,1}</td>
<td>d_{i+1,j}</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Parity Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1 1</td>
<td>1 0 1 0 1 1</td>
</tr>
<tr>
<td>1 1 1 1 0 0</td>
<td>1 0 1 1 0 0</td>
</tr>
<tr>
<td>0 1 1 1 0 1</td>
<td>0 1 1 0 1 1</td>
</tr>
<tr>
<td>0 0 1 0 1 0</td>
<td>0 0 1 0 1 0</td>
</tr>
</tbody>
</table>

- no errors

---

**Correctable single bit error**
Internet checksum (review)

**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?*
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 $G$ (modulo 2)
  - receiver knows $G$, divides $<D,R>$ by $G$. If non-zero remainder: error detected!
  - can detect all burst errors less than $r+1$ bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)

$$D \times 2^r \text{ XOR } R$$ mathematical formula
CRC example

want:
\[ D \cdot 2^r \text{ XOR } R = nG \]
equivalently:
\[ D \cdot 2^r = nG \text{ XOR } R \]
equivalently:
if we divide \( D \cdot 2^r \) by \( G \), want remainder \( R \) to satisfy:

\[
R = \text{remainder}\left[ \frac{D \cdot 2^r}{G} \right]
\]

All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction.
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
Multiple access links, protocols

two types of “links”:

- **point-to-point**
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host

- **broadcast (shared wire or medium)**
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN

(shared wire (e.g., cabled Ethernet)  
shared RF (e.g., 802.11 WiFi)  
shared RF (satellite)  
humans at a cocktail party (shared air, acoustical)
Multiple access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

**multiple access protocol**

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination
An ideal multiple access protocol

given: broadcast channel of rate R bps
desiderata:

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   • no special node to coordinate transmissions
   • no synchronization of clocks, slots
4. simple
MAC protocols: taxonomy

three broad classes:

- **channel partitioning**
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use

- **random access**
  - channel not divided, allow collisions
  - “recover” from collisions

- **“taking turns”**
  - nodes take turns, but nodes with more to send can take longer turns
Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle
Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle
Random access protocols

- when node has packet to send
  - transmit at full channel data rate $R$.
  - no *a priori* coordination among nodes
- two or more transmitting nodes $\Rightarrow$ “collision”,
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA
**Slotted ALOHA**

**assumptions:**
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

**operation:**
- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success
**Pros:**

- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only slots in nodes need to be in sync
- Simple

**Cons:**

- Collisions, wasting slots
- Idle slots
- Nodes may be able to detect collision in less than time to transmit packet
- Clock synchronization
Suppose: $N$ nodes with many frames to send, each transmits in slot with probability $p$.

- Probability that a given node has success in a slot: $p(1-p)^{N-1}$
- Probability that any node has a success: $Np(1-p)^{N-1}$

Max efficiency: find $p^*$ that maximizes $Np(1-p)^{N-1}$

For many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as $N$ goes to infinity, gives:

\[ \text{max efficiency} = 1/e = .37 \]

Efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

At best: channel used for useful transmissions 37% of time!
Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at $t_0$ collides with other frames sent in $[t_0-1,t_0+1]$
**Pure ALOHA efficiency**

\[ P(\text{success by given node}) = P(\text{node transmits}) \cdot \]
\[ P(\text{no other node transmits in } [t_0-1, t_0]) \cdot \]
\[ P(\text{no other node transmits in } [t_0, t_0+1]) \]

\[ = p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} \]
\[ = p \cdot (1-p)^{2(N-1)} \]

... choosing optimum \( p \) and then letting \( n \rightarrow \infty \)

\[ = 1/(2e) = .18 \]

even worse than slotted Aloha!
CSMA (carrier sense multiple access)

**CSMA:** listen before transmit:
- if channel sensed idle: transmit entire frame
  - if channel sensed busy, defer transmission

- human analogy: don’t interrupt others!
CSMA collisions

- **collisions can still occur:** propagation delay means two nodes may not hear each other’s transmission

- **collision:** entire packet transmission time wasted
  - distance & propagation delay play role in determining collision probability
CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

- human analogy: the polite conversationalist
CSMA/CD (collision detection)
“Taking turns” MAC protocols

channel partitioning MAC protocols:
- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, \( \frac{1}{N} \) bandwidth allocated even if only 1 active node!

random access MAC protocols
- efficient at low load: single node can fully utilize channel
- high load: collision overhead

“taking turns” protocols
look for best of both worlds!
“Taking turns” MAC protocols

polling:

- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)
“Taking turns” MAC protocols

token passing:

- control token passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)

Link Layer 5-31
Summary of MAC protocols

- **channel partitioning**, by time, frequency or code
  - Time Division, Frequency Division

- **random access** (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11

- **taking turns**
  - polling from central site, token passing
  - bluetooth, FDDI, token ring